For office use only

T1 _____

T2 _____

T3 _____

T4 _____

Team Control Number

2022031

For office use only

F1 _____

F2 _____

F3 _____

F4 _____

2022

The International Mathematical Modeling Challenge (IM^2C) Summary Sheet

(Your team's summary should be included as the first page of your electronic submission.)

Airplane boarding strategy and, to a lesser extent, disembarking strategy are widely studied due to their importance in the aviation industry. The problem of finding the optimal, i.e., quickest, boarding and disembarking methods are studied widely. Theoretical results have been produced, and some simulations are conducted. But most studies lack of consideration of practicality in real scenario, which we hope to provide.

Boarding and disembarking method are considered as a list of priorities, which we assign to each passenger. Passenger who is higher in priority must enter/leave the plane before passengers with lower priority can begin their processes of going to their seat/leaving their seat.

For the boarding model, we transform a boarding method to the ideal boarding sequence. Then, we add the element of imperfection by changing the positions of boarding sequence locally (queue jumping) at a specific rate. That rate of queue jumping is proportional to the complexity of the boarding method, which we quantify in our model. Some passengers are also assigned to be late passengers that will arrive at the back of the line. We use a probabilistic cellular automata model to simulate the boarding process on the plane. We make a passenger an automaton. There are two actions of an automaton: moving and stowing the bag, for they are the only actions significant to boarding time. These actions are performed according to the state of the automaton. The most time-consuming instances are when there is/are seated passenger(s) blocking another passenger in the same row from seating. Boarding time ends when all passengers are seated.

For the disembarking model, a disembarking method is used to specify passengers that are allowed to leave before some other passengers. However, some passengers are late-disembarking passengers. These passengers will not leave their rows unless all other passengers have left, causing passengers that need to pass them to get to the aisle to be unable to move as well. We use a similar cellular automata simulation as with the boarding process. Disembarking time ends when all passengers leave the plane.

We perform sensitivity tests with the boarding model and find that it is insensitive to the change in average bag stowing time. As with queue-jumping ratio sensitivity, we find that complicated method is susceptible to increase in queue jumping. Simple boarding methods, however, are completely unaffected by such increase. These two models are used with three aircraft (Narrow-body Aircraft, Flying Wing Aircraft, Two-entrance Two-aisle Aircraft) and various boarding and disembarking methods.

Steffen's method and random disembarking method are found to be, respectively, optimal boarding method and optimal disembarking method for Narrow-body Aircraft. For Flying Wing Aircraft and Two-entrance Two-aisle Aircraft, the optimal boarding methods are, respectively, modified boarding by seat and boarding by seat method.

Moreover, we model the seating in the pandemic situation and find the optimal boarding methods. We find that the optimal boarding method is boarding by luggage size for all patterns. Lastly, we evaluate our model. We find that the strengths of our models are their adaptability, practicality, and low time complexity. But their limitations are the lack the consideration of passenger groups and the sensitivity to complexity factor.

Contents

1 Introduction

1.1 Background

It is a truth universally acknowledged that the experience of boarding an airplane as a passenger is tiresome and time-consuming. For airlines, the more time spent on disembarking previous passengers and boarding new passengers means an increase in turnaround time, consequently a loss in flying slot and revenue. Therefore, aircraft boarding and disembarking strategy is widely studied mathematically [1, 2], computationally [3, 4, 5, 6, 7], and in field studies [8, 9] to improve the efficiency of both processes.

In those studies, many models are introduced to simulate the behaviors of passengers in boarding/disembarking. Most models focus on the process of passengers travelling to their seat in the plane, taking for granted that the queue is predetermined by the boarding strategy. In these models [4, 10, 11, 12], the best-performing boarding methods are always complicated methods that order every passenger in the queue, unlike the commonly-used methods for airlines that have larger groups. On one hand, these methods optimise the boarding time in the ideal situation because it minimises cases where many passengers having to wait for a passenger to stow his/her bags, which are the main cause of long boarding time [13]. On the other hand, these methods are extremely difficult to implement in reality, due to many arrangements in boarding gate and many possibilities for errors in terms of queue-jumping.

We recognise the fact that the queuing process in the real world is not ideal. Therefore, in our model, we focus on both the queuing and travelling processes. The complexity of boarding methods are factored directly into our models, with more complicated methods having more possibilities for queue-jumping. Furthermore, we also consider the cases where some passengers are forced into the end of the boarding queue because they arrive late at the boarding gate. Lastly, we use probabilistic cellular automata model to simulate the travelling process. As for the disembarking process, we use probabilistic cellular automata in a similar way to the boarding model.

Both the boarding and disembarking models are then used to determine the optimal boarding and disembarking methods in real practice for three different aircraft. Also, according to the real-world situation of the pandemic, we also modify our models to the social-distancing seating arrangements. Our models are then used again to find the optimal boarding and disembarking method during the pandemic.

The paper is organised as follows. Section 1 is the introduction. Section 2 is term definitions and general assumptions. Section 3 is detailed explanation of both models. Section 4, 5, and 6 are when we put our models to test with three aircraft. Section 7 concerns how the pandemic affects boarding and disembarking strategies. Section 8 discusses the advantages and disadvantages of our proposed models. Section 9 is the conclusion. Section 10 is the letter to an airline consecutive explaining our findings to a non-mathematical audience.

1.2 Problem Restatement

We will address four problems in this modeling:

1. Construct mathematical models to find boarding and disembarking time of an aircraft.

2. Use the boarding model to evaluate five methods of boarding a narrow-body aircraft: random, boarding by section, boarding by seat, and two more. Also, perform sensitivity analysis of the boarding model.

3. Propose and justify the optimal boarding and disembarking methods for three aircraft: a narrow-body aircraft, a flying wing aircraft, and a two-entrance, two-aisle aircraft.

4. Considering the pandemic situation, make adjustments to the optimal boarding and disembarking methods for the three aircraft.

2 Preliminaries

2.1 Definitions

Term	Definition
Boarding time	Time since the first passenger enter the plane until all passengers are seated
Disembarking time	Time since the first passenger begins to move until all passengers leave the plane
Boarding method	A map that assigns each passenger into a group, called priority, represented by a positive integer, whereby a passenger with smaller number will be before a passenger with larger number in the ideal boarding sequence
Ideal boarding sequence (IBS)	A queue of passengers that is produced from boarding method, before accounting for passengers not following the prescribed queue
Real boarding sequence (RBS)	A queue of passengers that comes from an IBS, after accounting for passengers not following the prescribed queue
Disembarking method	A map that assigns each passenger into a group, called priority, represented by a positive integer, where a passenger with larger number must wait until all passengers with smaller number moved to the aisle before moving to the aisle
Optimal boarding/ disembarking method	The boarding/disembarking method with with smallest boarding/ disembarking time

2.2 General Assumptions

The following are assumed throughout this modeling:

1. Assumption: There are only two cases of passengers not following the prescribed boarding method (entry queue): queue-jumping or late-arriving.
   Justification: The queue-jumping case accounts for many situations such as passenger cannot follow the boarding method, or was simply skipped or skips other passenger. Late-arriving passengers are considered separately because they make up a significant proportion of all passengers. Note that the late-arriving passengers are those who arrive when the boarding process already began, but before it ends.

2. Assumption: The aisle is *narrow* i.e., the width of the aisle only allows for one passenger. Consequently, the passengers cannot swap their positions in the aisle.
   Justification: This is assumed in the Problem Statement.

3. Assumption: All passengers move with a constant speed.
   Justification: In the real world, moving speed in the queue is very slow, so every passenger would move at a roughly equal slow speed.

4. Assumption: In boarding, there are only two possible actions: moving (in the aisle and traverse the row) and stowing the bag. In disembarking, these are moving (in the aisle and traverse the row) and collecting the bag.
   Justification: These are the only actions that significantly affect the boarding/disembarking time.

5. Assumption: Once on the plane, every person behaves *rationally* i.e., everyone tries to get to their seat with a predesignated path. There is no unnecessary stop, reversing path, or passenger getting to a wrong seat.
   Justification: This is for the ease of simulation. In the real world, it is very difficult to move back when there are other passengers waiting behind due to the narrow width. Stopping time can be assumed to be accounted for already in the average moving speed. Passengers getting to the wrong seat happen rare enough to be neglected.

6. Assumption: If a seated passenger sees another passenger whom he/she blocks the path to another passenger's seat, he/she will step out of their seat to allow another passenger to travel to his/her seat as soon as possible without delay.
   Justification: This is for the ease of simulation. Passengers refusing to unblock other passenger are considered discourteous and therefore happen rare enough to become negligible.

3 Model Construction

3.1 Parameters

The following table describes the parameters used in the model.

Variable	Meaning
N	Total number of passengers in the model
N_A	Maximum number of passengers of the Narrow-Body Passenger Aircraft
N_B	Maximum number of passengers of the Flying Wing Passenger Aircraft
N_C	Maximum number of passengers of the Two-Entrance, Two Aisle Passenger Aircraft
N_{3F}	Maximum number of passengers of the front section (business class) of the Two-Entrance, Two Aisle Passenger Aircraft
BT	Total aircraft boarding time
DT	Total aircraft disembarking time
C	Boarding/Disembarking method complexity factor
M	Number of boarding groups based on priority
m_i	Number of passengers whose priority group is i
R_J	Ratio between queue-jumping passengers and all passengers
R_L	Ratio between late/late-disembarking passengers and all passengers
$R_{J,max}$	Maximum possible queue jumping ratio
r	Range of queue jumping
$QJ(i,j)$	Probability that, in case they does not follow the prescribed boarding/ disembarking method, the passenger q_i queue jumps to position j in the queue
k, λ	Parameters in Weibull distribution

3.2 Boarding Model

3.2.1 Summary of Boarding Model

The model of boarding can be split into two major components: queuing model and travelling (to seat) model. We first convert boarding method to IBS, then RBS, which is sent to be simulated using cellular automata. The flowchart in Figure 1 summarises our model. The simulation is done in Python, the code of which can be found in the appendix.

3.2.2 Boarding Method and Ideal Boarding Sequence (IBS)

The boarding method divides passengers into M priorities (groups), namely $1, 2, \ldots, M$. Each passenger is assigned a priority $p \in \{1, 2, \ldots, M\}$. The IBS of passengers entering the plane will be determined by the priorities of passengers, with passengers with smaller priority number (i.e., higher priority) before passengers with larger priority number. Within the same priority, passengers are randomly listed in the queue.

As an example, consider a boarding method of a small aircraft in Figure 2, which can be turned into boarding sequence. The two shown on the right are one of many possibilities for boarding sequence from the priorities assigned by the boarding method.

The rationale behind this process is based on the real world. At boarding gates, depending on the boarding method, the cabin crews will call passengers in group (priority) e.g., "...Passengers with tickets number A1 to A33, please come to the counter with your boarding pass for ticket checking

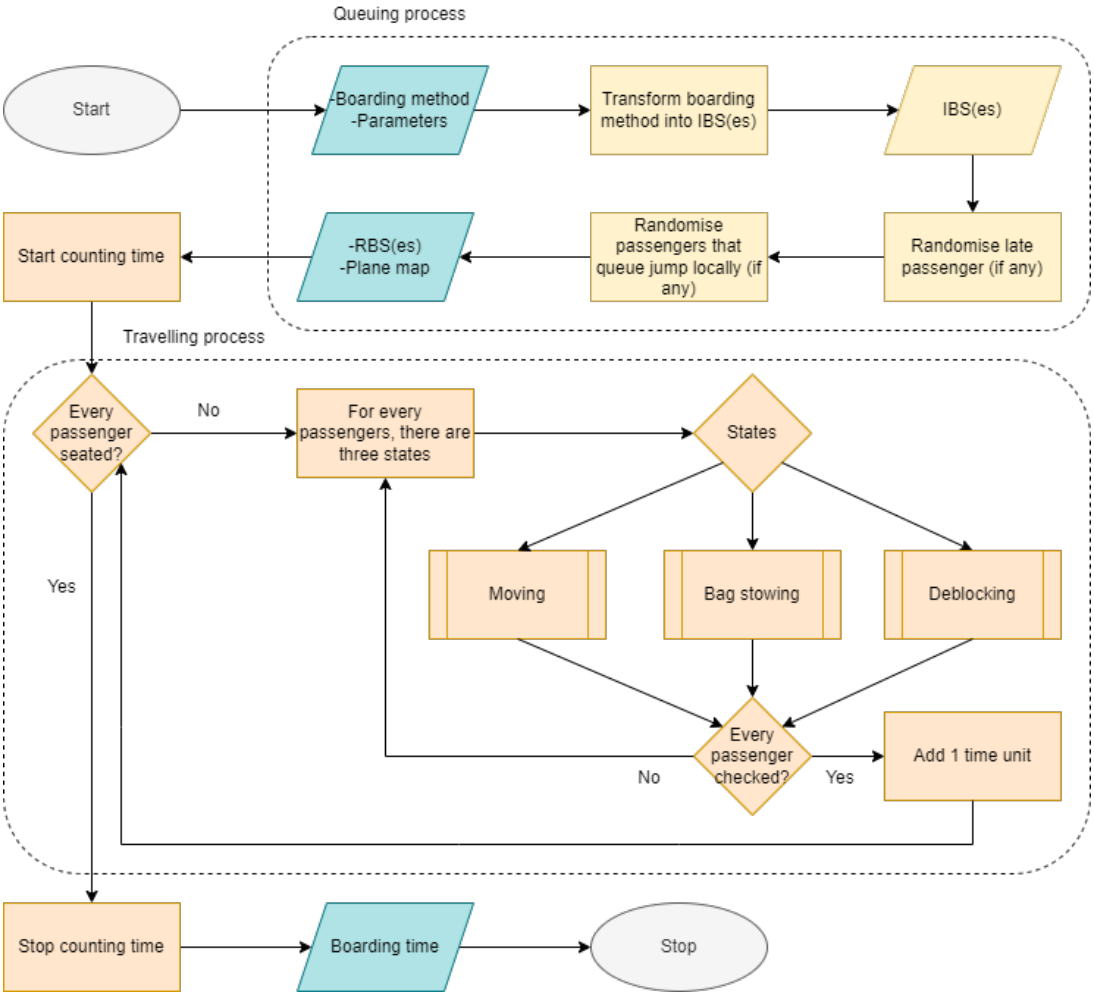

Figure 1: Flowchart of the boarding model

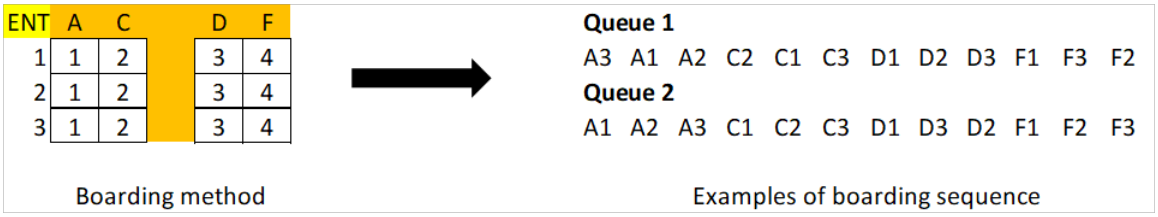

Figure 2: Turning a boarding method into boarding sequence.

and boarding. ...” Passengers whose seats are A1 to A33 will make up a line randomly. When the next group is called, the queue will be random within the group as well.

3.2.3 Boarding Method Complexity

In reality, it is very difficult to organise a boarding queue that absolutely follow a boarding method. Two factors make this nearly impossible: late passengers and passengers queue jumping (either forward or backward) locally. Obviously, the ratio of late passengers R_L is constant across all boarding methods. However, the ratio of queue-jumping passengers R_J is set to be dependent on the complexity of the boarding method. This is because the more complex the boarding method is, the more difficult it is for passengers to follow, which lead to more unintentional queue-jumping [14]. The complexity of boarding method can also frustrate some passengers and make them decide to ignore the queue intentionally.

In our model, we use complexity factor C for each boarding method to quantify its complexity. We define

$$C = \frac{\ln M}{\ln N} \tag{1}$$

where M and N are the number of priorities and the number of passengers, respectively. And from the previously stated reason, we write R_J as a multiple of C.

$$R_J = R_{J,\max} C \tag{2}$$

where $R_{J,\max}$ is the maximum possible queue-jumping ratio.

The number of priorities M is suitable for determining complexity because it is the number of times the cabin crew has to call passengers. The use of logarithmic scale is to account for the difference in significance between increasing from two priorities to three and from 50 priorities to 51. Division by $\ln N$ normalises C. The random boarding method has complexity $C = 0$ since $M = 1$. So, $R_J = 0$ for random boarding method. This agrees with our intuition that it is not possible to break the rules in random boarding method since there is no rule in the first place. At the other extreme, when the queue is completely determined (there are N priorities and each priority has one passenger), such as with Steffen's method in [4], we have $C = 1$ since $M = N$.

3.2.4 Late-arriving passengers

In the real world, there will be some passengers that arrive at the boarding gate after the queue is already made, or the boarding process has already started. These passengers will be forced to be at the back of the queue, regardless of their supposed position in the boarding method. The number of late passengers is $R_L N$, which are uniformly randomly selected. All of them are placed at the end of the RBS.

3.2.5 Queue jumping and Real Boarding Sequence (RBS)

Now, we will consider the queue-jumping behavior of queue-jumping passengers. Queue jumping can happen in two times: before boarding pass check and after. There are many causes for both cases,

but an example would be when walking from boarding gate to the plane, there are some passengers who walk significantly faster or slower than the others, causing him/her to be ahead or behind his/her actual queue. We employ binomial distribution to model this behavior, because the locality aspect of the binomial distribution better represents the real world, as opposed to globalised manipulation such as when $QJ(i,j)$ is a uniform distribution. It also has a positive side-effect in more impact of randomisation, as the numerical study [6] concludes that in a multicolumn plane, the boarding time is more sensitive to localised disorder than globalised disorder.

There will be $R_J N$ queue-jumping passengers, all of them being selected uniformly randomly. We let r be an even number that is the range of the distribution. The probability that the passenger q_i, if he/she queue jump, queue jumps to position j in the queue is

$$QJ(i,j) = \frac{1}{2^r}\binom{r}{r/2+i-j}. \tag{3}$$

When a passenger q_i queue jumps to position j, the passenger q_j and all passengers between q_i and q_j move to position will move back one position, if $i > j$; if $i < j$, they will move forward one position. In our model, we set $r = 12$. Thus, every queue-jumping passenger can only move between 6 positions forwards and 6 backwards.

The resulting sequence after queue-jumping is called the real boarding sequence. The RBS is the queue of passengers at the entrance of the plane.

3.2.6 Aisle and Row Traverse Speed

We assume that the speed of moving in the aisle and the row is constant. A study [15] shows that most economic class seats have seat spaces between 71-83 cm and width between 43-47 cm. So, we will assume that seat space of our model is 74 cm (29 in). From [16], the average row traverse speed when seat space is 29 in is 0.4 m/s and the average aisle speed is 0.52 m/s. We define 1 time unit (t.u.) equals to the time it requires to travel from one row to the next in the aisle. Therefore,

$$1 \text{ time unit} = 1.42 \text{ s}. \tag{4}$$

Also, we can calculate the time to travel one seat in the same row, and find that it is equal 1.1 s. For the sake of simplicity in simulation, we will assume that the row traverse speed is equals to 1 t.u. as well.

3.2.7 Bag Stowing Time

The range of bag stowing time in the real world is large (minimum at 1.9 s to maximum at 10.7 s [16]), therefore it is inappropriate to fix it to a constant. According to [17], the bag stowing time follows Weibull distribution. We will use a discretised version of Weibull distribution as the distribution function of our randomised bag stowing time. Weibull distribution has probability density function

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0. \end{cases} \tag{5}$$

The mean and standard deviation of bag stowing time is 7.0 s and 1.7 s [16], respectively. We convert this into t.u., so we get the mean $\bar{t} = 4.93$ and standard deviation $\sigma = 1.20$. We use these values to calibrate λ and k according to equations from [18]:

$$k = \left(\frac{\sigma}{\bar{t}}\right)^{-1.086} \text{ and } \lambda = \frac{\bar{t}}{\Gamma(1+1/k)}. \tag{6}$$

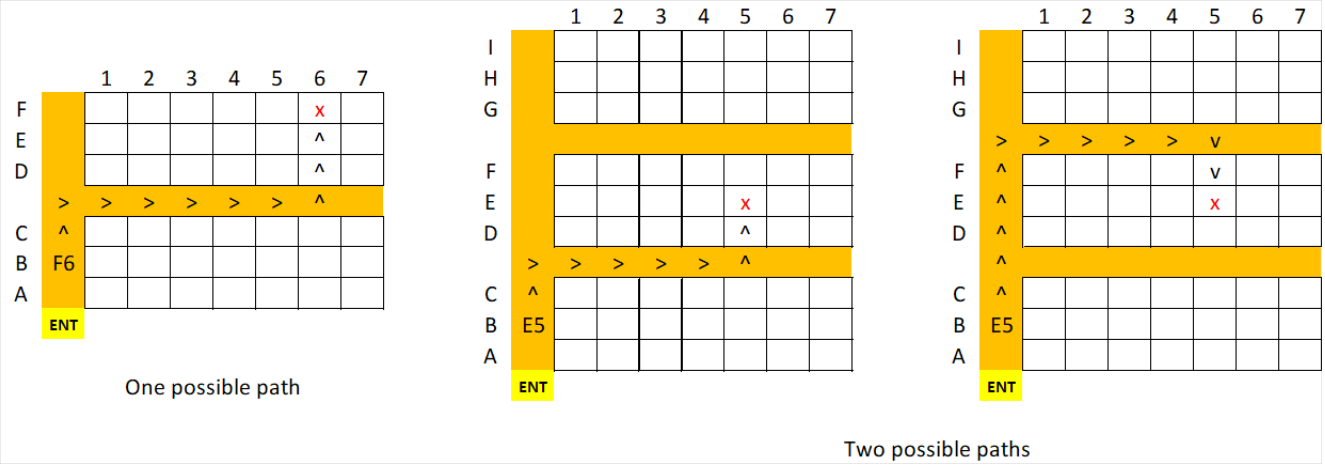

Figure 3: (Left) A seat with fixed simplest path, which almost always the case. (Middle & Right) A seat that has two simplest paths and the paths.

where $\Gamma(x)$ is the well-known gamma function $\Gamma(x) = \int_0^\infty e^{-u}u^{x-1}\mathrm{d}u$. A numerical calculation finds $k = 5.153$ and $\lambda = 7.774$. The code of this numerical calculation can be found in the appendix.

We use the parameters $k = 5.153$ and $\lambda = 7.774$ in equation 5 to get our probability distribution. Then, every passenger is randomised a bag stowing time according to this distribution. However, since our program has integer time scale, we round the bag stowing time for each passenger to the nearest integer.

3.2.8 Algorithm for Path Finding

To model an aircraft, we consider them as grids of seat, aisle, and walking space from the entrance to aisle. To simplify, we call aisles and walking space from the entrance to aisles the "aisle grids." A seat is an 1 x 1 grid, and it is arranged according to the plane's seat plan. Aisle and walking space has width 1, and are placed according to their positions in the plane. The aisle grids are also extended by 2 grids at the back of the plane, to allow for passengers in the back row to move when unblock another passenger in the same row. The grids of each aircraft (Aircraft I, Aircraft II, Aircraft III) will be presented in their respective section.

The algorithm for path finding is simple. Every automaton (passenger) will follow the "simplest" path to their seat. Passengers will try to avoid traversing seat rows; the only time they will traverse in the rows is when their seat is in that row. Otherwise, passengers will travel to the aisle closest to their seat. In almost all cases, there is only one simplest path. However, there could be two simplest paths if a seat is in the middle of two aisles. In that case, the path that a passenger will take would be randomised between the two simplest paths. Figure 3 not only illustrate the mentioned simplest paths, but also give an example the grids of a plane that we discussed earlier.

3.2.9 Algorithm for Moving, Bag Stowing, and Unblocking

Boarding time begins when the first passenger arrives at the plane. In our simulation, the first passenger will be spawned at the entrance, and move according to his/her path. The default state of every spawned passenger is moving. When the first passenger step out of the entrance, the second passenger will spawn at the entrance. When the second passenger move, the third will spawn, and so on.

There are four states for every automaton: moving, bag stowing, unblocking, and seated. We will describe the four states and how passengers switch between them.

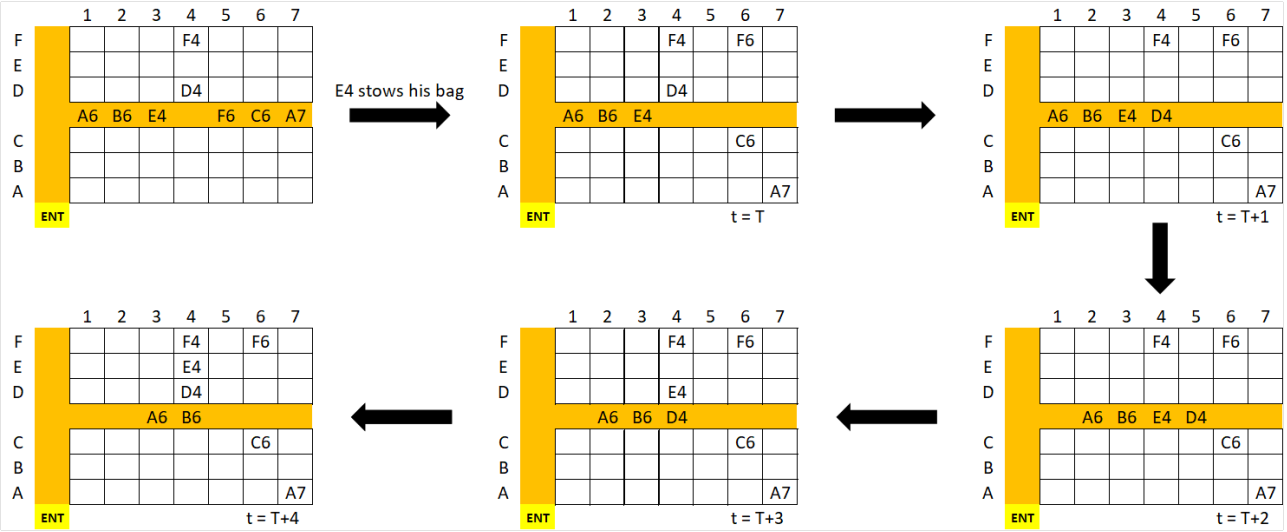

Figure 4: The unblocking process.

- MOVING: A passenger in the moving state will move one grid every time unit in the path to his/her seat, if possible.

- BAG STOWING: A passenger will transfer from moving to back stowing when he/she is in the aisle grid in the same row as his/her seat. But, if the line before the passenger is unmoving, he/she can transfer to this state in the aisle block one row before his/her seat's row. The passenger will spend some time unit determined by the discretised Weibull distribution in Subsubsection 3.2.7.

  During this state, the passenger remains *unmoving*. Since passengers cannot travel through other passengers, all passengers behind an unmoving passenger in the line must wait for him/her to change his/her state before the line can move once again. When the time required for bag stowing has passed, the passenger in this state will go back to the moving state again.

- UNBLOCKING: A passenger will transfer from the seated state to the unblocking state if he/she sees that there is a *blocked* passenger i.e., a passenger whose seat is in the same row and requires him/her to move to get to the passenger's seat, one row before his/her and has already stowed his/her bag.

  The passenger in this state would wait until there is a space in the aisle two grids in front of the blocked passenger and move to that space. Then, the blocked passenger will move to his/her seat, before the unblocking passenger move back to their seat. If there are two unblocking passengers in the same row, then they will wait until there is one space, then the outer unblocking passenger will move to that space. Then, he/she will wait until the space before him/her is empty and move to that space, and the inner unblocking passenger will move to the space previously occupied by the outer unblocking passengers. Then, the blocked passenger, the inner unblocking passenger, and the outer unblocking passenger will move to their respective seats, in that order. During the entire process, the queue behind the blocked is blocked from moving. Figure 4 shows the timeline of the process.

  Indeed, it would be less time-consuming if we allow the unblocking passengers to move before the blocked passenger stows his/her bag. However, few passengers in real life would voluntarily step out and stand waiting for the blocked passenger to stow his/her bag.

- SEATED: A passenger is seated if he/she is in his/her seat.

The algorithm terminates when every passenger is in his/her seat.

3.3 Disembarking Model

3.3.1 Summary of Disembarking Model

The disembarking model is developed using the similar notion of priorities as the boarding model. We start with the ideal disembarking method and add an element of imperfection by including late-disembarking passengers. We then run cellular automata simulation to simulate the disembarking process. Find the code in the appendix.

3.3.2 Disembarking Method

Disembarking method divides passengers into M priorities. Passenger with lower priority (larger number) must wait until all passengers with higher priority (smaller number) left the plane first. Consequently, it is required that, for every row, the priority of passenger closer to the aisle must be smaller or equal than that of passenger further from the aisle.

3.3.3 Late-disembarking Passengers

In real life, few passengers will choose to remain in their seats until other passengers have left to avoid crowding during disembarking process. There are $R_D N$ passengers that are uniformly randomised to be the initial late-disembarking passengers.

However, if a passenger choose to remain in their seat, then all passengers whose paths to the aisle are blocked by him/her are unable to move as well. So, these passengers are labelled late-disembarking passengers also. The means by which we force late-disembarking passengers to leave after other passengers have left the plane is by assigning them priority $M + 1$.

3.3.4 Bag Collection Time

For the same reason as and with the same method with the boarding simulation, we use discretised Weibull distribution to determine bag collection time. The mean and standard deviation of bag collection time is 7.0 s and 1.8 s, respectively [16]. Hence, $\bar{t} = 4.93$ and $\sigma = 1.27$. Therefore, we can calibrate Weibull distribution with $k = 4.568$ and $\lambda = 7.750$ via equation 6. The randomised time is rounded to the nearest integer.

3.3.5 Disembarking Simulation

In our cellular automata model, we use the same grids as in boarding simulation. Passengers will go to the closest aisle and move in that aisle to the exit. Now, there is a possible case where passengers between two aisles are blocked in the side of the closer aisle but can move to the further aisle. However, these passengers still would not move to the further aisle. This is because they have their luggage stored in the overhead bin in the closer aisle when the boarded the plane. This reality is reflected in our fixed model of path finding.

Each automaton has four states: waiting, bag collection, moving, and left.

- Waiting: Every passenger starts out waiting. Then, the passengers with priority 1 will move to the aisle and turn to bag collection state. If there are two passengers with priority 1 in both sides of the aisle, the passenger that will go to the aisle first will be uniformly randomised. Once all passengers with priority 1 left the plane, passengers with priority 2 can turn to bag collection state, and so on until passengers with priority $M + 1$. In this state, passenger will move in his/her row closer to the aisle if possible.

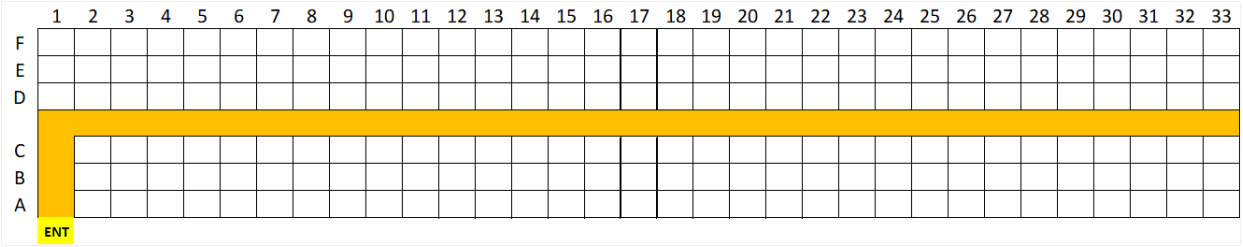

Figure 5: The grid map of Aircraft I.

- Bag Collection: Passengers in the bag collection state will be in the aisle in the same row as their seat. They will spend a in this state. During this state, passengers will be unmoving. They will then transfer to the moving state after time predetermined by Weibull distribution
- Moving: In this state, passengers will move one block every time unit according to their path to the exit, there is no unmoving passenger blocking the path.
- Left: Passenger who reaches the exit will leave the plane. He/she will be removed from the grids.

The algorithm terminates when all passengers leave the plane.

4 Aircraft I: Narrow-body Aircraft

Aircraft I is an aircraft with 195 seats, so $N_A = 195$. It has 33 rows and 6 columns of seats. The grids of this aircraft is shown in Figure 5.

4.1 Boarding Methods

The following boarding methods will be tested by our model:

1. Random Boarding: Random boarding method assigns every passengers as priority 1. The complexity of this method is $C = 0$.
2. Back-to-Front Boarding: This boarding method divides passengers into 3 priorities. Passengers whose seat is in row 23-33, row 12-22, and row 1-11 are assigned to be in priority 1, 2, and 3 respectively. This method has complexity $C \approx 0.208$.
3. Back-Front-Middle Boarding: Similar to the back to front method, passengers are divided into 3 priorities: row 23-33, row 1-11, and row 12-22 are priority 1, 2, and 3, respectively. It also has complexity $C \approx 0.208$.
4. Boarding by Seat: The cabin crew divides passengers into 3 priorities, window seats, middle seats and aisle seats. The complexity of this method $C \approx 0.208$.
5. Steffen's Method: This is a method in [4]. Passengers are completely ordered, as shown in Figure 6. Steffen's method is recommended by many works because it minimised the time wasted on bag stowing time by allowing multiple passengers to stow their bags simultaneously. However, this method is very complex for passengers, as shown by the complexity $C = 1$.
6. Boarding by Luggage Size: This is a method inspired by [10]. Firstly, the airline checks the passenger's carry-on bags and assigns the passengers into group 1, group 2 and group 3 based on bag stowing time 0-3 t.u., 4-6 t.u. and more than 6 t.u., respectively. Each of which

will be assigned into three subgroups, which are window seats, middle seats, and aisle seats. A passenger in ith group and jth subgroup will be assigned into priority $3(i-1)+j$. Figure 7 is an example of this method, but note that the actual priorities depend on the randomised luggage stowing time for each passenger. This method has complexity $C \approx 0.417$.

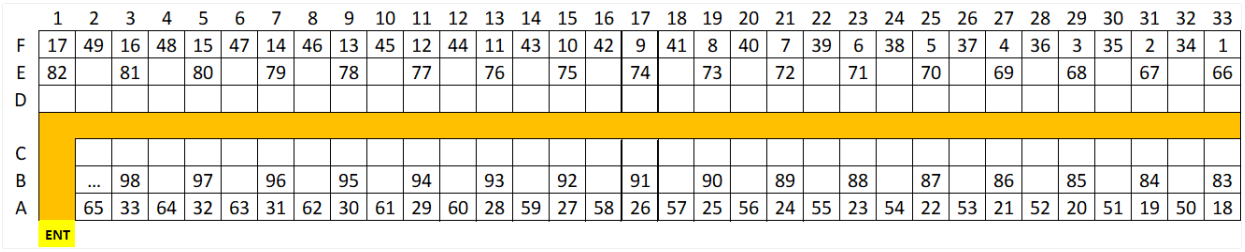

Figure 6: Steffen's method on Narrow-Body Passenger Aircraft

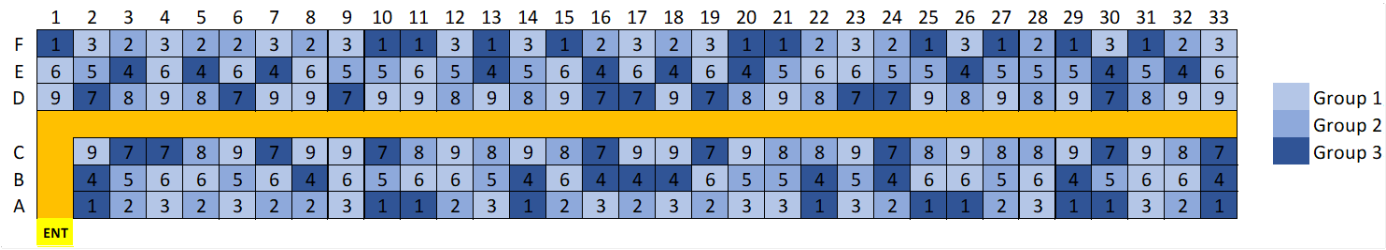

Figure 7: Boarding by luggage size method on Narrow-Body Passenger Aircraft

4.2 Optimal Boarding Method

We want to determine the optimal boarding method for Aircraft I. The parameters of our simulation are $R_L = 10\%$ and $R_{J,\max} = 50\%$. Knowing $R_{J,\max}$ and C, we can multiply them to get R_J for each boarding method.

We run the simulation 80 times for each method. Figure 8 and 9 are illustrations of our simulation of boarding by seat method and Steffen's method, respectively. Average boarding time, percentage of random boarding time, standard deviation, practical minimum (5th percentile), and practical

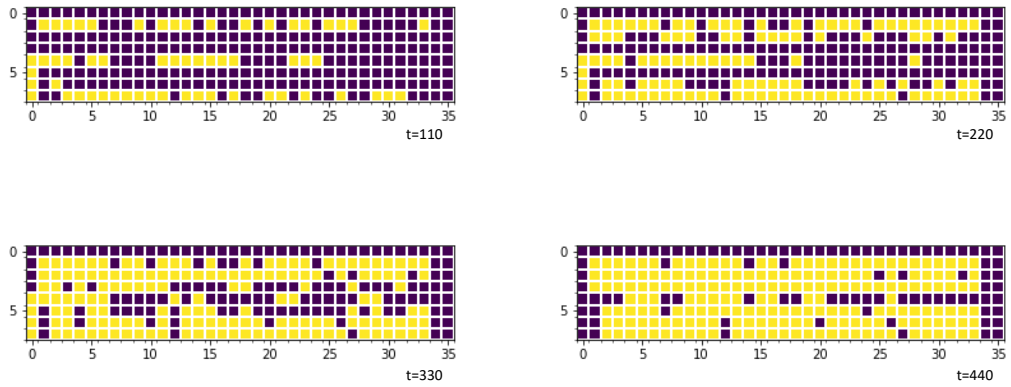

Figure 8: An example of the progression of a simulation using boarding by seat method. Note that $BT = 487$.

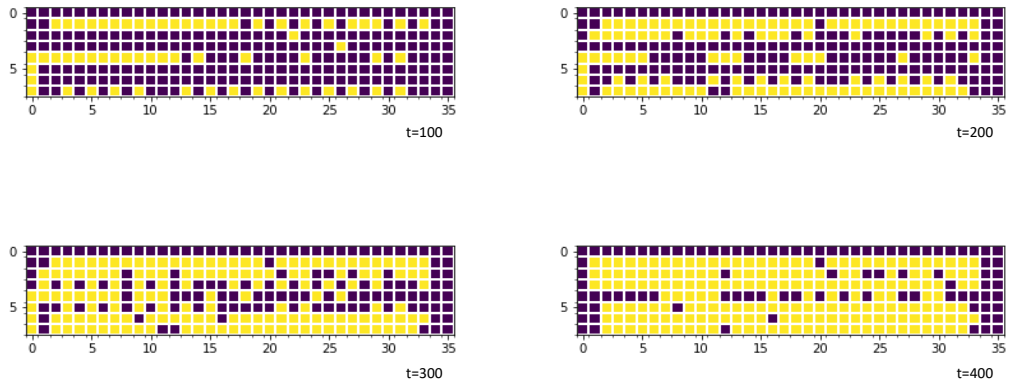

Figure 9: An example of the progression of a simulation using Steffen's method. Note that $BT = 436$.

maximum (95th percentile) of the data is shown in the following table. The unit for all data is time unit.

Boarding method	**Average** BT	**%**	**S.D.**	**Practical min.**	**Practical max.**
Random boarding	560	100	24.5	522	609
Back-to-front boarding	690	123	24.0	652	732
Back-front-middle boarding	677	121	22.9	641	718
Boarding by seat	484	86	17.5	455	511
Steffen's method	443	79	16.1	416	475
Boarding by luggage size	485	87	16.8	458	518

The result shows that **Steffen's method** is the optimal boarding method for this aircraft, with the average boarding time of 443 t.u. = 629 s. Boarding by seat and boarding by luggage size closely follow. We can also see that boarding by seat method and boarding by luggage size method are almost identical in all measures, suggesting that the additional effort to arrange passengers based on their luggage size do not improve boarding time at all. On the opposite side, back-to-front boarding and back-front-middle boarding, methods used by many airlines, is even less efficient than random boarding. Our result agrees with the experimental study [9] in terms of ranking, but boarding time in that study is much lower than ours due to the difference in airplane's size.

4.3 Sensitivity Analysis

4.3.1 Variation of Bag Stowing Time

We test the sensitivity to bag stowing time by varying the average bag stowing time $\bar{t}$ from 1 to 10 t.u., with each step equals 1 t.u. The standard deviation σ is assumed to be fixed. Then, we calibrate the parameters of Weibull distribution using $\bar{t}$ and σ. We use all six boarding methods previously discussed for this test. We fix $R_L = 10\%$ and $R_J = 30\%$. For every boarding method and average bag stowing time, we perform 40 simulations and find the average boarding time for each of them. The following table shows the average boarding time for each boarding method and average bag stowing time. Figure 10 demonstrates the result graphically.

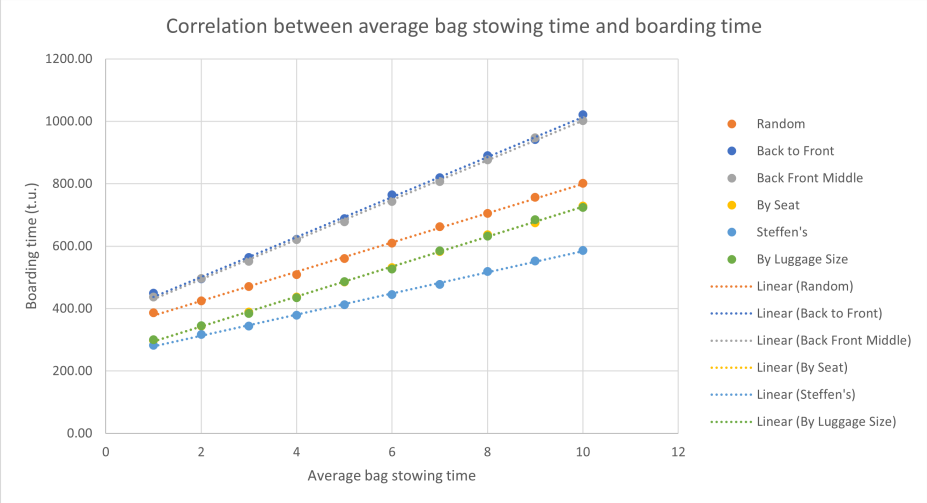

Figure 10: Graph between average bag stowing time and boarding time, along with the best-fit linear lines.

Boarding method/$\bar{t}$ (t.u.)	1	2	3	4	5	6	7	8	9	10
Random boarding	387	425	471	508	560	609	662	705	756	802
Back-to-front boarding	449	496	564	622	689	765	819	890	942	1021
Back-front-middle boarding	437	497	552	621	678	743	807	877	947	1003
Boarding by seat	298	344	389	437	485	531	583	637	675	729
Steffen's method	283	318	344	379	412	445	478	519	552	586
Boarding by luggage size	300	345	385	435	487	527	585	632	685	724

From Figure 10, we can see that the ranking of boarding methods does not change when average bag stowing time changes. This shows that our model is very stable to the change in passengers' luggage size.

4.3.2 Variation of Percentage of Queue-jumping Passengers

We test the sensitivity to queue-jumping passengers by varying the queue-jumping ratio from 0-85 %, with each step equals 5%. The ratio between late passengers and all passengers is set to be constant at 10%. For each boarding method and queue-jumping ratio, we perform 40 simulations and find the average value. The result is graphically shown in Figure 11.

Boarding method/ $\%R_J$	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85
Random boarding	562	567	565	568	564	560	562	574	565	563	568	563	563	570	562	565	567	568
Back-to-front boarding	687	685	693	686	688	685	688	690	689	683	691	689	681	682	681	685	682	679
Back-front-middle boarding	690	672	682	678	679	684	689	676	681	675	678	679	681	682	676	678	678	676
Boarding by seat	485	488	485	485	485	488	490	491	486	490	483	482	486	483	487	493	488	486
Steffen's method	344	363	374	384	394	405	410	424	426	437	440	444	453	458	459	463	465	468
Boarding by luggage size	486	483	486	488	488	483	483	485	488	486	491	486	484	485	484	489	483	484

The result shows that Steffen's Method has the highest sensitivity, while other methods are completely stable. Since the number of priority M is apparently larger than the rest, the chance that the passenger queue-jumps and change the priority is higher which significantly affects the boarding time. This reaffirms that our boarding model works as we intended.

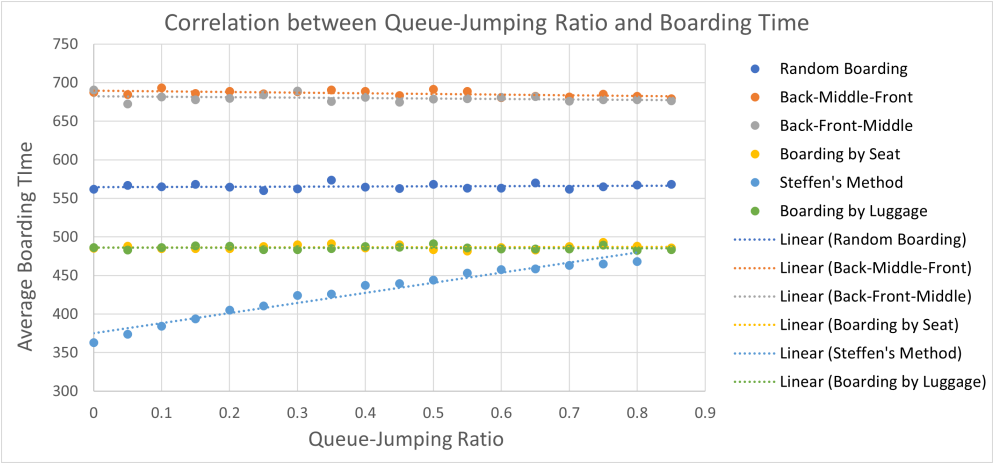

Figure 11: Graph between Queue-Jumping Ratio R_J and average boarding time .

4.4 Disembarking Methods

The following disembarking methods will be tested:

1. Random: Every passenger has priority 1. The only condition is imposed not by priority, but by the fact that passengers closer to the aisle must move before passengers further from the aisle.

2. Disembarking by Seat: The passengers in aisle seats, middle seats, and window seats have priority 1, 2, and 3, respectively. Disembarking by Seat is a method commonly used by airlines [20].

3. Reverse-Pyramid Disembarking: This is inspired by a boarding method in [19], which we adapt to be a disembarking method. Passengers are split into five priorities as shown in Figure 12.

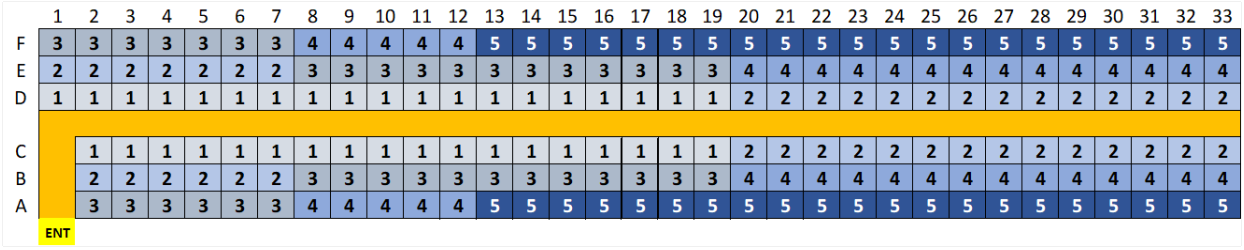

Figure 12: Reverse-pyramid disembarking method on Narrow Body Passenger Aircraft

4.5 Optimal Disembarking Method

We use $R_L = 0.1$, each disembarking method is simulated 80 times. The optimal disembarking method is random embarking, which is probably because passengers do not have to wait to get out of their seat.

Disembarking Method	Average DT	%	S.D.	Practical min.	Practical max.
Random Disembarking	316	100	12	298	333
Disembarking by Seat	318	102	8	301	335
Reversed Pyramid Disembarking	396	125	8	384	409

5 Aircraft II: Flying Wing Aircraft

Aircraft II is a Flying Wing Aircraft, with $N_B = 318$. It has four aisles and five groups of seats. Its main characteristics is its enormous width.

5.1 Optimal Boarding Method

Boarding by seat is the most effective in the Narrow-Body aircraft. By applying this method to the Flying Wing aircraft, the passenger whose seat is near the entrance will block the flow. Modified boarding by seat, inspired by Reverse-Pyramid boarding method, let the passengers whose seat is in the inner section go first, as shown in Figure 13(b). Modified-boarding by seat is tested along with random boarding and boarding by seat, as shown Figure 13(a).

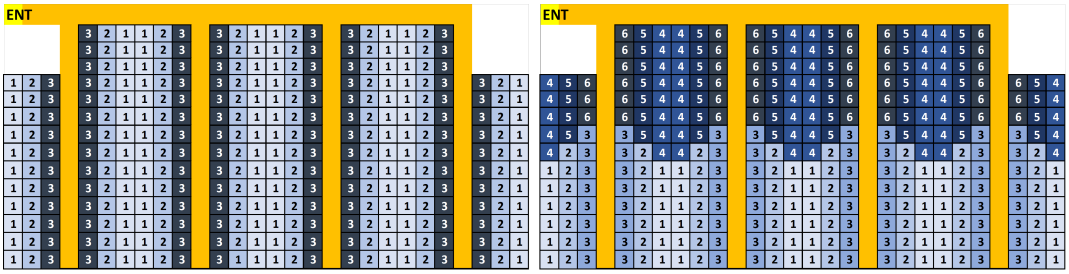

Figure 13: (a) Boarding by Seat ($C = 0.191$) and (b) Modified-Boarding by Seat ($C = 0.311$)

The results after running the simulation 80 times with $R_L = 0.1$ and $R_{J,\max} = 0.5$ is shown in the following table.

Boarding Method	**Average** BT	%	**S.D.**	**Practical min.**	**Practical max.**
Random Boarding	473	100	18.9	445	507
Boarding by seat	438	93	12.3	417	457
Modified-Boarding by seat	433	92	13.5	414	459

The optimal boarding method for this aircraft is the modified boarding by seat method, whose average boarding time (433 t.u. = 615 s) is slightly lower than boarding by seat and random boarding, respectively. Interestingly, the optimal boarding method uses less time than the optimal boarding method in Aircraft I, even though this aircraft has over 50% more passengers. This is because this aircraft is much wider than Aircraft I, allowing more passengers to move at the same time.

5.2 Optimal Disembarking Method

Similar to boarding method, modified-disembarking by seat is tested and compare with random disembarking and original disembarking by seat, as shown in Figure 14.

Disembarking Method	Average DT	S.D.	Practical Maximum	Practical Minimum
Random Disembarking	388	13	368	379
Disembarking by Seat	395	10	410	411

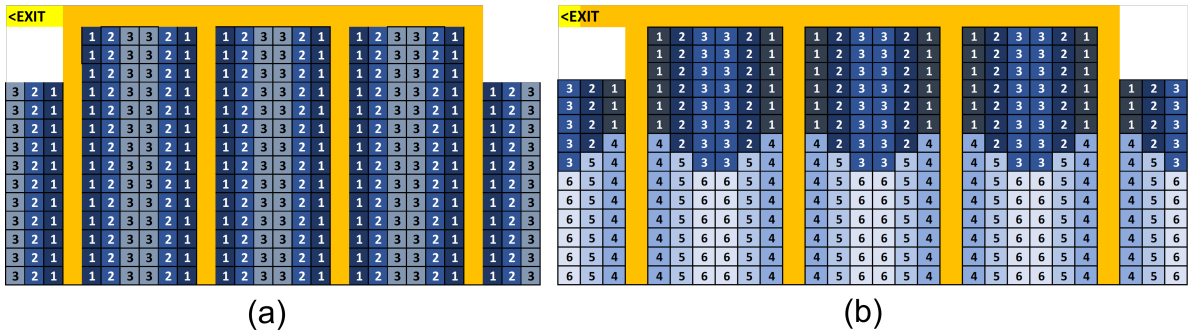

Figure 14: (a) Disembarking by Seat (b) Modified-Disembarking by Seat

6 Aircraft III: Two-entrance Two-aisle Aircraft

The Two-entrance, Two-aisle aircraft has two section, front and back. It has 242 seats ($N_C = 242$). We assume that passengers whose seat is in the front section are limited to enter from Entrance 1 at the front only. This also applies on the back section. Passengers also cannot go through another section during boarding. We also make a further assumption for this case (which obviously can be assumed in any commercial airline).

Assumption: Passengers in business class do not affect boarding time.
Justification: The section of the plane for business class passengers is on the other side of economic class passengers. The only chance they can interfere with the boarding line is when they are moving from the entrance to their section, which is eliminated by the fact that they are boarded before economic class passengers anyway.

6.1 Optimal Boarding Method

Since the seat configuration of Two-entrance two-aisle aircraft is comparable to Narrow-Body, boarding by seat method is expected to be optimal. Four possible methods, as shown in Figure 15, are tested throughout our model. Since there are two entrances, the cabin crew assigned the priority on passengers separately. Therefore, the number of priority is obtained from the sum of priority in both section, front and back. As an example, the boarding method, as shown in Figure 15(b), has complexity $C = \ln\left(\frac{2+4}{228}\right) \approx 0.330$. The result after running the simulation 80 times with $R_L = 0.1$ and $R_{J,\max} = 0.5C$ is shown in the following table.

Boarding Method	**Average** BT	%	**S.D.**	**Practical min.**	**Practical max.**
Random Boarding	207	100	12	189	230
Boarding (a)	221	107	17	194	248
Boarding (b)	196	95	10	178	212
Boarding (c)	237	115	14	215	262
Boarding (d)	196	95	10	180	212

According to the following table shows that boarding method (b) and (d) is one of the best performed methods. However, method (d) is more complicated. Therefore, method (b) is the optimal.

6.2 Optimal Disembarking Method

Similar to boarding methods in Figure 15, however, the priority number is reversed. Every passenger is assumed to exit the same gate as they entered.

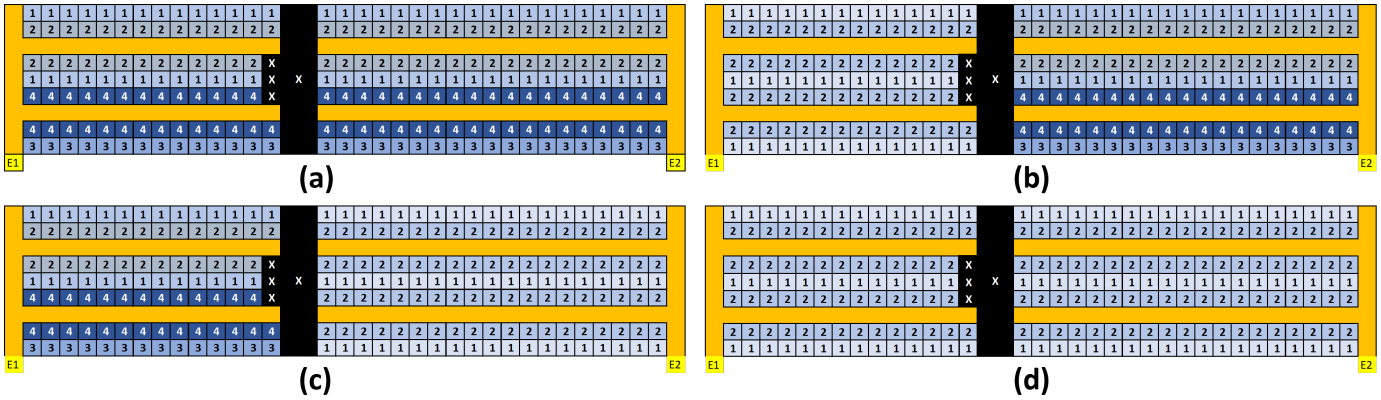

Figure 15: Four possible methods of boarding by seat, $C = 0.255$ for (a), $C = 0.330$ for (b) and (c) and $C = 0.255$ for (d).

7 Adjustments for the Pandemic

Due to the coronavirus pandemic, social distancing measures are employed to slow the spread of the virus. The social distancing measures affect the aviation industry, mainly because airlines can no longer utilise the full capacity of airplanes. Some seats will be unused to allow spaces between passengers [21], but the patterns of used and unused seats can differ between airlines to airlines depending on the guidelines. We will test four patterns of seating of Aircraft I in the pandemic (Figure 16) to see whether the optimal boarding method would still be the same as with the case when the aircraft carries full capacity.

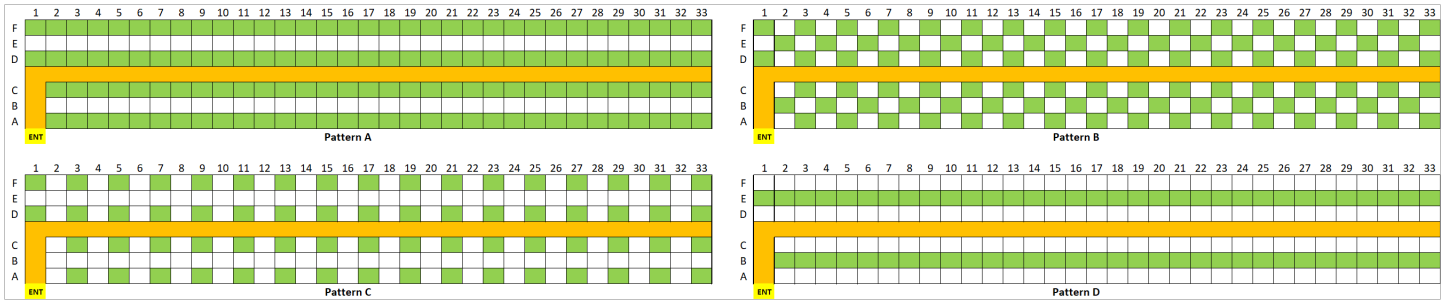

Figure 16: Four patterns of seating in Aircraft I in the pandemic.

We make an additional assumption that

Assumption: The distance between each passenger in the aisle remains the same.
Justification: This is for the ease of implementation.

7.1 Optimal Boarding Methods

We will test four patterns of seating of Aircraft I on layout A,B,C and D with four boarding methods: random, back-to-front, boarding by seat and boarding by luggage size.

Boarding Method	Random	Back to Front	By Seat	By Luggage Size
Layout A (66 %)	359	412	332	328
Layout B (50 %)	271	279	252	249
Layout C (33 %)	202	209	186	186
Layout D (33 %)	182	183	-	179

The optimal boarding method in the pandemic is boarding by luggage size for all seating patterns.

8 Model Evaluation

8.1 Strengths

1. Adaptability: Our models of boarding and disembarking are adaptable to a vast array of aircraft, as shown by our implementation in Aircraft I, II, and III. The cellular automata algorithms are rigorous, making the model general and easily modifiable.

2. Emphasis on Practicality: We quantify complexity of boarding methods via complexity factor. The complexity factor is used to find the rate of queue jumping. Our models are designed such that complex methods are "punished" with passengers' inability to follow, as is in real life.

3. Low Time Complexity: Our simulation algorithms are time-effective. If we look at the code, we can see that the time complexity of our simulations are $O(N \cdot BT)$ or $O(N \cdot DT)$. If we assume that the boarding time BT and disembarking time DT can be approximated as a linear function of N (which they likely are, according from data from [6] and [22]), then our algorithm has time complexity $O(N^2)$. This is considerably faster than other model of simulation, such as agent-based modeling.

8.2 Limitations and Improvements

1. Passenger Group/Family: Our model does not accommodate the fact that some passengers travel in groups/families. These passengers are unlikely to part with other passengers in the same group/family. By including these factors, our result can be a better representation of real boarding process.

2. Complexity factor: Even though the complexity has its physical meaning and able to analyse the method. However, expressing the complexity factor could be done multiple ways. Different expression of complexity might yield the different results.

9 Conclusions

We have investigated the optimal boarding and disembarking method for aircraft. We constructed models for boarding and disembarking processes to test various boarding and disembarking methods. To measure the boarding and disembarking time, our models use cellular automata algorithm to simulate the boarding and disembarking process.

Steffen's method and random disembarking method are found to be, respectively, optimal boarding method and optimal disembarking method for Narrow-body Aircraft. For Flying Wing Aircraft and Two-entrance Two-aisle Aircraft, the optimal boarding methods are, respectively, modified boarding by seat and boarding by seat method. In the pandemic, the optimal boarding method is boarding by luggage size.

10 Letter to an Airline Executive

Dear Sir/Madam,

We would like to thank you for your trust in our team. We have now completed our report about the optimal boarding and disembarking method for your airline, which will certainly allow your airline to generate more revenue due to lesser loss in turnaround time for each flight.

Firstly we design a model that simulate the passengers' behavior in the boarding and disembarking process. We realise that the real-world passengers would not always follow the ideal queue predetermined by a boarding method, so we account for late and queue-jumping passengers in our model. Then, we develop a simulation that considers passengers' moving and bag stowing time, which are the only two factors that significantly affect the boarding time. For disembarking model, we consider late-embarking passengers in our model to better reflect the reality.

We implement our models on three aircraft, each of them representing a category of planes your airline may own.

1. Narrow-body Aircraft: This is an example of a simple, multicolumn plane with one aisle. The optimal boarding method for this type of plane is Steffen's method, which is a efficient yet complicated method to implement. Our recommendation is to use boarding by seat method, i.e. boarding from window seats, middle seats, and aisle seats, which is a little more time-consuming than Steffen's method, but is way easier to practise. The best disembarking method is random disembarking method.

2. Flying Wing Aircraft: This aircraft represents wide aircraft, short but has many aisles. We found that the best boarding method is a modified boarding by seat method. This is like boarding by seat method, but passengers at the back boards before passengers at the front.

3. Two-entrance Two-aisle Aircraft: This is a typical wide-body plane. It has two entrances that allow passengers to board more quickly. The optimal boarding method is a boarding by seat method, boarding non-aisle passengers first then aisle passengers. This method is indeed on par with a similar method, but we recommend this method because it is simpler.

The methods we already discussed work very well when the plane is full or almost full. However, during the pandemic when restrictions are imposed on seating, the optimal boarding method for all cases is boarding by luggage size method.

We hope that you are satisfied with our works. Please do not hesitate to contact us if you have any inquiry. We look forward to more cooperation with your airline in the future. In the meantime, we are eager to see some of our methods being used the next time we are abroad with your airline!

Yours sincerely,
Team IMMC 2022031

References

[1] Bachmat, E., Berend, D., Sapir, L. and Skiena, S., 2007. Optimal boarding policies for thin passengers. *Advances in Applied Probability*, 39(4), pp.1098-1114.

[2] Bachmat, E., Khachaturov, V. and Kuperman, R., 2013. Optimal back-to-front airplane boarding. *Physical Review E*, 87(6), p.062805.

[3] Van Landeghem, H. and Beuselinck, A., 2002. Reducing passenger boarding time in airplanes: A simulation based approach. *European Journal of Operational Research*, 142(2), pp.294-308.

[4] Steffen, J.H., 2008. Optimal boarding method for airline passengers. *Journal of Air Transport Management*, 14(3), pp.146-150.

[5] Jafer, S. and Mi, W., 2017. Comparative study of aircraft boarding strategies using cellular discrete event simulation. *Aerospace*, 4(4), p.57.

[6] Baek, Y., Ha, M. and Jeong, H., 2013. Impact of sequential disorder on the scaling behavior of airplane boarding time. *Physical Review E*, 87(5), p.052803.

[7] Cimler, R. and Olševičová, K., 2013. Analysis Simulation of aircraft disembarking methods. *Global Journal on Technology*, 3.

[8] Steffen, J.H. and Hotchkiss, J., 2012. Experimental test of airplane boarding methods. *Journal of Air Transport Management*, 18(1), pp.64-67.

[9] Qiang, S., Jia, B. and Huang, Q., 2017. Evaluation of airplane boarding/deboarding strategies: A surrogate experimental test. *Symmetry*, 9(10), p.222.

[10] Milne, R.J. and Kelly, A.R., 2014. A new method for boarding passengers onto an airplane. *Journal of Air Transport Management*, 34, pp.93-100.

[11] Delcea, C., Cotfas, L.A., Chiriță, N. and Nica, I., 2018. A two-door airplane boarding approach when using apron buses. *Sustainability*, 10(10), p.3619.

[12] Delcea, C., Cotfas, L.A. and Paun, R., 2018. Agent-based evaluation of the airplane boarding strategies' efficiency and sustainability. *Sustainability*, 10(6), p.1879.

[13] Coppens, J., Dangal, S., Vendel, M., Anjani, S., Akkerman, S., Hiemstra-van Mastrigt, S. and Vink, P., 2018. Improving airplane boarding time: a review, a field study and an experiment with a new way of hand luggage stowing. *International Journal of Aviation, Aeronautics, and Aerospace*, 5(2), p.7.

[14] Jaehn, F. and Neumann, S., 2015. Airplane boarding. *European Journal of Operational Research*, 244(2), pp.339-359.

[15] Ahmadpour, N., Lindgaard, G., Robert, J.M. and Pownall, B., 2014. The thematic structure of passenger comfort experience and its relationship to the context features in the aircraft cabin. *Ergonomics*, 57(6), pp.801-815.

[16] Gwynne, S.M.V., Yapa, U.S., Codrington, L., Thomas, J.R., Jennings, S., Thompson, A.J.L. and Grewal, A., 2018. Small-scale trials on passenger microbehaviours during aircraft boarding and deplaning procedures. *Journal of air transport management*, 67, pp.115-133.

[17] Luo, L., Hong, S., Shang, S., Zhou, X., Yang, J. and Pan, Y., 2021. Intelligent Boarding Modelling and Evaluation: A Simulation-Based Approach. *Journal of Advanced Transportation.*

[18] Mohammadi, K. and Mostafaeipour, A., 2013. Using different methods for comprehensive study of wind turbine utilization in Zarrineh, Iran. *Energy conversion and management*, 65, pp.463-470.

[19] Van Den Briel, M.H., Villalobos, J.R., Hogg, G.L., Lindemann, T. and Mulé, A.V., 2005. America west airlines develops efficient boarding strategies. *Interfaces*, 35(3), pp.191-201.

[20] Schmidt, M., 2017. A review of aircraft turnaround operations and simulations. *Progress in Aerospace Sciences*, 92, pp.25-38.

[21] Salari, M., Milne, R.J., Delcea, C., Kattan, L. and Cotfas, L.A., 2020. Social distancing in airplane seat assignments. *Journal of Air Transport Management*, 89, p.101915.

[22] Schultz, M., 2017, June. Aircraft boarding-data, validation, analysis. In *Proceedings of the 12th USA/Europe Air Traffic Management Research and Development Seminar, Seattle, WA, USA* (pp. 26-30).

A Raw Data

A.1 Raw Data for Subsection 4.2

Aircraft I Method 1 $R_L = 0.1$ $R_J = 0.5C$		Aircraft I Method 2 $R_L = 0.1$ $R_J = 0.5C$		Aircraft I Method 3 $R_L = 0.1$ $R_J = 0.5C$		Aircraft I Method 4 $R_L = 0.1$ $R_J = 0.5C$		Aircraft I Method 5 $R_L = 0.1$ $R_J = 0.5C$		Aircraft I Method 6 $R_L = 0.1$ $R_J = 0.5C$	
566	559	675	652	686	697	471	478	439	469	481	466
576	543	675	677	679	701	478	468	481	428	479	499
554	551	673	691	692	683	472	488	452	424	501	478
548	533	678	691	746	653	485	487	456	429	466	447
537	535	702	703	681	692	511	474	426	434	481	454
553	557	706	663	681	642	500	501	467	431	469	501
547	551	733	657	682	683	495	491	413	438	465	496
533	544	737	710	705	612	531	482	416	464	489	484
561	535	717	695	641	689	479	473	444	439	462	522
553	503	692	686	721	686	507	490	431	447	490	474
562	559	652	693	709	672	509	500	423	448	504	491
561	537	712	705	704	666	479	473	475	464	482	477
529	562	677	690	678	687	510	482	442	415	484	513
611	601	726	687	662	676	504	473	445	440	499	479
582	555	683	719	677	666	478	494	438	441	479	481
546	603	661	625	624	671	454	466	423	444	534	504
564	544	731	718	662	636	483	504	437	432	465	458
571	551	713	683	672	728	455	503	466	453	475	496
583	514	698	668	678	675	489	477	450	454	459	501
589	555	714	694	670	656	506	454	450	457	505	488
524	582	676	678	660	673	501	473	449	431	474	479
522	555	662	649	668	718	481	446	431	436	466	492
578	596	669	702	663	649	501	473	446	451	481	480
544	579	702	730	672	649	469	480	447	435	482	498
563	562	727	677	647	648	477	490	458	432	497	492
544	567	685	693	666	681	459	484	442	443	499	495
551	647	687	663	715	697	476	467	441	425	488	486
546	534	672	692	699	678	506	497	436	438	504	475
609	562	663	680	698	643	466	464	467	436	502	463
536	536	738	701	670	687	468	485	427	460	464	495
593	552	712	642	689	715	521	506	461	476	485	491
560	553	692	684	671	682	469	527	439	435	484	477
558	571	721	672	711	684	458	466	443	452	499	486
574	605	732	661	670	671	501	474	423	411	483	497
551	515	714	686	665	674	470	466	443	448	490	466
564	556	717	670	703	679	465	485	471	488	520	447
570	553	664	669	657	684	463	486	447	419	504	476
611	582	703	695	657	686	483	490	468	443	518	479
562	581	675	680	654	684	473	471	452	440	506	480
579	571	705	719	671	669	500	499	436	426	484	470

A.2 Raw Data for Subsection 5.1

Aircraft II Method 1 $R_L = 0.1$ $R_J = 0.5C$		Aircraft II Method 2 $R_L = 0.1$ $R_J = 0.5C$		Aircraft II Method 3 $R_L = 0.1$ $R_J = 0.5C$	
456	453	450	445	449	435
476	475	440	438	435	432
451	496	440	419	428	421
504	452	411	430	416	410
482	474	428	416	434	432
475	452	454	432	426	459
502	481	436	469	461	453
485	476	417	448	435	432
448	454	457	443	415	437
454	507	448	442	438	461
480	465	440	431	444	438
471	447	440	422	435	418
480	469	426	433	441	454
495	490	433	453	408	423
435	453	428	448	411	442
488	454	442	421	442	445
446	492	436	441	414	449
481	469	459	449	423	419
469	476	434	431	439	431
498	459	438	453	455	433
438	515	455	445	440	432
441	477	450	426	418	444
515	464	443	429	434	421
468	469	448	454	431	416
459	484	431	451	437	420
479	477	438	450	455	428
445	480	432	427	432	444
465	468	436	412	431	427
500	490	437	445	415	425
480	479	432	430	416	439
463	466	418	430	422	414
460	487	463	447	431	456
460	464	449	419	443	438
475	485	426	420	444	421
499	485	426	452	439	450
501	457	418	428	471	420
451	481	448	435	419	433
477	463	445	438	430	424
517	450	447	441	441	419
455	499	442	435	442	423

A.3 Raw Data for Subsection 6.1

3		3		3		3		3	
1 (RANDOM)		2 (a)		3 (b)		4 (c)		5 (d)	
0.1		0.1		0.1		0.1		0.1	
0.5*C		0.5*C		0.5*C		0.5*C		0.5*C	
200	194	199	226	176	196	214	225	204	185
189	195	217	216	212	198	248	208	181	187
208	191	202	270	208	197	239	218	180	189
231	200	223	243	196	188	248	245	197	189
201	210	205	235	181	197	262	253	187	189
217	195	207	227	213	197	234	257	176	195
191	221	206	226	196	201	224	219	207	218
202	190	194	244	192	194	244	238	200	200
200	214	203	227	202	197	229	251	202	187
206	195	208	227	191	205	238	247	208	196
205	219	209	241	197	197	225	244	188	199
202	203	222	246	203	201	250	253	204	204
202	217	210	215	195	197	255	243	201	193
221	209	221	238	194	207	237	216	178	201
215	221	227	229	212	197	263	251	183	198
191	225	206	246	198	198	241	256	189	207
228	224	212	210	193	195	219	238	194	203
193	235	211	239	181	211	252	216	183	191
208	204	196	212	184	196	228	230	192	194
188	226	213	252	206	200	246	241	200	188
221	201	208	248	189	197	262	221	221	223
193	206	212	239	199	189	206	243	197	190
212	213	189	248	201	179	231	237	192	187
220	200	200	229	207	204	252	225	180	212
203	181	203	218	200	211	237	244	192	203
230	197	199	217	197	183	257	219	206	193
199	195	214	233	196	196	234	238	190	192
209	213	214	228	183	192	228	231	189	201
223	203	215	236	193	192	238	238	201	206
235	193	199	246	201	210	234	238	212	197
210	194	196	217	198	209	234	221	185	194
203	206	220	227	210	189	247	231	212	206
212	210	219	237	190	189	251	250	193	199
208	214	204	230	194	191	222	233	194	189
211	211	202	239	209	189	215	251	197	206
220	202	183	238	204	207	217	221	203	203
188	214	187	246	191	162	221	254	200	206
216	219	212	245	182	212	234	262	205	191
211	198	229	228	199	179	216	245	183	193
197	202	225	242	178	178	246	249	201	179

A.4 Raw Data for Subsection 7.1

1D		1D		1D	
1		2		6	
0.1		0.1		0.1	
0.5*C		0.5*C		0.5*C	
185	171	207	166	190	171
170	177	195	182	209	175
185	200	183	183	187	168
174	186	173	189	182	190
189	169	165	168	168	179
188	180	186	204	181	173
181	174	173	192	169	182
192	184	187	178	174	174
176	180	193	173	177	181
187	167	182	201	183	176
192	169	191	201	171	176
168	198	200	171	182	179
226	175	165	178	177	169
190	174	191	193	173	174
192	189	200	178	194	180
194	187	160	183	174	173
169	188	180	204	182	191
191	176	177	184	175	173
192	184	185	202	186	200
173	172	167	185	188	171
188	186	194	171	174	204
184	184	189	193	166	163
197	184	187	204	190	184
174	162	184	184	194	186
168	181	176	177	180	167
180	192	199	163	166	188
181	176	212	183	160	169
204	177	199	183	175	173
186	193	181	177	172	176
174	177	179	208	181	180
182	208	191	183	176	174
187	178	187	174	179	169
174	165	194	172	174	168
158	172	209	175	171	188
171	190	211	168	187	178
184	197	180	180	171	191
177	163	204	191	191	176
185	179	207	175	181	194
177	190	178	175	177	190
181	191	180	182	186	169
192	176	172	172	188	189

B Codes

B.1 Codes of Boarding Process, Airplane I

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import statistics as st
import random
from statistics import stdev
from scipy.integrate import quad

def run1(case,RL,RJ,N):
    # Create position of people in "Narrow Body" Passenger Aircraft
    arr = np.arange(1,196)
    list_0 = arr.tolist()
    for i in range (99):
        list_0[i] = [math.floor(list_0[i]/33)+1,list_0[i]%33]
    for i in range (99,195):
        list_0[i] = [math.floor((list_0[i]-99)/32)+5,(list_0[i]-99)%32+1]
    list_0[32] = [1,33]
    list_0[65] = [2,33]
    list_0[98] = [3,33]
    list_0[130] = [5,33]
    list_0[162] = [6,33]
    list_0[194] = [7,33]

    # Random luggage stow time of each people by experimental data and Weilbell distribution
    luggage = [6.2, 6.5, 6.6, 9.0, 7.7, 7.1, 5.3, 5.0, 5.5, 6.5, 6.2, 6.5, 4.9,
    5.1, 8.3, 8.4, 7.6, 8.9, 9.4, 7.6, 6.5, 7.6, 8.0, 6.4, 5.5, 6.4, 6.6, 6.9,
    9.2, 10.6, 8.1, 6.4, 7.7, 9.0, 8.9, 9.7, 8.1, 7.8, 8.2, 9.0, 7.6, 7.6, 5.1,
    3.9, 9.8, 7.7, 8.0, 6.6, 6.3, 6.5, 7.0, 9.6, 7.3, 7.2, 6.7, 8.6, 7.4, 7.6,
    7.2, 8.3, 8.3, 9.2, 8.8, 8.8, 7.0, 6.9, 5.7, 7.7, 6.5, 6.3, 8.8, 9.4, 7.1,
    6.4, 6.4, 5.3, 6.0, 5.7, 4.4, 4.0, 5.0, 1.9, 5.1, 8.2, 5.3, 6.7, 6.7, 10.7]
    luggage_arr = np.array(luggage)
    mean = luggage_arr.mean()
    std = stdev(luggage_arr)
    k = (std/mean)**(-1.086)
    z = 1 + 1/k
    def f(x):
        return math.exp(-x)*(x**(z-1))
    gamma,err = quad(f, 0, math.inf)
    c = mean/gamma
    for i in range (len(list_0)):
        weibull = (c*(np.random.weibull(k, 1))).tolist()
        time = round((weibull[0]/1.42))
        list_0[i].append(time)

    #Case1
    list_1 = random.sample(list_0,195)

    class agent_1:
        def __init__(self,char,seat,bag):
            self.char = char
            self.seat = seat
            self.bag = bag

```

```
    passenger_1 = []
    for i in range(len(list_0)):
        passenger_1.append(agent_1(list_1[i][0],list_1[i][1],list_1[i][2]))

    #Case2
    list_late_2 = random.sample(list_0,round(N*RL))
    list_no_late_2 = [x for x in list_0 if x not in list_late_2]

    list_bow = []
    list_middle_2 = []
    list_aft = []

    for i in range (len(list_no_late_2)):
        if 1 <= list_no_late_2[i][1] <= 11:
            list_bow.append(list_no_late_2[i])

    for i in range (len(list_no_late_2)):
        if 12 <= list_no_late_2[i][1] <= 22:
            list_middle_2.append(list_no_late_2[i])

    for i in range (len(list_no_late_2)):
        if 23 <= list_no_late_2[i][1] <= 33:
            list_aft.append(list_no_late_2[i])

    list_random_bow = random.sample(list_bow,len(list_bow))
    list_random_middle_2 = random.sample(list_middle_2,len(list_middle_2))
    list_random_aft = random.sample(list_aft,len(list_aft))
    list_random_2 = list_random_aft + list_random_middle_2 + list_random_bow

    list_bad_2 = random.sample(list_random_2,round(N*RJ))

    random_2 = []
    for i in range (len(list_random_2)):
        for j in range (len(list_bad_2)):
            if list_random_2[i] == list_bad_2[j]:
                random_2.append(i)

    list_bad_2 = []
    for i in range (len(random_2)):
        list_bad_2.append(list_random_2[random_2[i]])

    list_no_bad_2 = [x for x in list_random_2 if x not in list_bad_2]

    no_random_2 = []
    for i in range (len(list_random_2)):
        for j in range (len(list_no_bad_2)):
            if list_random_2[i] == list_no_bad_2[j]:
                no_random_2.append(i)

    normal_2 = []
    for i in range (len(random_2)):
        normal_2.append(np.random.normal(random_2[i], 5))

    for i in range (len(normal_2)):
        list_bad_2[i].append(normal_2[i])

    list_bad_2 = sorted(list_bad_2, key=lambda x: x[-1])

    for i in range (len(normal_2)):
        del list_bad_2[i][3]
```

```
    normal_2 = sorted(normal_2)

    list_shift_2 = []
    j=0
    for i in range (len(list_no_bad_2)+len(list_bad_2)):
        if (len(list_bad_2) != 0) & (j < len(no_random_2)):
            if normal_2[0] < no_random_2[j]:
                list_shift_2.append(list_bad_2[0])
                del list_bad_2[0]
                del normal_2[0]
            elif normal_2[0] > no_random_2[j]:
                list_shift_2.append(list_no_bad_2[j])
                j=j+1
        elif j >= len(no_random_2):
            list_shift_2.append(list_bad_2[0])
            del list_bad_2[0]
            del normal_2[0]
        else:
            list_shift_2.append(list_no_bad_2[j])
            j=j+1

    list_2 = list_shift_2 + list_late_2

    class agent_2:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_2 = []
    for i in range(len(list_0)):
        passenger_2.append(agent_2(list_2[i][0],list_2[i][1],list_2[i][2],0,0,0))

    #Case3
    list_late_3 = random.sample(list_0,round(N*RL))
    list_no_late_3 = [x for x in list_0 if x not in list_late_3]

    list_bow = []
    list_middle_3 = []
    list_aft = []

    for i in range (len(list_no_late_3)):
        if 1 <= list_no_late_3[i][1] <= 11:
            list_bow.append(list_no_late_3[i])

    for i in range (len(list_no_late_3)):
        if 12 <= list_no_late_3[i][1] <= 22:
            list_middle_3.append(list_no_late_3[i])

    for i in range (len(list_no_late_3)):
        if 23 <= list_no_late_3[i][1] <= 33:
            list_aft.append(list_no_late_3[i])

    list_random_bow = random.sample(list_bow,len(list_bow))
    list_random_middle_3 = random.sample(list_middle_3,len(list_middle_3))
    list_random_aft = random.sample(list_aft,len(list_aft))
```

```
    list_random_3 = list_random_aft + list_random_bow + list_random_middle_3

    list_bad_3 = random.sample(list_random_3,round(N*RJ))

    random_3 = []
    for i in range (len(list_random_3)):
        for j in range (len(list_bad_3)):
            if list_random_3[i] == list_bad_3[j]:
                random_3.append(i)

    list_bad_3 = []
    for i in range (len(random_3)):
        list_bad_3.append(list_random_3[random_3[i]])

    list_no_bad_3 = [x for x in list_random_3 if x not in list_bad_3]

    no_random_3 = []
    for i in range (len(list_random_3)):
        for j in range (len(list_no_bad_3)):
            if list_random_3[i] == list_no_bad_3[j]:
                no_random_3.append(i)

    normal_3 = []
    for i in range (len(random_3)):
        normal_3.append(np.random.normal(random_3[i], 5))

    for i in range (len(normal_3)):
        list_bad_3[i].append(normal_3[i])

    list_bad_3 = sorted(list_bad_3, key=lambda x: x[-1])

    for i in range (len(normal_3)):
        del list_bad_3[i][3]

    normal_3 = sorted(normal_3)

    list_shift_3 = []
    j=0
    for i in range (len(list_no_bad_3)+len(list_bad_3)):
        if (len(list_bad_3) != 0) & (j < len(no_random_3)):
            if normal_3[0] < no_random_3[j]:
                list_shift_3.append(list_bad_3[0])
                del list_bad_3[0]
                del normal_3[0]
            elif normal_3[0] > no_random_3[j]:
                list_shift_3.append(list_no_bad_3[j])
                j=j+1
        elif j >= len(no_random_3):
            list_shift_3.append(list_bad_3[0])
            del list_bad_3[0]
            del normal_3[0]
        else:
            list_shift_3.append(list_no_bad_3[j])
            j=j+1

    list_3 = list_shift_3 + list_late_3

    class agent_3:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
```

```
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_3 = []
    for i in range(len(list_0)):
        passenger_3.append(agent_3(list_3[i][0],list_3[i][1],list_3[i][2],0,0,0))

    #Case4
    list_late_4 = random.sample(list_0, round(N*RL))
    list_no_late_4 = [x for x in list_0 if x not in list_late_4]

    list_window_4 = []
    list_middle_4 = []
    list_aisle_4 = []

    for i in range (len(list_no_late_4)):
        if (list_no_late_4[i][0] == 1) | (list_no_late_4[i][0] == 7):
            list_window_4.append(list_no_late_4[i])
    for i in range (len(list_no_late_4)):
        if (list_no_late_4[i][0] == 2) | (list_no_late_4[i][0] == 6):
            list_middle_4.append(list_no_late_4[i])

    for i in range (len(list_no_late_4)):
        if (list_no_late_4[i][0] == 3) | (list_no_late_4[i][0] ==5):
            list_aisle_4.append(list_no_late_4[i])

    list_random_window_4 = random.sample(list_window_4,len(list_window_4))
    list_random_middle_4 = random.sample(list_middle_4,len(list_middle_4))
    list_random_aisle_4 = random.sample(list_aisle_4,len(list_aisle_4))
    list_random_4 = list_random_window_4 + list_random_middle_4 +
   list_random_aisle_4

    list_bad_4 = random.sample(list_random_4,round(N*RJ))

    random_4 = []
    for i in range (len(list_random_4)):
        for j in range (len(list_bad_4)):
            if list_random_4[i] == list_bad_4[j]:
                random_4.append(i)

    list_bad_4 = []
    for i in range (len(random_4)):
        list_bad_4.append(list_random_4[random_4[i]])

    list_no_bad_4 = [x for x in list_random_4 if x not in list_bad_4]

    no_random_4 = []
    for i in range (len(list_random_4)):
        for j in range (len(list_no_bad_4)):
            if list_random_4[i] == list_no_bad_4[j]:
                no_random_4.append(i)

    normal_4 = []
    for i in range (len(random_4)):
        normal_4.append(np.random.normal(random_4[i], 5))

    for i in range (len(normal_4)):
```

```
        list_bad_4[i].append(normal_4[i])

    list_bad_4 = sorted(list_bad_4, key=lambda x: x[-1])

    for i in range (len(normal_4)):
        del list_bad_4[i][3]

    normal_4 = sorted(normal_4)

    list_shift_4 = []
    j=0
    for i in range (len(list_no_bad_4)+len(list_bad_4)):
        if (len(list_bad_4) != 0) & (j < len(no_random_4)):
            if normal_4[0] < no_random_4[j]:
                list_shift_4.append(list_bad_4[0])
                del list_bad_4[0]
                del normal_4[0]
            elif normal_4[0] > no_random_4[j]:
                list_shift_4.append(list_no_bad_4[j])
                j=j+1
        elif j >= len(no_random_4):
            list_shift_4.append(list_bad_4[0])
            del list_bad_4[0]
            del normal_4[0]
        else:
            list_shift_4.append(list_no_bad_4[j])
            j=j+1

    list_4 = list_shift_4 + list_late_4

    class agent_4:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_4 = []
    for i in range(len(list_0)):
        passenger_4.append(agent_4(list_4[i][0],list_4[i][1],list_4[i][2],0,0,0))

    #Case5
    arr_1_ABC = np.arange(33,0,-2)
    arr_1_DEF = np.arange(33,1,-2)
    arr_2_all = np.arange(32,0,-2)
    list_1_ABC = arr_1_ABC.tolist()
    list_1_DEF = arr_1_DEF.tolist()
    list_2_all = arr_2_all.tolist()

    list_correct_5 = []

    for i in range (len(list_1_ABC)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 1) & (list_0[j][1] == list_1_ABC[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_1_DEF)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 7) & (list_0[j][1] == list_1_DEF[i]):
```

```
                list_correct_5.append(list_0[j])
    for i in range (len(list_2_all)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 1) & (list_0[j][1] == list_2_all[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_2_all)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 7) & (list_0[j][1] == list_2_all[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_1_ABC)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 2) & (list_0[j][1] == list_1_ABC[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_1_DEF)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 6) & (list_0[j][1] == list_1_DEF[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_2_all)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 2) & (list_0[j][1] == list_2_all[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_2_all)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 6) & (list_0[j][1] == list_2_all[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_1_ABC)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 3) & (list_0[j][1] == list_1_ABC[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_1_DEF)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 5) & (list_0[j][1] == list_1_DEF[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_2_all)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 3) & (list_0[j][1] == list_2_all[i]):
                list_correct_5.append(list_0[j])
    for i in range (len(list_2_all)):
        for j in range (len(list_0)):
            if (list_0[j][0] == 5) & (list_0[j][1] == list_2_all[i]):
                list_correct_5.append(list_0[j])

    list_late_5 = random.sample(list_correct_5,round(N*RL))
    list_no_late_5 = [x for x in list_correct_5 if x not in list_late_5]

    list_bad_5 = random.sample(list_no_late_5,round(N*RJ))

    random_5 = []
    for i in range (len(list_no_late_5)):
        for j in range (len(list_bad_5)):
            if list_no_late_5[i] == list_bad_5[j]:
                random_5.append(i)

    list_bad_5 = []
    for i in range (len(random_5)):
        list_bad_5.append(list_no_late_5[random_5[i]])

    list_no_bad_5 = [x for x in list_no_late_5 if x not in list_bad_5]

    no_random_5 = []
```

```
    for i in range (len(list_no_late_5)):
        for j in range (len(list_no_bad_5)):
            if list_no_late_5[i] == list_no_bad_5[j]:
                no_random_5.append(i)

    normal_5 = []
    for i in range (len(random_5)):
        normal_5.append(np.random.normal(random_5[i], 5))

    for i in range (len(normal_5)):
        list_bad_5[i].append(normal_5[i])

    list_bad_5 = sorted(list_bad_5, key=lambda x: x[-1])

    for i in range (len(normal_5)):
        del list_bad_5[i][3]

    normal_5 = sorted(normal_5)

    list_shift_5 = []
    j=0
    for i in range (len(list_no_bad_5)+len(list_bad_5)):
        if (len(list_bad_5) != 0) & (j < len(no_random_5)):
            if normal_5[0] < no_random_5[j]:
                list_shift_5.append(list_bad_5[0])
                del list_bad_5[0]
                del normal_5[0]
            elif normal_5[0] > no_random_5[j]:
                list_shift_5.append(list_no_bad_5[j])
                j=j+1
        elif j >= len(no_random_5):
            list_shift_5.append(list_bad_5[0])
            del list_bad_5[0]
            del normal_5[0]
        else:
            list_shift_5.append(list_no_bad_5[j])
            j=j+1

    list_5 = list_shift_5 + list_late_5

    class agent_5:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_5 = []
    for i in range(len(list_0)):
        passenger_5.append(agent_5(list_5[i][0],list_5[i][1],list_5[i][2],0,0,0))

    #Case6
    list_late_6 = random.sample(list_0,round(N*RL))
    list_no_late_6 = [x for x in list_0 if x not in list_late_6]

    list_window_6 = []
    list_middle_6 = []
    list_aisle_6 = []
```

```
    list_window_6_0 = []
    list_window_6_4 = []
    list_window_6_7 = []
    list_middle_6_0 = []
    list_middle_6_4 = []
    list_middle_6_7 = []
    list_aisle_6_0 = []
    list_aisle_6_4 = []
    list_aisle_6_7 = []

    for i in range (len(list_no_late_6)):
        if (list_no_late_6[i][0] == 1) | (list_no_late_6[i][0] == 7):
            list_window_6.append(list_no_late_6[i])

    for i in range (len(list_no_late_6)):
        if (list_no_late_6[i][0] == 2) | (list_no_late_6[i][0] == 6):
            list_middle_6.append(list_no_late_6[i])

    for i in range (len(list_no_late_6)):
        if (list_no_late_6[i][0] == 3) | (list_no_late_6[i][0] == 5):
            list_aisle_6.append(list_no_late_6[i])

    for i in range (len(list_window_6)):
        if 0 <= list_window_6[i][2] <= 3:
            list_window_6_0.append(list_window_6[i])

    for i in range (len(list_window_6)):
        if 4 <= list_window_6[i][2] <= 6:
            list_window_6_4.append(list_window_6[i])

    for i in range (len(list_window_6)):
        if 7 <= list_window_6[i][2]:
            list_window_6_7.append(list_window_6[i])

    for i in range (len(list_middle_6)):
        if 0 <= list_middle_6[i][2] <=3:
            list_middle_6_0.append(list_middle_6[i])

    for i in range (len(list_middle_6)):
        if 4 <= list_middle_6[i][2] <=6:
            list_middle_6_4.append(list_middle_6[i])

    for i in range (len(list_middle_6)):
        if 7 <= list_middle_6[i][2]:
            list_middle_6_7.append(list_middle_6[i])

    for i in range (len(list_aisle_6)):
        if 0 <= list_aisle_6[i][2] <=3:
            list_aisle_6_0.append(list_aisle_6[i])

    for i in range (len(list_aisle_6)):
        if 4 <= list_aisle_6[i][2] <=6:
            list_aisle_6_4.append(list_aisle_6[i])

    for i in range (len(list_aisle_6)):
        if 7 <= list_aisle_6[i][2]:
            list_aisle_6_7.append(list_aisle_6[i])

    list_random_window_6_0 = random.sample(list_window_6_0,len(list_window_6_0))
    list_random_window_6_4 = random.sample(list_window_6_4,len(list_window_6_4))
```

```
    list_random_window_6_7 = random.sample(list_window_6_7,len(list_window_6_7))
    list_random_middle_6_0 = random.sample(list_middle_6_0,len(list_middle_6_0))
    list_random_middle_6_4 = random.sample(list_middle_6_4,len(list_middle_6_4))
    list_random_middle_6_7 = random.sample(list_middle_6_7,len(list_middle_6_7))
    list_random_aisle_6_0 = random.sample(list_aisle_6_0,len(list_aisle_6_0))
    list_random_aisle_6_4 = random.sample(list_aisle_6_4,len(list_aisle_6_4))
    list_random_aisle_6_7 = random.sample(list_aisle_6_7,len(list_aisle_6_7))
    list_random_6 = list_random_window_6_7 + list_random_window_6_4 +
   list_random_window_6_0 + list_random_middle_6_7 + list_random_middle_6_4 +
   list_random_middle_6_0 + list_random_aisle_6_7 + list_random_aisle_6_4 +
   list_random_aisle_6_0

    list_bad_6 = random.sample(list_random_6,round(N*RJ))

    random_6 = []
    for i in range (len(list_random_6)):
        for j in range (len(list_bad_6)):
            if list_random_6[i] == list_bad_6[j]:
                random_6.append(i)

    list_bad_6 = []
    for i in range (len(random_6)):
        list_bad_6.append(list_random_6[random_6[i]])

    list_no_bad_6 = [x for x in list_random_6 if x not in list_bad_6]

    no_random_6 = []
    for i in range (len(list_random_6)):
        for j in range (len(list_no_bad_6)):
            if list_random_6[i] == list_no_bad_6[j]:
                no_random_6.append(i)

    normal_6 = []
    for i in range (len(random_6)):
        normal_6.append(np.random.normal(random_6[i], 5))

    for i in range (len(normal_6)):
        list_bad_6[i].append(normal_6[i])

    list_bad_6 = sorted(list_bad_6, key=lambda x: x[-1])

    for i in range (len(normal_6)):
        del list_bad_6[i][3]

    normal_6 = sorted(normal_6)

    list_shift_6 = []
    j=0
    for i in range (len(list_no_bad_6)+len(list_bad_6)):
        if (len(list_bad_6) != 0) & (j < len(no_random_6)):
            if normal_6[0] < no_random_6[j]:
                list_shift_6.append(list_bad_6[0])
                del list_bad_6[0]
                del normal_6[0]
            elif normal_6[0] > no_random_6[j]:
                list_shift_6.append(list_no_bad_6[j])
                j=j+1
        elif j >= len(no_random_6):
            list_shift_6.append(list_bad_6[0])
            del list_bad_6[0]
```

```
            del normal_6[0]
        else:
            list_shift_6.append(list_no_bad_6[j])
            j=j+1

    list_6 = list_shift_6 + list_late_6

    class agent_6:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_6 = []
    for i in range(len(list_0)):
        passenger_6.append(agent_6(list_6[i][0],list_6[i][1],list_6[i][2],0,0,0))

    class person:
        def __init__(self,char,seat,bag,num_out,t_1,t_2,check):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.t_1 = t_1
            self.t_2 = t_2
            self.check = check

    list_pass = [[person(0,0,0,0,0,0,0) for i in range(0,34)] for j in range(0,8)
    ]

    class grid:
        def __init__(self,type,value,pass_char,pass_seat):
            self.type = type
            # 0 -> block
            # 1 -> queue
            # 2 -> aisle
            # 3 -> seat
            self.value = value
            # 0 -> avaliable
            # 1 -> passenger
            self.pass_char = pass_char
            self.pass_seat = pass_seat

    plane = [[grid(0,0,0,0) for i in range(0,36)] for i in range(0,200)]

    for i in range(1,8):
        for j in range(1,34):
            plane[i][j].type = 3

    for i in range(5,8):
        plane[i][1].type = 0

    for i in range(0,36):
        plane[4][i].type = 2

    for i in range(5,200):
        plane[i][0].type = 1
```

```
645
646     def C1(passenger_1):
647         for i in range(len(passenger_1)):
648             list_pass[passenger_1[i].char][passenger_1[i].seat].char =
    passenger_1[i].char
649             list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
    passenger_1[i].seat
650             list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
    [i].bag
651             list_pass[passenger_1[i].char][passenger_1[i].seat].num_out = -1
652             list_pass[passenger_1[i].char][passenger_1[i].seat].t_1 = -1
653             list_pass[passenger_1[i].char][passenger_1[i].seat].t_2 = -1
654             list_pass[passenger_1[i].char][passenger_1[i].seat].check = 0
655         for i in range(0,len(passenger_1)):
656             plane[5+i][0].value = 1
657             plane[5+i][0].pass_char = passenger_1[i].char
658             plane[5+i][0].pass_seat = passenger_1[i].seat
659
660     def C2(passenger_2):
661         for i in range(len(passenger_2)):
662             list_pass[passenger_2[i].char][passenger_2[i].seat].char =
    passenger_2[i].char
663             list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
    passenger_2[i].seat
664             list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
    [i].bag
665             list_pass[passenger_2[i].char][passenger_2[i].seat].num_out = -1
666             list_pass[passenger_2[i].char][passenger_2[i].seat].t_1 = -1
667             list_pass[passenger_2[i].char][passenger_2[i].seat].t_2 = -1
668             list_pass[passenger_2[i].char][passenger_2[i].seat].check = 0
669         for i in range(0,len(passenger_2)):
670             plane[5+i][0].value = 1
671             plane[5+i][0].pass_char = passenger_2[i].char
672             plane[5+i][0].pass_seat = passenger_2[i].seat
673
674     def C3(passenger_3):
675         for i in range(len(passenger_3)):
676             list_pass[passenger_3[i].char][passenger_3[i].seat].char =
    passenger_3[i].char
677             list_pass[passenger_3[i].char][passenger_3[i].seat].seat =
    passenger_3[i].seat
678             list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3
    [i].bag
679             list_pass[passenger_3[i].char][passenger_3[i].seat].num_out = -1
680             list_pass[passenger_3[i].char][passenger_3[i].seat].t_1 = -1
681             list_pass[passenger_3[i].char][passenger_3[i].seat].t_2 = -1
682             list_pass[passenger_3[i].char][passenger_3[i].seat].check = 0
683         for i in range(0,len(passenger_3)):
684             plane[5+i][0].value = 1
685             plane[5+i][0].pass_char = passenger_3[i].char
686             plane[5+i][0].pass_seat = passenger_3[i].seat
687
688     def C4(passenger_4):
689         for i in range(len(passenger_4)):
690             list_pass[passenger_4[i].char][passenger_4[i].seat].char =
    passenger_4[i].char
691             list_pass[passenger_4[i].char][passenger_4[i].seat].seat =
    passenger_4[i].seat
692             list_pass[passenger_4[i].char][passenger_4[i].seat].bag = passenger_4
    [i].bag
```

```
            list_pass[passenger_4[i].char][passenger_4[i].seat].num_out = -1
            list_pass[passenger_4[i].char][passenger_4[i].seat].t_1 = -1
            list_pass[passenger_4[i].char][passenger_4[i].seat].t_2 = -1
            list_pass[passenger_4[i].char][passenger_4[i].seat].check = 0
        for i in range(0,len(passenger_4)):
            plane[5+i][0].value = 1
            plane[5+i][0].pass_char = passenger_4[i].char
            plane[5+i][0].pass_seat = passenger_4[i].seat

    def C5(passenger_5):
        for i in range(len(passenger_5)):
            list_pass[passenger_5[i].char][passenger_5[i].seat].char =
passenger_5[i].char
            list_pass[passenger_5[i].char][passenger_5[i].seat].seat =
passenger_5[i].seat
            list_pass[passenger_5[i].char][passenger_5[i].seat].bag = passenger_5
[i].bag
            list_pass[passenger_5[i].char][passenger_5[i].seat].num_out = -1
            list_pass[passenger_5[i].char][passenger_5[i].seat].t_1 = -1
            list_pass[passenger_5[i].char][passenger_5[i].seat].t_2 = -1
            list_pass[passenger_5[i].char][passenger_5[i].seat].check = 0
        for i in range(0,len(passenger_5)):
            plane[5+i][0].value = 1
            plane[5+i][0].pass_char = passenger_5[i].char
            plane[5+i][0].pass_seat = passenger_5[i].seat

    def C6(passenger_6):
        for i in range(len(passenger_6)):
            list_pass[passenger_6[i].char][passenger_6[i].seat].char =
passenger_6[i].char
            list_pass[passenger_6[i].char][passenger_6[i].seat].seat =
passenger_6[i].seat
            list_pass[passenger_6[i].char][passenger_6[i].seat].bag = passenger_6
[i].bag
            list_pass[passenger_6[i].char][passenger_6[i].seat].num_out = -1
            list_pass[passenger_6[i].char][passenger_6[i].seat].t_1 = -1
            list_pass[passenger_6[i].char][passenger_6[i].seat].t_2 = -1
            list_pass[passenger_6[i].char][passenger_6[i].seat].check = 0
        for i in range(0,len(passenger_6)):
            plane[5+i][0].value = 1
            plane[5+i][0].pass_char = passenger_6[i].char
            plane[5+i][0].pass_seat = passenger_6[i].seat

    if(case==1):
        C1(passenger_1)
    if(case==2):
        C2(passenger_2)
    if(case==3):
        C3(passenger_3)
    if(case==4):
        C4(passenger_4)
    if(case==5):
        C5(passenger_5)
    if(case==6):
        C6(passenger_6)

    a = [[0 for i in range(0,36)] for i in range(0,200)]

    for i in range(0,200):
            for j in range(0,36):
```

```
                #print(plane[i][j].value)
                a[i][j] = plane[i][j].type

    time = 0

    def check_pass(plane):
        check = 0
        for w in range(0,200):
            for z in range(0,36):
                if(plane[w][z].type == 3 and plane[w][z].value == 1):
                    check+=1
        return check

    time = 0
    while(1):
        #check
        check = check_pass(plane)
        if(check==N):
            #print(time)
            break
        time+=1

        i=2
        for j in range(1,34):
            if(plane[i][j].pass_char == 1 and plane[i][j].value == 1 and plane[i
-1][j].value == 0):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                plane[i-1][j].pass_char = p1r
                plane[i-1][j].pass_seat = p1c
                plane[i-1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].value = 0

        i=6
        for j in range(2,34):
            if(plane[i][j].pass_char==7 and plane[i][j].value == 1 and plane[i
+1][j].value == 0):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                plane[i+1][j].pass_char = p1r
                plane[i+1][j].pass_seat = p1c
                plane[i+1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].value = 0

        i=3
        for j in range(1,34):
            if((plane[i][j].pass_char==1 or plane[i][j].pass_char==2) and plane[i
][j].value == 1 and plane[i-1][j].value == 0):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                plane[i-1][j].pass_char = p1r
                plane[i-1][j].pass_seat = p1c
                plane[i-1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].value = 0
```

```
804
805         i=5
806         for j in range(2,34):
807             if((plane[i][j].pass_char==6 or plane[i][j].pass_char==7) and plane[i
    ][j].value == 1 and plane[i+1][j].value == 0):
808                 p1r = plane[i][j].pass_char
809                 p1c = plane[i][j].pass_seat
810                 plane[i+1][j].pass_char = p1r
811                 plane[i+1][j].pass_seat = p1c
812                 plane[i+1][j].value = 1
813                 plane[i][j].pass_char = 0
814                 plane[i][j].pass_seat = 0
815                 plane[i][j].value = 0
816
817         i = 4
818         for j in reversed(range(0,36)):
819             if(plane[i][j].value==0):
820                 continue
821             if(plane[i][j].pass_seat==j):
822                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2>0)
    :
823                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2
    -=1
824                     continue
825                 else:
826                     p1r = plane[i][j].pass_char
827                     p1c = plane[i][j].pass_seat
828                     if(plane[i][j].pass_char<4):
829                         if(plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].num_out-1][j].value == 1):
830                             p2r = plane[i-list_pass[plane[i][j].pass_char][plane[
    i][j].pass_seat].num_out-1][j].pass_char
831                             p2c = plane[i-list_pass[plane[i][j].pass_char][plane[
    i][j].pass_seat].num_out-1][j].pass_seat
832                             plane[i-1][j].pass_char = p2r
833                             plane[i-1][j].pass_seat = p2c
834                             plane[i-1][j].value = 1
835                             plane[i-2][j].pass_char = p1r
836                             plane[i-2][j].pass_seat = p1c
837                             plane[i-2][j].value = 1
838                             plane[i][j].pass_char = 0
839                             plane[i][j].pass_seat = 0
840                             plane[i][j].value = 0
841                             continue
842                         plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].num_out-1][j].pass_char = p1r
843                         plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].num_out-1][j].pass_seat = p1c
844                         plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].num_out-1][j].value = 1
845                     else:
846                         if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].num_out+1][j].value == 1):
847                             p2r = plane[i+list_pass[plane[i][j].pass_char][plane[
    i][j].pass_seat].num_out+1][j].pass_char
848                             p2c = plane[i+list_pass[plane[i][j].pass_char][plane[
    i][j].pass_seat].num_out+1][j].pass_seat
849                             plane[i+1][j].pass_char = p2r
850                             plane[i+1][j].pass_seat = p2c
851                             plane[i+1][j].value = 1
```

```
                            plane[i+2][j].pass_char = p1r
                            plane[i+2][j].pass_seat = p1c
                            plane[i+2][j].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0
                            continue
                        if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value == 1):
                            continue
                        plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_char = p1r
                        plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_seat = p1c
                        plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value = 1
                    plane[i][j].pass_char = 0
                    plane[i][j].pass_seat = 0
                    plane[i][j].value = 0
            if(plane[i][j].pass_seat>j):
                if(plane[i][j].pass_seat-j==1):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
                        continue
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
check==0):
                        sum = 0
                        if(plane[i][j].pass_char<4):
                            for m in range(1,4):
                                if(plane[m][j+1].value == 1 and plane[m][j+1].
pass_char!=m):
                                    sum+=1
                        else:
                            for m in range(5,8):
                                if(plane[m][j+1].value == 1 and plane[m][j+1].
pass_char!=m):
                                    sum+=1
                        if(sum!=0):
                            continue
                        sum = 0
                        if(plane[i][j].pass_char<4):
                            for m in reversed(range(plane[i][j].pass_char+1,4)):
                                if(plane[m][j+1].value == 1):
                                    sum+=1
                        else:
                            for m in range(5,plane[i][j].pass_char):
                                if(plane[m][j+1].value == 1):
                                    sum+=1
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
num_out = sum
                        mov = 0
                        for n in range(0,list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].num_out+1):
                            if(plane[i][j+1+n].value==1):
                                mov = 1
                        if(mov == 1):
                            continue
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
```

```
    check = 1
                        if(plane[i][j].pass_char<4):
                            if(plane[i][j].pass_char==3):
                                list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 0
                                list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 0
                            if(plane[i][j].pass_char==2):
                                if(plane[i-1][j+1].value==1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 1
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 2
                                else:
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 0
                            if(plane[i][j].pass_char==1):
                                if(plane[i-1][j+1].value==1 and plane[i-2][j+1].
    value==1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 2
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 3
                                if(plane[i-1][j+1].value==1 and plane[i-2][j+1].
    value==0):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 1
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 2
                                if(plane[i-1][j+1].value==0 and plane[i-2][j+1].
    value==1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 2
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 2
                                if(plane[i-1][j+1].value==0 and plane[i-2][j+1].
    value==0):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 0
                        else:
                            if(plane[i][j].pass_char==5):
                                list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 0
                                list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 0
                            if(plane[i][j].pass_char==6):
                                if(plane[i+1][j+1].value==1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 1
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 2
                                else:
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 0
```

```
                            if(plane[i][j].pass_char==7):
                                if(plane[i+1][j+1].value==1 and plane[i+2][j+1].
    value==1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 2
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 3
                                if(plane[i+1][j+1].value==1 and plane[i+2][j+1].
    value==0):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 1
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 2
                                if(plane[i+1][j+1].value==0 and plane[i+2][j+1].
    value==1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 2
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 2
                                if(plane[i+1][j+1].value==0 and plane[i+2][j+1].
    value==0):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_2 = 0
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
    check == 1):
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
    ].t_1>0):
                            list_pass[plane[i][j].pass_char][plane[i][j].
    pass_seat].t_1-=1
                        else:
                            if(plane[i][j+1].value==1):
                                continue
                            p1r = plane[i][j].pass_char
                            p1c = plane[i][j].pass_seat
                            plane[i][j+1].pass_char = p1r
                            plane[i][j+1].pass_seat = p1c
                            plane[i][j+1].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0
                else:
                    if(plane[i][j+1].value==0):
                        p1r = plane[i][j].pass_char
                        p1c = plane[i][j].pass_seat
                        plane[i][j+1].pass_char = p1r
                        plane[i][j+1].pass_seat = p1c
                        plane[i][j+1].value = 1
                        plane[i][j].pass_char = 0
                        plane[i][j].pass_seat = 0
                        plane[i][j].value = 0

        j = 0
        for i in range(5,200):
            if(plane[i-1][j].value==0 and plane[i][j].value==1):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                plane[i-1][j].pass_char = p1r
                plane[i-1][j].pass_seat = p1c
```

```
                plane[i-1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].value = 0

        #print("time",time)
        #print("check",check)
        """i = 4
        for j in reversed(range(0,36)):
            if(plane[i][j].value==1):
                print("i = ",i,"j = ",j,"goal = ",plane[i][j].pass_char,plane[i][
    j].pass_seat,list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out,
    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1,list_pass[plane[i
    ][j].pass_char][plane[i][j].pass_seat].t_2,list_pass[plane[i][j].pass_char][
    plane[i][j].pass_seat].check)
        print("\n")"""
        """for i in range(0,200):
            for j in range(0,36):
                #print(plane[i][j].value)
                a[i][j] = plane[i][j].value
        plt.figure('time'+str(time))
        im = plt.imshow(a[0:8])
        ax = plt.gca()
        ax.set_xticks(np.arange(-.5, 36, 1), minor=True)
        ax.set_yticks(np.arange(-.5, 8, 1), minor=True)
        ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
        plt.savefig('time'+str(time)+'.png')
        #plt.show()"""
    return time

print(run1(1,0,0.3,195))
```

B.2 Codes of Boarding Process, Airplane II

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import statistics as st
import random
from statistics import stdev
from scipy.integrate import quad

def run2(case,RL,RJ,N):
    # Create position of people in "Narrow Body" Passenger Aircraft
    arr = np.arange(1,319)
    list_0 = arr.tolist()
    for i in range (33):
        list_0[i] = [math.floor(list_0[i]/11)+1, (list_0[i]%11)+3]
    for i in range (33,117):
        list_0[i] = [math.floor((list_0[i]-33)/14)+5, ((list_0[i]-33)%14)]
    for i in range (117,201):
        list_0[i] = [math.floor((list_0[i]-117)/14)+12, ((list_0[i]-117)%14)]
    for i in range (201,285):
        list_0[i] = [math.floor((list_0[i]-201)/14)+19, ((list_0[i]-201)%14)]
    for i in range (285,318):
        list_0[i] = [math.floor((list_0[i]-285)/11)+26, ((list_0[i]-285)%11)+3]
    for i in range (len(list_0)):
        if list_0[i][1] == 13:
            list_0[i+1] = [list_0[i][0], 14]
```

```
27
28      # Random luggage stow time of each people by experimental data and Weilbell
    distribution
29      luggage = [6.2, 6.5, 6.6, 9.0, 7.7, 7.1, 5.3, 5.0, 5.5, 6.5, 6.2, 6.5, 4.9,
    5.1, 8.3, 8.4, 7.6, 8.9, 9.4, 7.6, 6.5, 7.6, 8.0, 6.4, 5.5, 6.4, 6.6, 6.9,
    9.2, 10.6, 8.1, 6.4, 7.7, 9.0, 8.9, 9.7, 8.1, 7.8, 8.2, 9.0, 7.6, 7.6, 5.1,
    3.9, 9.8, 7.7, 8.0, 6.6, 6.3, 6.5, 7.0, 9.6, 7.3, 7.2, 6.7, 8.6, 7.4, 7.6,
    7.2, 8.3, 8.3, 9.2, 8.8, 8.8, 7.0, 6.9, 5.7, 7.7, 6.5, 6.3, 8.8, 9.4, 7.1,
    6.4, 6.4, 5.3, 6.0, 5.7, 4.4, 4.0, 5.0, 1.9, 5.1, 8.2, 5.3, 6.7, 6.7, 10.7]
30      luggage_arr = np.array(luggage)
31      mean = luggage_arr.mean()
32      std = stdev(luggage_arr)
33      k = (std/mean)**(-1.086)
34      z = 1 + 1/k
35      def f(x):
36          return math.exp(-x)*(x**(z-1))
37      gamma,err = quad(f, 0, math.inf)
38      c = mean/gamma
39      for i in range (len(list_0)):
40          weibull = (c*(np.random.weibull(k, 1))).tolist()
41          time = round((weibull[0]/1.42))
42          list_0[i].append(time)
43
44      #Case1
45      list_1 = random.sample(list_0,len(list_0))
46
47      class agent_1:
48          def __init__(self,char,seat,bag):
49              self.char = char
50              self.seat = seat
51              self.bag = bag
52
53      passenger_1 = []
54      for i in range(len(list_0)):
55          passenger_1.append(agent_1(list_1[i][0],list_1[i][1],list_1[i][2]))
56
57      #Case2
58      list_in_2 = []
59      list_mid_2 = []
60      list_out_2 = []
61
62      for i in range (len(list_0)):
63          if (list_0[i][0] == 1) or (list_0[i][0] == 7) or (list_0[i][0] == 8) or (
    list_0[i][0] == 14) or (list_0[i][0] == 15) or (list_0[i][0] == 21) or (list_0
    [i][0] == 22) or (list_0[i][0] == 28):
64              list_in_2.append(list_0[i])
65      for i in range (len(list_0)):
66          if (list_0[i][0] == 2) or (list_0[i][0] == 6) or (list_0[i][0] == 9) or (
    list_0[i][0] == 13) or (list_0[i][0] == 16) or (list_0[i][0] == 20) or (list_0
    [i][0] == 23) or (list_0[i][0] == 27):
67              list_mid_2.append(list_0[i])
68      for i in range (len(list_0)):
69          if (list_0[i][0] == 3) or (list_0[i][0] == 5) or (list_0[i][0] == 10) or
    (list_0[i][0] == 12) or (list_0[i][0] == 17) or (list_0[i][0] == 19) or (
    list_0[i][0] == 24) or (list_0[i][0] == 26):
70              list_out_2.append(list_0[i])
71
72      list_random_in_2 = random.sample(list_in_2,len(list_in_2))
73      list_random_mid_2 = random.sample(list_mid_2,len(list_mid_2))
74      list_random_out_2 = random.sample(list_out_2,len(list_out_2))
```

```
    list_correct_2 = list_random_in_2 + list_random_mid_2 + list_random_out_2

    list_late_2 = random.sample(list_correct_2,round(N*RL))
    list_no_late_2 = [x for x in list_correct_2 if x not in list_late_2]

    list_bad_2 = random.sample(list_no_late_2,round(N*RJ))

    random_2 = []
    for i in range (len(list_no_late_2)):
        for j in range (len(list_bad_2)):
            if list_no_late_2[i] == list_bad_2[j]:
                random_2.append(i)

    list_bad_2 = []
    for i in range (len(random_2)):
        list_bad_2.append(list_no_late_2[random_2[i]])

    list_no_bad_2 = [x for x in list_no_late_2 if x not in list_bad_2]

    no_random_2 = []
    for i in range (len(list_no_late_2)):
        for j in range (len(list_no_bad_2)):
            if list_no_late_2[i] == list_no_bad_2[j]:
                no_random_2.append(i)

    normal_2 = []
    for i in range (len(random_2)):
        normal_2.append(np.random.normal(random_2[i], 5))

    for i in range (len(normal_2)):
        list_bad_2[i].append(normal_2[i])

    list_bad_2 = sorted(list_bad_2, key=lambda x: x[-1])

    for i in range (len(normal_2)):
        del list_bad_2[i][3]

    normal_2 = sorted(normal_2)

    list_shift_2 = []
    j=0
    for i in range (len(list_no_bad_2)+len(list_bad_2)):
        if (len(list_bad_2) != 0) & (j < len(no_random_2)):
            if normal_2[0] < no_random_2[j]:
                list_shift_2.append(list_bad_2[0])
                del list_bad_2[0]
                del normal_2[0]
            elif normal_2[0] > no_random_2[j]:
                list_shift_2.append(list_no_bad_2[j])
                j=j+1
        elif j >= len(no_random_2):
            list_shift_2.append(list_bad_2[0])
            del list_bad_2[0]
            del normal_2[0]
        else:
            list_shift_2.append(list_no_bad_2[j])
            j=j+1

    list_2 = list_shift_2 + list_late_2

```

```
    class agent_2:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_2 = []
    for i in range(len(list_0)):
        passenger_2.append(agent_2(list_2[i][0],list_2[i][1],list_2[i][2],0,0,0))

    #Case3
    list_1_3 = []
    list_2_3 = []
    list_3_3 = []
    list_4_3 = []
    list_5_3 = []
    list_6_3 = []
    list_7_3 = []
    list_8_3 = []
    list_9_3 = []
    list_10_3 = []
    list_11_3 = []
    list_12_3 = []

    for i in range (len(list_0)):
        if (list_0[i][0] == 1) or (list_0[i][0] == 7):
            list_1_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 8) or (list_0[i][0] == 14):
            list_2_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 15) or (list_0[i][0] == 21):
            list_3_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 22) or (list_0[i][0] == 28):
            list_4_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 2) or (list_0[i][0] == 6):
            list_5_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 9) or (list_0[i][0] == 13):
            list_6_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 16) or (list_0[i][0] == 20):
            list_7_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 23) or (list_0[i][0] == 27):
            list_8_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 3) or (list_0[i][0] == 5):
            list_9_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 10) or (list_0[i][0] == 12):
            list_10_3.append(list_0[i])
    for i in range (len(list_0)):
        if (list_0[i][0] == 17) or (list_0[i][0] == 19):
            list_11_3.append(list_0[i])
```

```
    for i in range (len(list_0)):
        if (list_0[i][0] == 24) or (list_0[i][0] == 26):
            list_12_3.append(list_0[i])

    list_random_1_3 = random.sample(list_1_3,len(list_1_3))
    list_random_2_3 = random.sample(list_2_3,len(list_2_3))
    list_random_3_3 = random.sample(list_3_3,len(list_3_3))
    list_random_4_3 = random.sample(list_4_3,len(list_4_3))
    list_random_5_3 = random.sample(list_5_3,len(list_5_3))
    list_random_6_3 = random.sample(list_6_3,len(list_6_3))
    list_random_7_3 = random.sample(list_7_3,len(list_7_3))
    list_random_8_3 = random.sample(list_8_3,len(list_8_3))
    list_random_9_3 = random.sample(list_9_3,len(list_9_3))
    list_random_10_3 = random.sample(list_10_3,len(list_10_3))
    list_random_11_3 = random.sample(list_11_3,len(list_11_3))
    list_random_12_3 = random.sample(list_12_3,len(list_12_3))
    list_correct_3 = list_random_1_3 + list_random_2_3 + list_random_3_3 +
   list_random_4_3 + list_random_5_3 + list_random_6_3 + list_random_7_3 +
   list_random_8_3 + list_random_9_3 + list_random_10_3 + list_random_11_3 +
   list_random_12_3

    list_late_3 = random.sample(list_correct_3,round(N*RL))
    list_no_late_3 = [x for x in list_correct_3 if x not in list_late_3]

    list_bad_3 = random.sample(list_no_late_3,round(N*RJ))

    random_3 = []
    for i in range (len(list_no_late_3)):
        for j in range (len(list_bad_3)):
            if list_no_late_3[i] == list_bad_3[j]:
                random_3.append(i)

    list_bad_3 = []
    for i in range (len(random_3)):
        list_bad_3.append(list_no_late_3[random_3[i]])

    list_no_bad_3 = [x for x in list_no_late_3 if x not in list_bad_3]

    no_random_3 = []
    for i in range (len(list_no_late_3)):
        for j in range (len(list_no_bad_3)):
            if list_no_late_3[i] == list_no_bad_3[j]:
                no_random_3.append(i)

    normal_3 = []
    for i in range (len(random_3)):
        normal_3.append(np.random.normal(random_3[i], 5))

    for i in range (len(normal_3)):
        list_bad_3[i].append(normal_3[i])

    list_bad_3 = sorted(list_bad_3, key=lambda x: x[-1])

    for i in range (len(normal_3)):
        del list_bad_3[i][3]

    normal_3 = sorted(normal_3)

    list_shift_3 = []
    j=0
```

```
    for i in range (len(list_no_bad_3)+len(list_bad_3)):
        if (len(list_bad_3) != 0) & (j < len(no_random_3)):
            if normal_3[0] < no_random_3[j]:
                list_shift_3.append(list_bad_3[0])
                del list_bad_3[0]
                del normal_3[0]
            elif normal_3[0] > no_random_3[j]:
                list_shift_3.append(list_no_bad_3[j])
                j=j+1
        elif j >= len(no_random_3):
            list_shift_3.append(list_bad_3[0])
            del list_bad_3[0]
            del normal_3[0]
        else:
            list_shift_3.append(list_no_bad_3[j])
            j=j+1

    list_3 = list_shift_3 + list_late_3

    class agent_3:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_3 = []
    for i in range(len(list_0)):
        passenger_3.append(agent_3(list_3[i][0],list_3[i][1],list_3[i][2],0,0,0))

    #Case4

    list_1_4 = []
    list_2_4 = []
    list_3_4 = []
    list_4_4 = []
    list_5_4 = []
    list_6_4 = []

    for i in range (len(list_0)):
        if ((list_0[i][0] == 1) and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0]
== 7) and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 8) and (9 <= list_0[
i][1] <= 14)) or ((list_0[i][0] == 14) and (9 <= list_0[i][1] <= 14)) or ((
list_0[i][0] == 15) and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 21)
and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 22) and (9 <= list_0[i][1]
 <= 14)) or ((list_0[i][0] == 28) and (9 <= list_0[i][1] <= 14)):
            list_1_4.append(list_0[i])
    for i in range (len(list_0)):
        if ((list_0[i][0] == 2) and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0]
== 6) and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 9) and (8 <= list_0[
i][1] <= 14)) or ((list_0[i][0] == 13) and (8 <= list_0[i][1] <= 14)) or ((
list_0[i][0] == 16) and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 20)
and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 23) and (8 <= list_0[i][1]
 <= 14)) or ((list_0[i][0] == 27) and (8 <= list_0[i][1] <= 14)):
            list_2_4.append(list_0[i])
    for i in range (len(list_0)):
        if ((list_0[i][0] == 3) and (7 <= list_0[i][1] <= 14)) or ((list_0[i][0]
== 5) and (7 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 10) and (7 <= list_0
```

```
    [i][1] <= 14)) or ((list_0[i][0] == 12) and (7 <= list_0[i][1] <= 14)) or ((
    list_0[i][0] == 17) and (7 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 19)
    and (7 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 24) and (7 <= list_0[i][1]
     <= 14)) or ((list_0[i][0] == 26) and (7 <= list_0[i][1] <= 14)):
301             list_3_4.append(list_0[i])
302     for i in range (len(list_0)):
303         if ((list_0[i][0] == 1) and (4 <= list_0[i][1] <= 8)) or ((list_0[i][0]
    == 7) and (1 <= list_0[i][1] <= 8)) or ((list_0[i][0] == 8) and (1 <= list_0[i
    ][1] <= 8)) or ((list_0[i][0] == 14) and (1 <= list_0[i][1] <= 8)) or ((list_0
    [i][0] == 15) and (1 <= list_0[i][1] <= 8)) or ((list_0[i][0] == 21) and (1 <=
     list_0[i][1] <= 8)) or ((list_0[i][0] == 22) and (1 <= list_0[i][1] <= 8)) or
     ((list_0[i][0] == 28) and (4 <= list_0[i][1] <= 8)):
304             list_4_4.append(list_0[i])
305     for i in range (len(list_0)):
306         if ((list_0[i][0] == 2) and (4 <= list_0[i][1] <= 7)) or ((list_0[i][0]
    == 6) and (1 <= list_0[i][1] <= 7)) or ((list_0[i][0] == 9) and (1 <= list_0[i
    ][1] <= 7)) or ((list_0[i][0] == 13) and (1 <= list_0[i][1] <= 7)) or ((list_0
    [i][0] == 16) and (1 <= list_0[i][1] <= 7)) or ((list_0[i][0] == 20) and (1 <=
     list_0[i][1] <= 7)) or ((list_0[i][0] == 23) and (1 <= list_0[i][1] <= 7)) or
     ((list_0[i][0] == 27) and (4 <= list_0[i][1] <= 7)):
307             list_5_4.append(list_0[i])
308     for i in range (len(list_0)):
309         if ((list_0[i][0] == 3) and (4 <= list_0[i][1] <= 6)) or ((list_0[i][0]
    == 5) and (1 <= list_0[i][1] <= 6)) or ((list_0[i][0] == 10) and (1 <= list_0[
    i][1] <= 6)) or ((list_0[i][0] == 12) and (1 <= list_0[i][1] <= 6)) or ((
    list_0[i][0] == 17) and (1 <= list_0[i][1] <= 6)) or ((list_0[i][0] == 19) and
     (1 <= list_0[i][1] <= 6)) or ((list_0[i][0] == 24) and (1 <= list_0[i][1] <=
    6)) or ((list_0[i][0] == 26) and (4 <= list_0[i][1] <= 6)):
310             list_6_4.append(list_0[i])
311
312     list_random_1_4 = random.sample(list_1_4,len(list_1_4))
313     list_random_2_4 = random.sample(list_2_4,len(list_2_4))
314     list_random_3_4 = random.sample(list_3_4,len(list_3_4))
315     list_random_4_4 = random.sample(list_4_4,len(list_4_4))
316     list_random_5_4 = random.sample(list_5_4,len(list_5_4))
317     list_random_6_4 = random.sample(list_6_4,len(list_6_4))
318     list_correct_4 = list_random_1_4 + list_random_2_4 + list_random_3_4 +
    list_random_4_4 + list_random_5_4 + list_random_6_4
319
320     list_late_4 = random.sample(list_correct_4,round(N*RL))
321     list_no_late_4 = [x for x in list_correct_4 if x not in list_late_4]
322
323     list_bad_4 = random.sample(list_no_late_4,round(N*RJ))
324
325     random_4 = []
326     for i in range (len(list_no_late_4)):
327         for j in range (len(list_bad_4)):
328             if list_no_late_4[i] == list_bad_4[j]:
329                 random_4.append(i)
330
331     list_bad_4 = []
332     for i in range (len(random_4)):
333         list_bad_4.append(list_no_late_4[random_4[i]])
334
335     list_no_bad_4 = [x for x in list_no_late_4 if x not in list_bad_4]
336
337     no_random_4 = []
338     for i in range (len(list_no_late_4)):
339         for j in range (len(list_no_bad_4)):
340             if list_no_late_4[i] == list_no_bad_4[j]:
```

```
                no_random_4.append(i)

    normal_4 = []
    for i in range (len(random_4)):
        normal_4.append(np.random.normal(random_4[i], 5))

    for i in range (len(normal_4)):
        list_bad_4[i].append(normal_4[i])

    list_bad_4 = sorted(list_bad_4, key=lambda x: x[-1])

    for i in range (len(normal_4)):
        del list_bad_4[i][3]

    normal_4 = sorted(normal_4)

    list_shift_4 = []
    j=0;
    for i in range (len(list_no_bad_4)+len(list_bad_4)):
        if (len(list_bad_4) != 0) & (j < len(no_random_4)):
            if normal_4[0] < no_random_4[j]:
                list_shift_4.append(list_bad_4[0])
                del list_bad_4[0]
                del normal_4[0]
            elif normal_4[0] > no_random_4[j]:
                list_shift_4.append(list_no_bad_4[j])
                j=j+1
        elif j >= len(no_random_4):
            list_shift_4.append(list_bad_4[0])
            del list_bad_4[0]
            del normal_4[0]
        else:
            list_shift_4.append(list_no_bad_4[j])
            j=j+1

    list_4 = list_shift_4 + list_late_4

    class agent_4:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_4 = []
    for i in range(len(list_0)):
        passenger_4.append(agent_4(list_4[i][0],list_4[i][1],list_4[i][2],0,0,0))

    class person:
        def __init__(self,char,seat,bag,num_out,t_1,t_2,check):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.t_1 = t_1
            self.t_2 = t_2
            self.check = check
```

```
    list_pass = [[person(0,0,0,0,0,0,0) for i in range(0,15)] for j in range
(0,29)]

    class grid:
        def __init__(self,type,value,pass_char,pass_seat):
            self.type = type
            # 0 -> block
            # 1 -> queue
            # 2 -> aisle
            # 3 -> seat
            self.value = value
            # 0 -> avaliable
            # 1 -> passenger
            self.pass_char = pass_char
            self.pass_seat = pass_seat

    plane = [[grid(0,0,0,0) for i in range(0,17)] for j in range(0,344)]

    for i in range(1,4):
        for j in range(4,15):
            plane[i][j].type = 3

    for i in range(5,11):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(12,18):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(19,25):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(26,29):
        for j in range(4,15):
            plane[i][j].type = 3

    for i in range(4,344):
        plane[i][0].type = 1

    q = [4,11,18,25]
    for i in q:
        for j in range(0,17):
            plane[i][j].type = 2

    def C1(passenger_1):
        for i in range(len(passenger_1)):
            list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
            list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
            list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
            list_pass[passenger_1[i].char][passenger_1[i].seat].num_out = -1
            list_pass[passenger_1[i].char][passenger_1[i].seat].t_1 = -1
            list_pass[passenger_1[i].char][passenger_1[i].seat].t_2 = -1
            list_pass[passenger_1[i].char][passenger_1[i].seat].check = 0
        for i in range(0,len(passenger_1)):
            plane[26+i][0].value = 1
```

```
            plane[26+i][0].pass_char = passenger_1[i].char
            plane[26+i][0].pass_seat = passenger_1[i].seat

    def C2(passenger_2):
        for i in range(len(passenger_2)):
            list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
            list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
passenger_2[i].seat
            list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
[i].bag
            list_pass[passenger_2[i].char][passenger_2[i].seat].num_out = -1
            list_pass[passenger_2[i].char][passenger_2[i].seat].t_1 = -1
            list_pass[passenger_2[i].char][passenger_2[i].seat].t_2 = -1
            list_pass[passenger_2[i].char][passenger_2[i].seat].check = 0
        for i in range(0,len(passenger_2)):
            plane[26+i][0].value = 1
            plane[26+i][0].pass_char = passenger_2[i].char
            plane[26+i][0].pass_seat = passenger_2[i].seat

    def C3(passenger_3):
        for i in range(len(passenger_3)):
            list_pass[passenger_3[i].char][passenger_3[i].seat].char =
passenger_3[i].char
            list_pass[passenger_3[i].char][passenger_3[i].seat].seat =
passenger_3[i].seat
            list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3
[i].bag
            list_pass[passenger_3[i].char][passenger_3[i].seat].num_out = -1
            list_pass[passenger_3[i].char][passenger_3[i].seat].t_1 = -1
            list_pass[passenger_3[i].char][passenger_3[i].seat].t_2 = -1
            list_pass[passenger_3[i].char][passenger_3[i].seat].check = 0
        for i in range(0,len(passenger_3)):
            plane[26+i][0].value = 1
            plane[26+i][0].pass_char = passenger_3[i].char
            plane[26+i][0].pass_seat = passenger_3[i].seat

    def C4(passenger_4):
        for i in range(len(passenger_4)):
            list_pass[passenger_4[i].char][passenger_4[i].seat].char =
passenger_4[i].char
            list_pass[passenger_4[i].char][passenger_4[i].seat].seat =
passenger_4[i].seat
            list_pass[passenger_4[i].char][passenger_4[i].seat].bag = passenger_4
[i].bag
            list_pass[passenger_4[i].char][passenger_4[i].seat].num_out = -1
            list_pass[passenger_4[i].char][passenger_4[i].seat].t_1 = -1
            list_pass[passenger_4[i].char][passenger_4[i].seat].t_2 = -1
            list_pass[passenger_4[i].char][passenger_4[i].seat].check = 0
        for i in range(0,len(passenger_4)):
            plane[26+i][0].value = 1
            plane[26+i][0].pass_char = passenger_4[i].char
            plane[26+i][0].pass_seat = passenger_4[i].seat

    if(case==1):
        C1(passenger_1)
    if(case==2):
        C2(passenger_2)
    if(case==3):
        C3(passenger_3)
```

```
    if(case==4):
        C4(passenger_4)

    a = [[0 for i in range(0,17)] for i in range(0,344)]

    for i in range(0,344):
        for j in range(0,17):
            #print(plane[i][j].value)
            a[i][j] = plane[i][j].type

    time = 0

    """plt.figure('time'+str(time))
    im = plt.imshow(a[0:29])
    ax = plt.gca()
    ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
    ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
    plt.savefig("figure_2.png")
    plt.show()"""

    def check_pass(plane):
        check = 0
        for w in range(0,344):
            for z in range(0,17):
                if(plane[w][z].type==3 and plane[w][z].value == 1):
                    check+=1
        return check

    time = 0
    while(1):
        #check
        check = check_pass(plane)
        if(check==N):
            #print(time)
            break
        time+=1

        left_1 = [2,9,16,23]
        for i in left_1:
            for j in range(1,15):
                if(plane[i][j].type == 3 and plane[i][j].pass_char == i-1 and
plane[i][j].value == 1 and plane[i-1][j].value == 0):
                    p1r = plane[i][j].pass_char
                    p1c = plane[i][j].pass_seat
                    plane[i-1][j].pass_char = p1r
                    plane[i-1][j].pass_seat = p1c
                    plane[i-1][j].value = 1
                    plane[i][j].pass_char = 0
                    plane[i][j].pass_seat = 0
                    plane[i][j].value = 0

        right_1 = [6,13,20,27]
        for i in right_1:
            for j in range(1,15):
                if(plane[i][j].type == 3 and plane[i][j].pass_char == i+1 and
plane[i][j].value == 1 and plane[i+1][j].value == 0):
                    p1r = plane[i][j].pass_char
                    p1c = plane[i][j].pass_seat
                    plane[i+1][j].pass_char = p1r
```

```
                    plane[i+1][j].pass_seat = p1c
                    plane[i+1][j].value = 1
                    plane[i][j].pass_char = 0
                    plane[i][j].pass_seat = 0
                    plane[i][j].value = 0

        left_2 = [3,10,17,24]
        for i in left_2:
            for j in range(1,15):
                if(plane[i][j].type == 3 and (plane[i][j].pass_char == i-1 or
plane[i][j].pass_char == i-2) and plane[i][j].value == 1 and plane[i-1][j].
value == 0):
                    p1r = plane[i][j].pass_char
                    p1c = plane[i][j].pass_seat
                    plane[i-1][j].pass_char = p1r
                    plane[i-1][j].pass_seat = p1c
                    plane[i-1][j].value = 1
                    plane[i][j].pass_char = 0
                    plane[i][j].pass_seat = 0
                    plane[i][j].value = 0

        right_2 = [5,12,19,26]
        for i in right_2:
            for j in range(1,15):
                if(plane[i][j].type == 3 and (plane[i][j].pass_char == i+1 or
plane[i][j].pass_char == i+2) and plane[i][j].value == 1 and plane[i-1][j].
value == 0):
                    p1r = plane[i][j].pass_char
                    p1c = plane[i][j].pass_seat
                    plane[i+1][j].pass_char = p1r
                    plane[i+1][j].pass_seat = p1c
                    plane[i+1][j].value = 1
                    plane[i][j].pass_char = 0
                    plane[i][j].pass_seat = 0
                    plane[i][j].value = 0

        aisle = [4,11,18,25]
        for i in aisle:
            for j in reversed(range(0,17)):
                if(math.ceil(plane[i][j].pass_char/7)*7-3!=i):
                    continue
                if(plane[i][j].value==0):
                    continue
                if(plane[i][j].pass_seat==j):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2-=1
                        continue
                    else:
                        p1r = plane[i][j].pass_char
                        p1c = plane[i][j].pass_seat
                        if(plane[i][j].pass_char<i):
                            if(plane[i-list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out-1][j].value == 1):
                                p2r = plane[i-list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out-1][j].pass_char
                                p2c = plane[i-list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out-1][j].pass_seat
                                plane[i-1][j].pass_char = p2r
```

```
                                plane[i-1][j].pass_seat = p2c
                                plane[i-1][j].value = 1
                                plane[i-2][j].pass_char = p1r
                                plane[i-2][j].pass_seat = p1c
                                plane[i-2][j].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                                continue
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_char = p1r
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_seat = p1c
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].value = 1
                        else:
                            if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value == 1):
                                p2r = plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_char
                                p2c = plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_seat
                                plane[i+1][j].pass_char = p2r
                                plane[i+1][j].pass_seat = p2c
                                plane[i+1][j].value = 1
                                plane[i+2][j].pass_char = p1r
                                plane[i+2][j].pass_seat = p1c
                                plane[i+2][j].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                                continue
                            if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value == 1):
                                continue
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_char = p1r
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_seat = p1c
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value = 1
                        plane[i][j].pass_char = 0
                        plane[i][j].pass_seat = 0
                        plane[i][j].value = 0
                if(plane[i][j].pass_seat>j):
                    if(plane[i][j].pass_seat-j==1):
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag>0):
                            list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag-=1
                            continue
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].check==0):
                            sum = 0
                            if(plane[i][j].pass_char<i):
                                for m in range(i-3,i):
                                    if(plane[m][j+1].value == 1 and plane[m][j+1].pass_char!=m):
                                        sum+=1
                            else:
```

```
                                for m in range(i+1,i+4):
                                    if(plane[m][j+1].value == 1 and plane[m][j
+1].pass_char!=m):
                                        sum+=1
                            if(sum!=0):
                                continue
                            sum = 0
                            if(plane[i][j].pass_char<i):
                                for m in reversed(range(plane[i][j].pass_char+1,i
)):
                                    if(plane[m][j+1].value == 1):
                                        sum+=1
                            else:
                                for m in range(i+1,plane[i][j].pass_char):
                                    if(plane[m][j+1].value == 1):
                                        sum+=1
                            list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out = sum
                            mov = 0
                            for n in range(0,list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1):
                                if(plane[i][j+1+n].value==1):
                                    mov = 1
                            if(mov == 1):
                                continue
                            list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].check = 1
                            if(plane[i][j].pass_char<i):
                                if(plane[i][j].pass_char==i-1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
                                if(plane[i][j].pass_char==i-2):
                                    if(plane[i-1][j+1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                    else:
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                                if(plane[i][j].pass_char==i-3):
                                    if(plane[i-1][j+1].value==1 and plane[i-2][j
+1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 2
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 3
                                    if(plane[i-1][j+1].value==1 and plane[i-2][j
+1].value==0):
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                    if(plane[i-1][j+1].value==0 and plane[i-2][j
+1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i
```

```
][j].pass_seat].t_1 = 2
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                    if(plane[i-1][j+1].value==0 and plane[i-2][j
+1].value==0):
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                                else:
                                    if(plane[i][j].pass_char==i+1):
                                        list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
                                    if(plane[i][j].pass_char==i+2):
                                        if(plane[i+1][j+1].value==1):
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                        else:
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                                    if(plane[i][j].pass_char==i+3):
                                        if(plane[i+1][j+1].value==1 and plane[i+2][j
+1].value==1):
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 2
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 3
                                        if(plane[i+1][j+1].value==1 and plane[i+2][j
+1].value==0):
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                        if(plane[i+1][j+1].value==0 and plane[i+2][j
+1].value==1):
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 2
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                        if(plane[i+1][j+1].value==0 and plane[i+2][j
+1].value==0):
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
                                            list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                                if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].check == 1):
                                    if(list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1>0):
                                        list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1-=1
                                    else:
                                        if(plane[i][j+1].value==1):
                                            continue
```

```
                                    p1r = plane[i][j].pass_char
                                    p1c = plane[i][j].pass_seat
                                    plane[i][j+1].pass_char = p1r
                                    plane[i][j+1].pass_seat = p1c
                                    plane[i][j+1].value = 1
                                    plane[i][j].pass_char = 0
                                    plane[i][j].pass_seat = 0
                                    plane[i][j].value = 0
                        else:
                            if(plane[i][j+1].value==0):
                                p1r = plane[i][j].pass_char
                                p1c = plane[i][j].pass_seat
                                plane[i][j+1].pass_char = p1r
                                plane[i][j+1].pass_seat = p1c
                                plane[i][j+1].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0

        j = 0
        for i in range(5,344):
            if(plane[i-1][j].value==1):
                continue
            else:
                if(plane[i][j].value==1):
                    if(math.ceil(plane[i][j].pass_char/7)*7-3<i):
                        p1r = plane[i][j].pass_char
                        p1c = plane[i][j].pass_seat
                        plane[i-1][j].pass_char = p1r
                        plane[i-1][j].pass_seat = p1c
                        plane[i-1][j].value = 1
                        plane[i][j].pass_char = 0
                        plane[i][j].pass_seat = 0
                        plane[i][j].value = 0

        """print("time",time)
        print("check",check)

        for i in range(0,344):
            for j in range(0,17):
                #print(plane[i][j].value)
                a[i][j] = plane[i][j].value
        plt.figure('time'+str(time))
        im = plt.imshow(a[0:29])
        ax = plt.gca()
        ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
        ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
        ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
        plt.savefig('time'+str(time)+'.png')
        plt.show()"""
    return time
print(run2(3,0,0,318))
```

B.3 Codes of Boarding Process, Airplane III

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import statistics as st
```

```
import random
from statistics import stdev
from scipy.integrate import quad

def run3(case,RL,RJ,N):
    N1 = 95
    N2 = 147
    # Create position of people in "Narrow Body" Passenger Aircraft
    arr = np.arange(1,243)
    list_0 = arr.tolist()
    for i in range (28):
        list_0[i] = [math.floor(list_0[i]/14)+1, (list_0[i]%14)]
    for i in range (28,67):
        list_0[i] = [math.floor((list_0[i]-28)/13)+4, ((list_0[i]-28)%13)]
    for i in range (67,95):
        list_0[i] = [math.floor((list_0[i]-67)/14)+8, ((list_0[i]-67)%14)]
    for i in range (95,137):
        list_0[i] = [math.floor((list_0[i]-95)/21)+1, ((list_0[i]-95)%21)+19]
    for i in range (137,200):
        list_0[i] = [math.floor((list_0[i]-137)/21)+4, ((list_0[i]-137)%21)+19]
    for i in range (200,242):
        list_0[i] = [math.floor((list_0[i]-200)/21)+8, ((list_0[i]-200)%21)+19]
    for i in range (242):
        if (list_0[i][1] == 0) | (list_0[i][1] == 19):
            list_0[i] = [list_0[i-1][0],list_0[i-1][1]+1]

    # Random luggage stow time of each people by experimental data and Weilbell distribution
    luggage = [6.2, 6.5, 6.6, 9.0, 7.7, 7.1, 5.3, 5.0, 5.5, 6.5, 6.2, 6.5, 4.9,
    5.1, 8.3, 8.4, 7.6, 8.9, 9.4, 7.6, 6.5, 7.6, 8.0, 6.4, 5.5, 6.4, 6.6, 6.9,
    9.2, 10.6, 8.1, 6.4, 7.7, 9.0, 8.9, 9.7, 8.1, 7.8, 8.2, 9.0, 7.6, 7.6, 5.1,
    3.9, 9.8, 7.7, 8.0, 6.6, 6.3, 6.5, 7.0, 9.6, 7.3, 7.2, 6.7, 8.6, 7.4, 7.6,
    7.2, 8.3, 8.3, 9.2, 8.8, 8.8, 7.0, 6.9, 5.7, 7.7, 6.5, 6.3, 8.8, 9.4, 7.1,
    6.4, 6.4, 5.3, 6.0, 5.7, 4.4, 4.0, 5.0, 1.9, 5.1, 8.2, 5.3, 6.7, 6.7, 10.7]
    luggage_arr = np.array(luggage)
    mean = luggage_arr.mean()
    std = stdev(luggage_arr)
    k = (std/mean)**(-1.086)
    z = 1 + 1/k
    def f(x):
        return math.exp(-x)*(x**(z-1))
    gamma,err = quad(f, 0, math.inf)
    c = mean/gamma
    for i in range (len(list_0)):
        weibull = (c*(np.random.weibull(k, 1))).tolist()
        time = round((weibull[0]/1.42))
        list_0[i].append(time)

    #Separate 2 entrances
    list_0_e1 = []
    list_0_e2 = []
    for i in range (len(list_0)):
        if list_0[i][1] < 17:
            list_0_e1.append(list_0[i])
    for i in range (len(list_0)):
        if list_0[i][1] > 17:
            list_0_e2.append(list_0[i])

    #Case1
    list_1_e1 = random.sample(list_0_e1,len(list_0_e1))
```

```
    list_1_e2 = random.sample(list_0_e2,len(list_0_e2))

    class agent_1_e1:
        def __init__(self,char,seat,bag):
            self.char = char
            self.seat = seat
            self.bag = bag

    class agent_1_e2:
        def __init__(self,char,seat,bag):
            self.char = char
            self.seat = seat
            self.bag = bag

    passenger_1_e1 = []
    for i in range(len(list_0_e1)):
        passenger_1_e1.append(agent_1_e1(list_1_e1[i][0],list_1_e1[i][1],
    list_1_e1[i][2]))

    passenger_1_e2 = []
    for i in range(len(list_0_e2)):
        passenger_1_e2.append(agent_1_e2(list_1_e2[i][0],list_1_e2[i][1],
    list_1_e2[i][2]))

    #Case2
    list_1_2_e1 = []
    list_2_2_e1 = []
    list_3_2_e1 = []
    list_4_2_e1 = []

    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5):
            list_1_2_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4):
            list_2_2_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 9):
            list_3_2_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 6) | (list_0_e1[i][0] == 8):
            list_4_2_e1.append(list_0_e1[i])

    list_random_1_2_e1 = random.sample(list_1_2_e1,len(list_1_2_e1))
    list_random_2_2_e1 = random.sample(list_2_2_e1,len(list_2_2_e1))
    list_random_3_2_e1 = random.sample(list_3_2_e1,len(list_3_2_e1))
    list_random_4_2_e1 = random.sample(list_4_2_e1,len(list_4_2_e1))
    list_correct_2_e1 = list_random_1_2_e1 + list_random_2_2_e1 +
    list_random_3_2_e1 + list_random_4_2_e1

    list_late_2_e1 = random.sample(list_correct_2_e1,round(N1*RL))
    list_no_late_2_e1 = [x for x in list_correct_2_e1 if x not in list_late_2_e1]

    list_bad_2_e1 = random.sample(list_no_late_2_e1,round(N1*RJ))

    random_2_e1 = []
    for i in range (len(list_no_late_2_e1)):
        for j in range (len(list_bad_2_e1)):
            if list_no_late_2_e1[i] == list_bad_2_e1[j]:
                random_2_e1.append(i)
```

```
117
118     list_bad_2_e1 = []
119     for i in range (len(random_2_e1)):
120         list_bad_2_e1.append(list_no_late_2_e1[random_2_e1[i]])
121
122     list_no_bad_2_e1 = [x for x in list_no_late_2_e1 if x not in list_bad_2_e1]
123
124     no_random_2_e1 = []
125     for i in range (len(list_no_late_2_e1)):
126         for j in range (len(list_no_bad_2_e1)):
127             if list_no_late_2_e1[i] == list_no_bad_2_e1[j]:
128                 no_random_2_e1.append(i)
129
130     normal_2_e1 = []
131     for i in range (len(random_2_e1)):
132         normal_2_e1.append(np.random.normal(random_2_e1[i], 5))
133
134     for i in range (len(normal_2_e1)):
135         list_bad_2_e1[i].append(normal_2_e1[i])
136
137     list_bad_2_e1 = sorted(list_bad_2_e1, key=lambda x: x[-1])
138
139     for i in range (len(normal_2_e1)):
140         del list_bad_2_e1[i][3]
141
142     normal_2_e1 = sorted(normal_2_e1)
143
144     list_shift_2_e1 = []
145     j=0;
146     for i in range (len(list_no_bad_2_e1)+len(list_bad_2_e1)):
147         if (len(list_bad_2_e1) != 0) & (j < len(no_random_2_e1)):
148             if normal_2_e1[0] < no_random_2_e1[j]:
149                 list_shift_2_e1.append(list_bad_2_e1[0])
150                 del list_bad_2_e1[0]
151                 del normal_2_e1[0]
152             elif normal_2_e1[0] > no_random_2_e1[j]:
153                 list_shift_2_e1.append(list_no_bad_2_e1[j])
154                 j=j+1
155         elif j >= len(no_random_2_e1):
156             list_shift_2_e1.append(list_bad_2_e1[0])
157             del list_bad_2_e1[0]
158             del normal_2_e1[0]
159         else:
160             list_shift_2_e1.append(list_no_bad_2_e1[j])
161             j=j+1
162
163     list_2_e1 = list_shift_2_e1 + list_late_2_e1
164
165     class agent_2_e1:
166         def __init__(self,char,seat,bag,num_out,row,col):
167             self.char = char
168             self.seat = seat
169             self.bag = bag
170             self.num_out = num_out
171             self.row = row
172             self.col = col
173
174     passenger_2_e1 = []
175     for i in range(len(list_0_e1)):
176         passenger_2_e1.append(agent_2_e1(list_2_e1[i][0],list_2_e1[i][1],
```

```
    list_2_e1[i][2],0,0,0))

    list_1_2_e2 = []
    list_2_2_e2 = []
    list_3_2_e2 = []
    list_4_2_e2 = []

    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5):
            list_1_2_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4):
            list_2_2_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 9):
            list_3_2_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 6) | (list_0_e2[i][0] == 8):
            list_4_2_e2.append(list_0_e2[i])

    list_random_1_2_e2 = random.sample(list_1_2_e2,len(list_1_2_e2))
    list_random_2_2_e2 = random.sample(list_2_2_e2,len(list_2_2_e2))
    list_random_3_2_e2 = random.sample(list_3_2_e2,len(list_3_2_e2))
    list_random_4_2_e2 = random.sample(list_4_2_e2,len(list_4_2_e2))
    list_correct_2_e2 = list_random_1_2_e2 + list_random_2_2_e2 +
list_random_3_2_e2 + list_random_4_2_e2

    list_late_2_e2 = random.sample(list_correct_2_e2,round(N2*RL))
    list_no_late_2_e2 = [x for x in list_correct_2_e2 if x not in list_late_2_e2]

    list_bad_2_e2 = random.sample(list_no_late_2_e2,round(N2*RJ))

    random_2_e2 = []
    for i in range (len(list_no_late_2_e2)):
        for j in range (len(list_bad_2_e2)):
            if list_no_late_2_e2[i] == list_bad_2_e2[j]:
                random_2_e2.append(i)

    list_bad_2_e2 = []
    for i in range (len(random_2_e2)):
        list_bad_2_e2.append(list_no_late_2_e2[random_2_e2[i]])

    list_no_bad_2_e2 = [x for x in list_no_late_2_e2 if x not in list_bad_2_e2]

    no_random_2_e2 = []
    for i in range (len(list_no_late_2_e2)):
        for j in range (len(list_no_bad_2_e2)):
            if list_no_late_2_e2[i] == list_no_bad_2_e2[j]:
                no_random_2_e2.append(i)

    normal_2_e2 = []
    for i in range (len(random_2_e2)):
        normal_2_e2.append(np.random.normal(random_2_e2[i], 5))

    for i in range (len(normal_2_e2)):
        list_bad_2_e2[i].append(normal_2_e2[i])

    list_bad_2_e2 = sorted(list_bad_2_e2, key=lambda x: x[-1])

    for i in range (len(normal_2_e2)):
```

```
        del list_bad_2_e2[i][3]

    normal_2_e2 = sorted(normal_2_e2)

    list_shift_2_e2 = []
    j=0;
    for i in range (len(list_no_bad_2_e2)+len(list_bad_2_e2)):
        if (len(list_bad_2_e2) != 0) & (j < len(no_random_2_e2)):
            if normal_2_e2[0] < no_random_2_e2[j]:
                list_shift_2_e2.append(list_bad_2_e2[0])
                del list_bad_2_e2[0]
                del normal_2_e2[0]
            elif normal_2_e2[0] > no_random_2_e2[j]:
                list_shift_2_e2.append(list_no_bad_2_e2[j])
                j=j+1
        elif j >= len(no_random_2_e2):
            list_shift_2_e2.append(list_bad_2_e2[0])
            del list_bad_2_e2[0]
            del normal_2_e2[0]
        else:
            list_shift_2_e2.append(list_no_bad_2_e2[j])
            j=j+1

    list_2_e2 = list_shift_2_e2 + list_late_2_e2

    class agent_2_e2:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_2_e2 = []
    for i in range(len(list_0_e2)):
        passenger_2_e2.append(agent_2_e2(list_2_e2[i][0],list_2_e2[i][1],
    list_2_e2[i][2],0,0,0))

    #Case3
    list_1_3_e1 = []
    list_2_3_e1 = []

    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5):
            list_1_3_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 9) & (1 <= list_0_e1[i][1] <= 14):
            list_1_3_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4):
            list_2_3_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 6) & (1 <= list_0_e1[i][0] <= 13):
            list_2_3_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 8) & (1 <= list_0_e1[i][0] <= 14):
            list_2_3_e1.append(list_0_e1[i])

    list_random_1_3_e1 = random.sample(list_1_3_e1,len(list_1_3_e1))
```

```
    list_random_2_3_e1 = random.sample(list_2_3_e1,len(list_2_3_e1))
    list_correct_3_e1 = list_random_1_3_e1 + list_random_2_3_e1

    list_late_3_e1 = random.sample(list_correct_3_e1,round(N1*RL))
    list_no_late_3_e1 = [x for x in list_correct_3_e1 if x not in list_late_3_e1]

    list_bad_3_e1 = random.sample(list_no_late_3_e1,round(N1*RJ))

    random_3_e1 = []
    for i in range (len(list_no_late_3_e1)):
        for j in range (len(list_bad_3_e1)):
            if list_no_late_3_e1[i] == list_bad_3_e1[j]:
                random_3_e1.append(i)

    list_bad_3_e1 = []
    for i in range (len(random_3_e1)):
        list_bad_3_e1.append(list_no_late_3_e1[random_3_e1[i]])

    list_no_bad_3_e1 = [x for x in list_no_late_3_e1 if x not in list_bad_3_e1]

    no_random_3_e1 = []
    for i in range (len(list_no_late_3_e1)):
        for j in range (len(list_no_bad_3_e1)):
            if list_no_late_3_e1[i] == list_no_bad_3_e1[j]:
                no_random_3_e1.append(i)

    normal_3_e1 = []
    for i in range (len(random_3_e1)):
        normal_3_e1.append(np.random.normal(random_3_e1[i], 5))

    for i in range (len(normal_3_e1)):
        list_bad_3_e1[i].append(normal_3_e1[i])

    list_bad_3_e1 = sorted(list_bad_3_e1, key=lambda x: x[-1])

    for i in range (len(normal_3_e1)):
        del list_bad_3_e1[i][3]

    normal_3_e1 = sorted(normal_3_e1)

    list_shift_3_e1 = []
    j=0;
    for i in range (len(list_no_bad_3_e1)+len(list_bad_3_e1)):
        if (len(list_bad_3_e1) != 0) & (j < len(no_random_3_e1)):
            if normal_3_e1[0] < no_random_3_e1[j]:
                list_shift_3_e1.append(list_bad_3_e1[0])
                del list_bad_3_e1[0]
                del normal_3_e1[0]
            elif normal_3_e1[0] > no_random_3_e1[j]:
                list_shift_3_e1.append(list_no_bad_3_e1[j])
                j=j+1
        elif j >= len(no_random_3_e1):
            list_shift_3_e1.append(list_bad_3_e1[0])
            del list_bad_3_e1[0]
            del normal_3_e1[0]
        else:
            list_shift_3_e1.append(list_no_bad_3_e1[j])
            j=j+1

    list_3_e1 = list_shift_3_e1 + list_late_3_e1
```

```
    class agent_3_e1:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_3_e1 = []
    for i in range(len(list_0_e1)):
        passenger_3_e1.append(agent_3_e1(list_3_e1[i][0],list_3_e1[i][1],
    list_3_e1[i][2],0,0,0))

    list_1_3_e2 = []
    list_2_3_e2 = []
    list_3_3_e2 = []
    list_4_3_e2 = []

    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5):
            list_1_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 9) & (1 <= list_0_e2[i][1] <= 14):
            list_1_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4):
            list_2_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 6) & (1 <= list_0_e2[i][0] <= 13):
            list_2_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 8) & (1 <= list_0_e2[i][0] <= 14):
            list_2_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 9) & (20 <= list_0_e2[i][1] <= 40):
            list_3_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 6) & (20 <= list_0_e2[i][0] <= 40):
            list_4_3_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 8) & (20 <= list_0_e2[i][0] <= 40):
            list_4_3_e2.append(list_0_e2[i])

    list_random_1_3_e2 = random.sample(list_1_3_e2,len(list_1_3_e2))
    list_random_2_3_e2 = random.sample(list_2_3_e2,len(list_2_3_e2))
    list_random_3_3_e2 = random.sample(list_3_3_e2,len(list_3_3_e2))
    list_random_4_3_e2 = random.sample(list_4_3_e2,len(list_4_3_e2))
    list_correct_3_e2 = list_random_1_3_e2 + list_random_2_3_e2 +
    list_random_3_3_e2 + list_random_4_3_e2

    list_late_3_e2 = random.sample(list_correct_3_e2,round(N2*RL))
    list_no_late_3_e2 = [x for x in list_correct_3_e2 if x not in list_late_3_e2]

    list_bad_3_e2 = random.sample(list_no_late_3_e2,round(N2*RJ))

    random_3_e2 = []
    for i in range (len(list_no_late_3_e2)):
        for j in range (len(list_bad_3_e2)):
```

```
            if list_no_late_3_e2[i] == list_bad_3_e2[j]:
                random_3_e2.append(i)

    list_bad_3_e2 = []
    for i in range (len(random_3_e2)):
        list_bad_3_e2.append(list_no_late_3_e2[random_3_e2[i]])

    list_no_bad_3_e2 = [x for x in list_no_late_3_e2 if x not in list_bad_3_e2]

    no_random_3_e2 = []
    for i in range (len(list_no_late_3_e2)):
        for j in range (len(list_no_bad_3_e2)):
            if list_no_late_3_e2[i] == list_no_bad_3_e2[j]:
                no_random_3_e2.append(i)

    normal_3_e2 = []
    for i in range (len(random_3_e2)):
        normal_3_e2.append(np.random.normal(random_3_e2[i], 5))

    for i in range (len(normal_3_e2)):
        list_bad_3_e2[i].append(normal_3_e2[i])

    list_bad_3_e2 = sorted(list_bad_3_e2, key=lambda x: x[-1])

    for i in range (len(normal_3_e2)):
        del list_bad_3_e2[i][3]

    normal_3_e2 = sorted(normal_3_e2)

    list_shift_3_e2 = []
    j=0;
    for i in range (len(list_no_bad_3_e2)+len(list_bad_3_e2)):
        if (len(list_bad_3_e2) != 0) & (j < len(no_random_3_e2)):
            if normal_3_e2[0] < no_random_3_e2[j]:
                list_shift_3_e2.append(list_bad_3_e2[0])
                del list_bad_3_e2[0]
                del normal_3_e2[0]
            elif normal_3_e2[0] > no_random_3_e2[j]:
                list_shift_3_e2.append(list_no_bad_3_e2[j])
                j=j+1
        elif j >= len(no_random_3_e2):
            list_shift_3_e2.append(list_bad_3_e2[0])
            del list_bad_3_e2[0]
            del normal_3_e2[0]
        else:
            list_shift_3_e2.append(list_no_bad_3_e2[j])
            j=j+1

    list_3_e2 = list_shift_3_e2 + list_late_3_e2

    class agent_3_e2:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_3_e2 = []
```

```
    for i in range(len(list_0_e2)):
        passenger_3_e2.append(agent_3_e2(list_3_e2[i][0],list_3_e2[i][1],
list_3_e2[i][2],0,0,0))

    #Case4
    list_1_4_e1 = []
    list_2_4_e1 = []
    list_3_4_e1 = []
    list_4_4_e1 = []

    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5):
            list_1_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 9) & (1 <= list_0_e1[i][1] <= 14):
            list_3_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4):
            list_2_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 6) & (1 <= list_0_e1[i][0] <= 13):
            list_4_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 8) & (1 <= list_0_e1[i][0] <= 14):
            list_4_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 9) & (20 <= list_0_e1[i][1] <= 40):
            list_1_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 6) & (20 <= list_0_e1[i][0] <= 40):
            list_2_4_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 8) & (20 <= list_0_e1[i][0] <= 40):
            list_2_4_e1.append(list_0_e1[i])

    list_random_1_4_e1 = random.sample(list_1_4_e1,len(list_1_4_e1))
    list_random_2_4_e1 = random.sample(list_2_4_e1,len(list_2_4_e1))
    list_random_3_4_e1 = random.sample(list_3_4_e1,len(list_3_4_e1))
    list_random_4_4_e1 = random.sample(list_4_4_e1,len(list_4_4_e1))
    list_correct_4_e1 = list_random_1_4_e1 + list_random_2_4_e1 +
list_random_3_4_e1 + list_random_4_4_e1

    list_late_4_e1 = random.sample(list_correct_4_e1,round(N1*RL))
    list_no_late_4_e1 = [x for x in list_correct_4_e1 if x not in list_late_4_e1]

    list_bad_4_e1 = random.sample(list_no_late_4_e1,round(N1*RJ))

    random_4_e1 = []
    for i in range (len(list_no_late_4_e1)):
        for j in range (len(list_bad_4_e1)):
            if list_no_late_4_e1[i] == list_bad_4_e1[j]:
                random_4_e1.append(i)

    list_bad_4_e1 = []
    for i in range (len(random_4_e1)):
        list_bad_4_e1.append(list_no_late_4_e1[random_4_e1[i]])

    list_no_bad_4_e1 = [x for x in list_no_late_4_e1 if x not in list_bad_4_e1]

    no_random_4_e1 = []
```

```
    for i in range (len(list_no_late_4_e1)):
        for j in range (len(list_no_bad_4_e1)):
            if list_no_late_4_e1[i] == list_no_bad_4_e1[j]:
                no_random_4_e1.append(i)

    normal_4_e1 = []
    for i in range (len(random_4_e1)):
        normal_4_e1.append(np.random.normal(random_4_e1[i], 5))

    for i in range (len(normal_4_e1)):
        list_bad_4_e1[i].append(normal_4_e1[i])

    list_bad_4_e1 = sorted(list_bad_4_e1, key=lambda x: x[-1])

    for i in range (len(normal_4_e1)):
        del list_bad_4_e1[i][3]

    normal_4_e1 = sorted(normal_4_e1)

    list_shift_4_e1 = []
    j=0;
    for i in range (len(list_no_bad_4_e1)+len(list_bad_4_e1)):
        if (len(list_bad_4_e1) != 0) & (j < len(no_random_4_e1)):
            if normal_4_e1[0] < no_random_4_e1[j]:
                list_shift_4_e1.append(list_bad_4_e1[0])
                del list_bad_4_e1[0]
                del normal_4_e1[0]
            elif normal_4_e1[0] > no_random_4_e1[j]:
                list_shift_4_e1.append(list_no_bad_4_e1[j])
                j=j+1
        elif j >= len(no_random_4_e1):
            list_shift_4_e1.append(list_bad_4_e1[0])
            del list_bad_4_e1[0]
            del normal_4_e1[0]
        else:
            list_shift_4_e1.append(list_no_bad_4_e1[j])
            j=j+1

    list_4_e1 = list_shift_4_e1 + list_late_4_e1

    class agent_4_e1:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_4_e1 = []
    for i in range(len(list_0_e1)):
        passenger_4_e1.append(agent_4_e1(list_4_e1[i][0],list_4_e1[i][1],
list_4_e1[i][2],0,0,0))

    list_1_4_e2 = []
    list_2_4_e2 = []
    list_3_4_e2 = []
    list_4_4_e2 = []

    for i in range (len(list_0_e2)):
```

```
        if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5):
            list_1_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 9) & (1 <= list_0_e2[i][1] <= 14):
            list_3_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4):
            list_2_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 6) & (1 <= list_0_e2[i][0] <= 13):
            list_4_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 8) & (1 <= list_0_e2[i][0] <= 14):
            list_4_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 9) & (20 <= list_0_e2[i][1] <= 40):
            list_1_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 6) & (20 <= list_0_e2[i][0] <= 40):
            list_2_4_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 8) & (20 <= list_0_e2[i][0] <= 40):
            list_2_4_e2.append(list_0_e2[i])

    list_random_1_4_e2 = random.sample(list_1_4_e2,len(list_1_4_e2))
    list_random_2_4_e2 = random.sample(list_2_4_e2,len(list_2_4_e2))
    list_random_3_4_e2 = random.sample(list_3_4_e2,len(list_3_4_e2))
    list_random_4_4_e2 = random.sample(list_4_4_e2,len(list_4_4_e2))
    list_correct_4_e2 = list_random_1_4_e2 + list_random_2_4_e2 +
   list_random_3_4_e2 + list_random_4_4_e2

    list_late_4_e2 = random.sample(list_correct_4_e2,round(N2*RL))
    list_no_late_4_e2 = [x for x in list_correct_4_e2 if x not in list_late_4_e2]

    list_bad_4_e2 = random.sample(list_no_late_4_e2,round(N2*RJ))

    random_4_e2 = []
    for i in range (len(list_no_late_4_e2)):
        for j in range (len(list_bad_4_e2)):
            if list_no_late_4_e2[i] == list_bad_4_e2[j]:
                random_4_e2.append(i)

    list_bad_4_e2 = []
    for i in range (len(random_4_e2)):
        list_bad_4_e2.append(list_no_late_4_e2[random_4_e2[i]])

    list_no_bad_4_e2 = [x for x in list_no_late_4_e2 if x not in list_bad_4_e2]

    no_random_4_e2 = []
    for i in range (len(list_no_late_4_e2)):
        for j in range (len(list_no_bad_4_e2)):
            if list_no_late_4_e2[i] == list_no_bad_4_e2[j]:
                no_random_4_e2.append(i)

    normal_4_e2 = []
    for i in range (len(random_4_e2)):
        normal_4_e2.append(np.random.normal(random_4_e2[i], 5))

    for i in range (len(normal_4_e2)):
        list_bad_4_e2[i].append(normal_4_e2[i])
```

```
    list_bad_4_e2 = sorted(list_bad_4_e2, key=lambda x: x[-1])

    for i in range (len(normal_4_e2)):
        del list_bad_4_e2[i][3]

    normal_4_e2 = sorted(normal_4_e2)

    list_shift_4_e2 = []
    j=0;
    for i in range (len(list_no_bad_4_e2)+len(list_bad_4_e2)):
        if (len(list_bad_4_e2) != 0) & (j < len(no_random_4_e2)):
            if normal_4_e2[0] < no_random_4_e2[j]:
                list_shift_4_e2.append(list_bad_4_e2[0])
                del list_bad_4_e2[0]
                del normal_4_e2[0]
            elif normal_4_e2[0] > no_random_4_e2[j]:
                list_shift_4_e2.append(list_no_bad_4_e2[j])
                j=j+1
        elif j >= len(no_random_4_e2):
            list_shift_4_e2.append(list_bad_4_e2[0])
            del list_bad_4_e2[0]
            del normal_4_e2[0]
        else:
            list_shift_4_e2.append(list_no_bad_4_e2[j])
            j=j+1

    list_4_e2 = list_shift_4_e2 + list_late_4_e2

    class agent_4_e2:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_4_e2 = []
    for i in range(len(list_0_e2)):
        passenger_4_e2.append(agent_4_e2(list_4_e2[i][0],list_4_e2[i][1],
    list_4_e2[i][2],0,0,0))

    #Case5
    list_1_5_e1 = []
    list_2_5_e1 = []

    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5) | (list_0_e1[i][0] ==
    9):
            list_1_5_e1.append(list_0_e1[i])
    for i in range (len(list_0_e1)):
        if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4) | (list_0_e1[i][0] ==
    6) | (list_0_e1[i][0] == 8):
            list_2_5_e1.append(list_0_e1[i])

    list_random_1_5_e1 = random.sample(list_1_5_e1,len(list_1_5_e1))
    list_random_2_5_e1 = random.sample(list_2_5_e1,len(list_2_5_e1))
    list_correct_5_e1 = list_random_1_5_e1 + list_random_2_5_e1
```

```
    list_late_5_e1 = random.sample(list_correct_5_e1,round(N1*RL))
    list_no_late_5_e1 = [x for x in list_correct_5_e1 if x not in list_late_5_e1]

    list_bad_5_e1 = random.sample(list_no_late_5_e1,round(N1*RJ))

    random_5_e1 = []
    for i in range (len(list_no_late_5_e1)):
        for j in range (len(list_bad_5_e1)):
            if list_no_late_5_e1[i] == list_bad_5_e1[j]:
                random_5_e1.append(i)

    list_bad_5_e1 = []
    for i in range (len(random_5_e1)):
        list_bad_5_e1.append(list_no_late_5_e1[random_5_e1[i]])

    list_no_bad_5_e1 = [x for x in list_no_late_5_e1 if x not in list_bad_5_e1]

    no_random_5_e1 = []
    for i in range (len(list_no_late_5_e1)):
        for j in range (len(list_no_bad_5_e1)):
            if list_no_late_5_e1[i] == list_no_bad_5_e1[j]:
                no_random_5_e1.append(i)

    normal_5_e1 = []
    for i in range (len(random_5_e1)):
        normal_5_e1.append(np.random.normal(random_5_e1[i], 5))

    for i in range (len(normal_5_e1)):
        list_bad_5_e1[i].append(normal_5_e1[i])

    list_bad_5_e1 = sorted(list_bad_5_e1, key=lambda x: x[-1])

    for i in range (len(normal_5_e1)):
        del list_bad_5_e1[i][3]

    normal_5_e1 = sorted(normal_5_e1)

    list_shift_5_e1 = []
    j=0;
    for i in range (len(list_no_bad_5_e1)+len(list_bad_5_e1)):
        if (len(list_bad_5_e1) != 0) & (j < len(no_random_5_e1)):
            if normal_5_e1[0] < no_random_5_e1[j]:
                list_shift_5_e1.append(list_bad_5_e1[0])
                del list_bad_5_e1[0]
                del normal_5_e1[0]
            elif normal_5_e1[0] > no_random_5_e1[j]:
                list_shift_5_e1.append(list_no_bad_5_e1[j])
                j=j+1
        elif j >= len(no_random_5_e1):
            list_shift_5_e1.append(list_bad_5_e1[0])
            del list_bad_5_e1[0]
            del normal_5_e1[0]
        else:
            list_shift_5_e1.append(list_no_bad_5_e1[j])
            j=j+1

    list_5_e1 = list_shift_5_e1 + list_late_5_e1

    class agent_5_e1:
        def __init__(self,char,seat,bag,num_out,row,col):
```

```
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_5_e1 = []
    for i in range(len(list_0_e1)):
        passenger_5_e1.append(agent_5_e1(list_5_e1[i][0],list_5_e1[i][1],
    list_5_e1[i][2],0,0,0))

    list_1_5_e2 = []
    list_2_5_e2 = []

    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5) | (list_0_e2[i][0] ==
    9):
            list_1_5_e2.append(list_0_e2[i])
    for i in range (len(list_0_e2)):
        if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4) | (list_0_e2[i][0] ==
    6) | (list_0_e2[i][0] == 8):
            list_2_5_e2.append(list_0_e2[i])

    list_random_1_5_e2 = random.sample(list_1_5_e2,len(list_1_5_e2))
    list_random_2_5_e2 = random.sample(list_2_5_e2,len(list_2_5_e2))
    list_correct_5_e2 = list_random_1_5_e2 + list_random_2_5_e2

    list_late_5_e2 = random.sample(list_correct_5_e2,round(N2*RL))
    list_no_late_5_e2 = [x for x in list_correct_5_e2 if x not in list_late_5_e2]

    list_bad_5_e2 = random.sample(list_no_late_5_e2,round(N2*RJ))

    random_5_e2 = []
    for i in range (len(list_no_late_5_e2)):
        for j in range (len(list_bad_5_e2)):
            if list_no_late_5_e2[i] == list_bad_5_e2[j]:
                random_5_e2.append(i)

    list_bad_5_e2 = []
    for i in range (len(random_5_e2)):
        list_bad_5_e2.append(list_no_late_5_e2[random_5_e2[i]])

    list_no_bad_5_e2 = [x for x in list_no_late_5_e2 if x not in list_bad_5_e2]

    no_random_5_e2 = []
    for i in range (len(list_no_late_5_e2)):
        for j in range (len(list_no_bad_5_e2)):
            if list_no_late_5_e2[i] == list_no_bad_5_e2[j]:
                no_random_5_e2.append(i)

    normal_5_e2 = []
    for i in range (len(random_5_e2)):
        normal_5_e2.append(np.random.normal(random_5_e2[i], 5))

    for i in range (len(normal_5_e2)):
        list_bad_5_e2[i].append(normal_5_e2[i])

    list_bad_5_e2 = sorted(list_bad_5_e2, key=lambda x: x[-1])

```

```
    for i in range (len(normal_5_e2)):
        del list_bad_5_e2[i][3]

    normal_5_e2 = sorted(normal_5_e2)

    list_shift_5_e2 = []
    j=0;
    for i in range (len(list_no_bad_5_e2)+len(list_bad_5_e2)):
        if (len(list_bad_5_e2) != 0) & (j < len(no_random_5_e2)):
            if normal_5_e2[0] < no_random_5_e2[j]:
                list_shift_5_e2.append(list_bad_5_e2[0])
                del list_bad_5_e2[0]
                del normal_5_e2[0]
            elif normal_5_e2[0] > no_random_5_e2[j]:
                list_shift_5_e2.append(list_no_bad_5_e2[j])
                j=j+1
        elif j >= len(no_random_5_e2):
            list_shift_5_e2.append(list_bad_5_e2[0])
            del list_bad_5_e2[0]
            del normal_5_e2[0]
        else:
            list_shift_5_e2.append(list_no_bad_5_e2[j])
            j=j+1

    list_5_e2 = list_shift_5_e2 + list_late_5_e2

    class agent_5_e2:
        def __init__(self,char,seat,bag,num_out,row,col):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.row = row
            self.col = col

    passenger_5_e2 = []
    for i in range(len(list_0_e2)):
        passenger_5_e2.append(agent_5_e2(list_5_e2[i][0],list_5_e2[i][1],
list_5_e2[i][2],0,0,0))

    class person:
        def __init__(self,char,seat,bag,num_out,t_1,t_2,check):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.num_out = num_out
            self.t_1 = t_1
            self.t_2 = t_2
            self.check = check

    list_pass = [[person(0,0,0,0,0,0,0) for i in range(0,41)] for j in range
(0,10)]

    class grid:
        def __init__(self,type,value,pass_char,pass_seat):
            self.type = type
            # -1 -> block
            # 0 -> cabin
            # 1 -> queue
            # 2 -> aisle
```

```
            # 3 -> seat
            self.value = value
            # 0 -> avaliable
            # 1 -> passenger
            self.pass_char = pass_char
            self.pass_seat = pass_seat

    plane = [[grid(-1,0,0,0) for i in range(0,42)] for j in range(0,250)]

    for i in range(1,3):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(4,7):
        for j in range(1,14):
            plane[i][j].type = 3

    for i in range(8,10):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(1,3):
        for j in range(20,41):
            plane[i][j].type = 3

    for i in range(4,7):
        for j in range(20,41):
            plane[i][j].type = 3

    for i in range(8,10):
        for j in range(20,41):
            plane[i][j].type = 3

    for i in range(3,250):
        plane[i][0].type = 1
        plane[i][41].type = 1

    q = [3,7]
    for i in q:
        for j in range(0,17):
            plane[i][j].type = 2
        for j in range(18,42):
            plane[i][j].type = 2

    for i in range(1,10):
        plane[i][17].type = 0

    def C1(passenger_1_e1,passenger_1_e2):
        for i in range(len(passenger_1_e1)):
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].char =
passenger_1_e1[i].char
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].seat =
passenger_1_e1[i].seat
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].bag =
passenger_1_e1[i].bag
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].num_out =
-1
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].t_1 = -1
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].t_2 = -1
            list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].check = 0
```

```
936         for i in range(0,len(passenger_1_e1)):
937             plane[8+i][0].value = 1
938             plane[8+i][0].pass_char = passenger_1_e1[i].char
939             plane[8+i][0].pass_seat = passenger_1_e1[i].seat
940         for i in range(len(passenger_1_e2)):
941             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].char =
    passenger_1_e2[i].char
942             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].seat =
    passenger_1_e2[i].seat
943             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].bag =
    passenger_1_e2[i].bag
944             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].num_out =
    -1
945             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].t_1 = -1
946             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].t_2 = -1
947             list_pass[passenger_1_e2[i].char][passenger_1_e2[i].seat].check = 0
948         for i in range(0,len(passenger_1_e2)):
949             plane[8+i][41].value = 1
950             plane[8+i][41].pass_char = passenger_1_e2[i].char
951             plane[8+i][41].pass_seat = passenger_1_e2[i].seat
952 
953     def C2(passenger_2_e1,passenger_2_e2):
954         for i in range(len(passenger_2_e1)):
955             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].char =
    passenger_2_e1[i].char
956             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].seat =
    passenger_2_e1[i].seat
957             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].bag =
    passenger_2_e1[i].bag
958             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].num_out =
    -1
959             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].t_1 = -1
960             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].t_2 = -1
961             list_pass[passenger_2_e1[i].char][passenger_2_e1[i].seat].check = 0
962         for i in range(0,len(passenger_2_e1)):
963             plane[8+i][0].value = 1
964             plane[8+i][0].pass_char = passenger_2_e1[i].char
965             plane[8+i][0].pass_seat = passenger_2_e1[i].seat
966         for i in range(len(passenger_2_e2)):
967             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].char =
    passenger_2_e2[i].char
968             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].seat =
    passenger_2_e2[i].seat
969             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].bag =
    passenger_2_e2[i].bag
970             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].num_out =
    -1
971             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].t_1 = -1
972             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].t_2 = -1
973             list_pass[passenger_2_e2[i].char][passenger_2_e2[i].seat].check = 0
974         for i in range(0,len(passenger_2_e2)):
975             plane[8+i][41].value = 1
976             plane[8+i][41].pass_char = passenger_2_e2[i].char
977             plane[8+i][41].pass_seat = passenger_2_e2[i].seat
978 
979     def C3(passenger_3_e1,passenger_3_e2):
980         for i in range(len(passenger_3_e1)):
981             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].char =
    passenger_3_e1[i].char
982             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].seat =
```

```
    passenger_3_e1[i].seat
983             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].bag =
    passenger_3_e1[i].bag
984             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].num_out =
    -1
985             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].t_1 = -1
986             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].t_2 = -1
987             list_pass[passenger_3_e1[i].char][passenger_3_e1[i].seat].check = 0
988         for i in range(0,len(passenger_3_e1)):
989             plane[8+i][0].value = 1
990             plane[8+i][0].pass_char = passenger_3_e1[i].char
991             plane[8+i][0].pass_seat = passenger_3_e1[i].seat
992         for i in range(len(passenger_3_e2)):
993             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].char =
    passenger_3_e2[i].char
994             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].seat =
    passenger_3_e2[i].seat
995             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].bag =
    passenger_3_e2[i].bag
996             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].num_out =
    -1
997             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].t_1 = -1
998             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].t_2 = -1
999             list_pass[passenger_3_e2[i].char][passenger_3_e2[i].seat].check = 0
1000        for i in range(0,len(passenger_3_e2)):
1001            plane[8+i][41].value = 1
1002            plane[8+i][41].pass_char = passenger_3_e2[i].char
1003            plane[8+i][41].pass_seat = passenger_3_e2[i].seat
1004
1005     def C4(passenger_4_e1,passenger_4_e2):
1006         for i in range(len(passenger_4_e1)):
1007             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].char =
    passenger_4_e1[i].char
1008             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].seat =
    passenger_4_e1[i].seat
1009             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].bag =
    passenger_4_e1[i].bag
1010             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].num_out =
    -1
1011             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].t_1 = -1
1012             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].t_2 = -1
1013             list_pass[passenger_4_e1[i].char][passenger_4_e1[i].seat].check = 0
1014         for i in range(0,len(passenger_4_e1)):
1015             plane[8+i][0].value = 1
1016             plane[8+i][0].pass_char = passenger_4_e1[i].char
1017             plane[8+i][0].pass_seat = passenger_4_e1[i].seat
1018         for i in range(len(passenger_4_e2)):
1019             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].char =
    passenger_4_e2[i].char
1020             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].seat =
    passenger_4_e2[i].seat
1021             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].bag =
    passenger_4_e2[i].bag
1022             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].num_out =
    -1
1023             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].t_1 = -1
1024             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].t_2 = -1
1025             list_pass[passenger_4_e2[i].char][passenger_4_e2[i].seat].check = 0
1026         for i in range(0,len(passenger_4_e2)):
1027             plane[8+i][41].value = 1
```

```
            plane[8+i][41].pass_char = passenger_4_e2[i].char
            plane[8+i][41].pass_seat = passenger_4_e2[i].seat

    def C5(passenger_5_e1,passenger_5_e2):
        for i in range(len(passenger_5_e1)):
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].char = passenger_5_e1[i].char
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].seat = passenger_5_e1[i].seat
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].bag = passenger_5_e1[i].bag
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].num_out = -1
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].t_1 = -1
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].t_2 = -1
            list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].check = 0
        for i in range(0,len(passenger_5_e1)):
            plane[8+i][0].value = 1
            plane[8+i][0].pass_char = passenger_5_e1[i].char
            plane[8+i][0].pass_seat = passenger_5_e1[i].seat
        for i in range(len(passenger_5_e2)):
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].char = passenger_5_e2[i].char
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].seat = passenger_5_e2[i].seat
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].bag = passenger_5_e2[i].bag
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].num_out = -1
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].t_1 = -1
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].t_2 = -1
            list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].check = 0
        for i in range(0,len(passenger_5_e2)):
            plane[8+i][41].value = 1
            plane[8+i][41].pass_char = passenger_5_e2[i].char
            plane[8+i][41].pass_seat = passenger_5_e2[i].seat

    if(case==1):
        C1(passenger_1_e1,passenger_1_e2)
    if(case==2):
        C2(passenger_2_e1,passenger_2_e2)
    if(case==3):
        C3(passenger_3_e1,passenger_3_e2)
    if(case==4):
        C4(passenger_4_e1,passenger_4_e2)
    if(case==5):
        C5(passenger_5_e1,passenger_5_e2)

    a = [[0 for i in range(0,42)] for i in range(0,250)]

    for i in range(0,250):
        for j in range(0,42):
            #print(plane[i][j].value)
            a[i][j] = plane[i][j].type

    time = 0

    """print(len(passenger_1_e1))
    print(len(passenger_1_e2))
```

```
    plt.figure('time'+str(time))
    im = plt.imshow(a)
    ax = plt.gca()
    #ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
    #ax.set_yticks(np.arange(-.5, 250, 1), minor=True)
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
    plt.savefig("figure_3.png")
    plt.show()"""

    def check_pass(plane):
        check = 0
        for w in range(1,10):
            for z in range(1,41):
                if(plane[w][z].type == 3 and plane[w][z].value == 1):
                    check+=1
        return check

    time = 0
    while(1):
        #check
        check = check_pass(plane)
        if(check==N):
            #print(time)
            break
        time+=1


        left_1 = [2,6]
        for i in left_1:
            for j in range(1,41):
                if(plane[i][j].type == 3 and plane[i][j].pass_char == i-1 and plane[i][j].value == 1 and plane[i-1][j].value == 0):
                    p1r = plane[i][j].pass_char
                    p1c = plane[i][j].pass_seat
                    plane[i-1][j].pass_char = p1r
                    plane[i-1][j].pass_seat = p1c
                    plane[i-1][j].value = 1
                    plane[i][j].pass_char = 0
                    plane[i][j].pass_seat = 0
                    plane[i][j].value = 0

        i = 8
        for j in range(1,41):
            if(plane[i][j].type == 3 and plane[i][j].pass_char == i+1 and plane[i][j].value == 1 and plane[i+1][j].value == 0):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                plane[i+1][j].pass_char = p1r
                plane[i+1][j].pass_seat = p1c
                plane[i+1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].value = 0

        #front
        aisle = [3,7]
        for i in aisle:
            for j in reversed(range(0,17)):
                if(plane[i][j].pass_char<5 and i==7):
                    continue
```

```
                if(plane[i][j].value==0):
                    continue
                if(plane[i][j].pass_seat==j):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2-=1
                        continue
                    else:
                        p1r = plane[i][j].pass_char
                        p1c = plane[i][j].pass_seat
                        if(plane[i][j].pass_char<i):
                            if(plane[i-list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out-1][j].value == 1):
                                p2r = plane[i-list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out-1][j].pass_char
                                p2c = plane[i-list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out-1][j].pass_seat
                                plane[i-1][j].pass_char = p2r
                                plane[i-1][j].pass_seat = p2c
                                plane[i-1][j].value = 1
                                plane[i-2][j].pass_char = p1r
                                plane[i-2][j].pass_seat = p1c
                                plane[i-2][j].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                                continue
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].pass_char = p1r
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].pass_seat = p1c
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].value = 1
                        else:
                            if(plane[i+list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out+1][j].value == 1):
                                p2r = plane[i+list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1][j].pass_char
                                p2c = plane[i+list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1][j].pass_seat
                                plane[i+1][j].pass_char = p2r
                                plane[i+1][j].pass_seat = p2c
                                plane[i+1][j].value = 1
                                plane[i+2][j].pass_char = p1r
                                plane[i+2][j].pass_seat = p1c
                                plane[i+2][j].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                                continue
                            if(plane[i+list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out+1][j].value == 1):
                                continue
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_char = p1r
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_seat = p1c
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value = 1
```

```
                        plane[i][j].pass_char = 0
                        plane[i][j].pass_seat = 0
                        plane[i][j].value = 0
                if(plane[i][j].pass_seat>j):
                    if(plane[i][j].pass_seat-j==1):
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag>0):
                            list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag-=1
                            continue
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].check==0):
                            sum = 0
                            if(plane[i][j].pass_char<i):
                                for m in range(i-2,i):
                                    if(plane[m][j+1].value == 1 and plane[m][j+1].pass_char!=m):
                                        sum+=1
                            else:
                                for m in range(i+1,i+3):
                                    if(plane[m][j+1].value == 1 and plane[m][j+1].pass_char!=m):
                                        sum+=1
                            if(sum!=0):
                                continue
                            sum = 0
                            if(plane[i][j].pass_char<i):
                                for m in reversed(range(plane[i][j].pass_char+1,i)):
                                    if(plane[m][j+1].value == 1):
                                        sum+=1
                            else:
                                for m in range(i+1,plane[i][j].pass_char):
                                    if(plane[m][j+1].value == 1):
                                        sum+=1
                            list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out = sum
                            mov = 0
                            for n in range(0,list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1):
                                if(plane[i][j+1+n].value==1):
                                    mov = 1
                            if(mov == 1):
                                continue
                            list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].check = 1
                            if(plane[i][j].pass_char<i):
                                if(plane[i][j].pass_char==i-1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2 = 0
                                if(plane[i][j].pass_char==i-2):
                                    if(plane[i-1][j+1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1 = 1
                                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2 = 2
                                    else:
                                        list_pass[plane[i][j].pass_char][plane[i
```

```
][j].pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                            else:
                                if(plane[i][j].pass_char==i+1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
                                if(plane[i][j].pass_char==i+2):
                                    if(plane[i+1][j+1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                    else:
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].check == 1):
                            if(list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1>0):
                                list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1-=1
                            else:
                                if(plane[i][j+1].value==1):
                                    continue
                                p1r = plane[i][j].pass_char
                                p1c = plane[i][j].pass_seat
                                plane[i][j+1].pass_char = p1r
                                plane[i][j+1].pass_seat = p1c
                                plane[i][j+1].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                    else:
                        if(plane[i][j+1].value==0):
                            p1r = plane[i][j].pass_char
                            p1c = plane[i][j].pass_seat
                            plane[i][j+1].pass_char = p1r
                            plane[i][j+1].pass_seat = p1c
                            plane[i][j+1].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0

        #back
        for i in aisle:
            for j in range(18,42):
                if(plane[i][j].pass_char<5 and i==7):
                    continue
                if(plane[i][j].value==0):
                    continue
                if(plane[i][j].pass_seat==j):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2-=1
```

```
                        continue
                    else:
                        p1r = plane[i][j].pass_char
                        p1c = plane[i][j].pass_seat
                        if(plane[i][j].pass_char<i):
                            if(plane[i-list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out-1][j].value == 1):
                                p2r = plane[i-list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out-1][j].pass_char
                                p2c = plane[i-list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out-1][j].pass_seat
                                plane[i-1][j].pass_char = p2r
                                plane[i-1][j].pass_seat = p2c
                                plane[i-1][j].value = 1
                                plane[i-2][j].pass_char = p1r
                                plane[i-2][j].pass_seat = p1c
                                plane[i-2][j].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                                continue
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].pass_char = p1r
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].pass_seat = p1c
                            plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].value = 1
                        else:
                            if(plane[i+list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out+1][j].value == 1):
                                p2r = plane[i+list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1][j].pass_char
                                p2c = plane[i+list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1][j].pass_seat
                                plane[i+1][j].pass_char = p2r
                                plane[i+1][j].pass_seat = p2c
                                plane[i+1][j].value = 1
                                plane[i+2][j].pass_char = p1r
                                plane[i+2][j].pass_seat = p1c
                                plane[i+2][j].value = 1
                                plane[i][j].pass_char = 0
                                plane[i][j].pass_seat = 0
                                plane[i][j].value = 0
                                continue
                            if(plane[i+list_pass[plane[i][j].pass_char][plane[i][
j].pass_seat].num_out+1][j].value == 1):
                                continue
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_char = p1r
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_seat = p1c
                            plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value = 1
                        plane[i][j].pass_char = 0
                        plane[i][j].pass_seat = 0
                        plane[i][j].value = 0
                if(plane[i][j].pass_seat<j):
                    if(j-plane[i][j].pass_seat-j==1):
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].bag>0):
```

```
                            list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].bag-=1
                            continue
                        if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].check==0):
                            sum = 0
                            if(plane[i][j].pass_char<i):
                                for m in range(i-2,i):
                                    if(plane[m][j-1].value == 1 and plane[m][j
-1].pass_char!=m):
                                        sum+=1
                            else:
                                for m in range(i+1,i+3):
                                    if(plane[m][j-1].value == 1 and plane[m][j
-1].pass_char!=m):
                                        sum+=1
                            if(sum!=0):
                                continue
                            sum = 0
                            if(plane[i][j].pass_char<i):
                                for m in reversed(range(plane[i][j].pass_char+1,i
)):
                                    if(plane[m][j-1].value == 1):
                                        sum+=1
                            else:
                                for m in range(i+1,plane[i][j].pass_char):
                                    if(plane[m][j-1].value == 1):
                                        sum+=1
                            list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out = sum
                            mov = 0
                            for n in range(0,list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1):
                                if(plane[i][j-1-n].value==1):
                                    mov = 1
                            if(mov == 1):
                                continue
                            list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].check = 1
                            if(plane[i][j].pass_char<i):
                                if(plane[i][j].pass_char==i-1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
                                if(plane[i][j].pass_char==i-2):
                                    if(plane[i-1][j-1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
                                    else:
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
                            else:
                                if(plane[i][j].pass_char==i+1):
                                    list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
```

```
                                    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2 = 0
                                if(plane[i][j].pass_char==i+2):
                                    if(plane[i+1][j-1].value==1):
                                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1 = 1
                                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2 = 2
                                    else:
                                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1 = 0
                                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2 = 0
                            if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].check == 1):
                                if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1>0):
                                    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1-=1
                                else:
                                    if(plane[i][j-1].value==1):
                                        continue
                                    p1r = plane[i][j].pass_char
                                    p1c = plane[i][j].pass_seat
                                    plane[i][j-1].pass_char = p1r
                                    plane[i][j-1].pass_seat = p1c
                                    plane[i][j-1].value = 1
                                    plane[i][j].pass_char = 0
                                    plane[i][j].pass_seat = 0
                                    plane[i][j].value = 0
                            else:
                                if(plane[i][j-1].value==0):
                                    p1r = plane[i][j].pass_char
                                    p1c = plane[i][j].pass_seat
                                    plane[i][j-1].pass_char = p1r
                                    plane[i][j-1].pass_seat = p1c
                                    plane[i][j-1].value = 1
                                    plane[i][j].pass_char = 0
                                    plane[i][j].pass_seat = 0
                                    plane[i][j].value = 0

        j = 0
        for i in range(4,250):
            if(plane[i-1][j].value==1):
                continue
            else:
                if(plane[i][j].value==1):
                    if(plane[i][j].pass_char<5):
                        if(i>3):
                            p1r = plane[i][j].pass_char
                            p1c = plane[i][j].pass_seat
                            plane[i-1][j].pass_char = p1r
                            plane[i-1][j].pass_seat = p1c
                            plane[i-1][j].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0
                    else:
                        if(i>7):
                            p1r = plane[i][j].pass_char
```

```
                            p1c = plane[i][j].pass_seat
                            plane[i-1][j].pass_char = p1r
                            plane[i-1][j].pass_seat = p1c
                            plane[i-1][j].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0

        j = 41
        for i in range(4,250):
            if(plane[i-1][j].value==1):
                continue
            else:
                if(plane[i][j].value==1):
                    if(plane[i][j].pass_char<5):
                        if(i>3):
                            p1r = plane[i][j].pass_char
                            p1c = plane[i][j].pass_seat
                            plane[i-1][j].pass_char = p1r
                            plane[i-1][j].pass_seat = p1c
                            plane[i-1][j].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0
                    else:
                        if(i>7):
                            p1r = plane[i][j].pass_char
                            p1c = plane[i][j].pass_seat
                            plane[i-1][j].pass_char = p1r
                            plane[i-1][j].pass_seat = p1c
                            plane[i-1][j].value = 1
                            plane[i][j].pass_char = 0
                            plane[i][j].pass_seat = 0
                            plane[i][j].value = 0

        #print("time",time)
        #print("check",check)


        """print(plane[8][0].pass_char,plane[8][0].pass_seat)
        print(plane[8][41].pass_char,plane[8][41].pass_seat)
        for i in range(0,250):
            for j in range(0,42):
                #print(plane[i][j].value)
                a[i][j] = plane[i][j].value
        plt.figure('time'+str(time))
        im = plt.imshow(a[0:10])
        ax = plt.gca()
        ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
        ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
        ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
        plt.savefig('time'+str(time)+'.png')
        #plt.show()"""
    return time

print(run3(2,0.3,0.5,242))
```

B.4 Codes of Disembarking Process, Airplane I

```
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import math
import statistics as st
import random
from statistics import stdev
from scipy.integrate import quad
import copy

def run1(case,RL,RJ,N):
    # Create position of people in "Narrow Body" Passenger Aircraft
    arr = np.arange(1,196)
    list_0 = arr.tolist()
    for i in range (99):
        list_0[i] = [math.floor(list_0[i]/33)+1,list_0[i]%33]
    for i in range (99,195):
        list_0[i] = [math.floor((list_0[i]-99)/32)+5,(list_0[i]-99)%32+1]
    list_0[32] = [1,33]
    list_0[65] = [2,33]
    list_0[98] = [3,33]
    list_0[130] = [5,33]
    list_0[162] = [6,33]
    list_0[194] = [7,33]

    # Random luggage stow time of each people by experimental data and Weilbell distribution
    luggage = [9.6, 8.2, 7.5, 9.2, 8.1, 7.8, 6.8, 5.5, 5.1, 6.3, 5.7, 6.2, 4.9, 5.5, 6.1, 6.6, 8.1, 5.5, 6.8, 8.5, 9.0, 6.9, 9.2, 6.0, 5.9, 5.7, 7.3, 7.4, 6.1, 3.3, 6.0, 8.2, 8.6, 8.7, 7.5, 7.4, 9.1, 7.4, 7.4, 4.8, 6.8, 3.7, 4.8, 5.0, 5.7, 7.7, 7.6, 7.5, 6.5, 5.2, 9.1, 8.9, 9.0, 7.7, 6.8, 7.1, 9.2, 8.2, 10.2, 10.0, 9.3, 8.9, 8.3, 7.7, 7.9, 7.2, 5.8, 6.1, 6.1, 10.0, 9.5, 9.6, 9.3, 5.5, 5.0, 0.9, 2.7, 5.8, 7.3, 5.5, 7.8, 8.1, 6.2, 7.5, 6.1, 5.3, 9.8, 6.7, 7.5]
    luggage_arr = np.array(luggage)
    mean = luggage_arr.mean()
    std = stdev(luggage_arr)
    k = (std/mean)**(-1.086)
    z = 1 + 1/k
    def f(x):
        return math.exp(-x)*(x**(z-1))
    gamma,err = quad(f, 0, math.inf)
    c = mean/gamma
    for i in range (len(list_0)):
        weibull = (c*(np.random.weibull(k, 1))).tolist()
        time = round((weibull[0]/1.42))
        list_0[i].append(time)

    #Case1
    list_1 = copy.deepcopy(list_0)
    pri_max_1 = 1

    for i in range (len(list_1)):
        list_1[i].append(1)

    list_late_1 = random.sample(list_1,round(N*RL))

    for i in range (len(list_late_1)):
        for j in range (len(list_1)):
            if list_late_1[i] == list_1[j]:
                list_1[j][3] = pri_max_1+1
```

```
                if list_1[j][0] == 3:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 1)
| (list_1[k][0] == 2)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 2:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 1):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 5:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 6)
| (list_1[k][0] == 7)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 6:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 7):
                            list_1[k][3] = pri_max_1+1

    class agent_1:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_1 = []
    for i in range(len(list_1)):
        passenger_1.append(agent_1(list_1[i][0],list_1[i][1],list_1[i][2],list_1[
i][3]))


    #Case2
    list_2 = copy.deepcopy(list_0)
    pri_max_2 = 3

    for i in range (len(list_2)):
        if (list_2[i][0] == 1) | (list_2[i][0] == 7):
            list_2[i].append(3)
        elif (list_2[i][0] == 2) | (list_2[i][0] == 6):
            list_2[i].append(2)
        elif (list_2[i][0] == 3) | (list_2[i][0] == 5):
            list_2[i].append(1)

    list_late_2 = random.sample(list_2,round(N*RL))

    for i in range (len(list_late_2)):
        for j in range (len(list_2)):
            if list_late_2[i] == list_2[j]:
                list_2[j][3] = pri_max_2+1
                if list_2[j][0] == 3:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 1)
| (list_2[k][0] == 2)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 2:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 1):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 5:
```

```
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 6)
| (list_2[k][0] == 7)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 6:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 7):
                            list_2[k][3] = pri_max_2+1

    class agent_2:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_2 = []
    for i in range(len(list_2)):
        passenger_2.append(agent_2(list_2[i][0],list_2[i][1],list_2[i][2],list_2[
i][3]))

    #Case3
    list_3 = copy.deepcopy(list_0)
    pri_max_3 = 5

    for i in range (len(list_3)):
        if (list_3[i][0] == 3) & (1 <= list_3[i][1] <= 19):
            list_3[i].append(1)
        elif (list_3[i][0] == 5) & (2 <= list_3[i][1] <= 19):
            list_3[i].append(1)
        elif (list_3[i][0] == 2) & (1 <= list_3[i][1] <= 7):
            list_3[i].append(2)
        elif (list_3[i][0] == 6) & (2 <= list_3[i][1] <= 7):
            list_3[i].append(2)
        elif (list_3[i][0] == 3) & (20 <= list_3[i][1] <= 33):
            list_3[i].append(2)
        elif (list_3[i][0] == 5) & (20 <= list_3[i][1] <= 33):
            list_3[i].append(2)
        elif (list_3[i][0] == 1) & (1 <= list_3[i][1] <= 7):
            list_3[i].append(3)
        elif (list_3[i][0] == 2) & (8 <= list_3[i][1] <= 19):
            list_3[i].append(3)
        elif (list_3[i][0] == 6) & (8 <= list_3[i][1] <= 19):
            list_3[i].append(3)
        elif (list_3[i][0] == 7) & (2 <= list_3[i][1] <= 7):
            list_3[i].append(3)
        elif (list_3[i][0] == 1) & (8 <= list_3[i][1] <= 12):
            list_3[i].append(4)
        elif (list_3[i][0] == 2) & (20 <= list_3[i][1] <= 33):
            list_3[i].append(4)
        elif (list_3[i][0] == 6) & (20 <= list_3[i][1] <= 33):
            list_3[i].append(4)
        elif (list_3[i][0] == 7) & (8 <= list_3[i][1] <= 12):
            list_3[i].append(4)
        elif (list_3[i][0] == 1) & (13 <= list_3[i][1] <= 33):
            list_3[i].append(5)
        elif (list_3[i][0] == 7) & (13 <= list_3[i][1] <= 33):
            list_3[i].append(5)


```

```
    list_late_3 = random.sample(list_3,round(N*RL))

    for i in range (len(list_late_3)):
        for j in range (len(list_3)):
            if list_late_3[i] == list_3[j]:
                list_3[j][3] = pri_max_3+1
                if list_3[j][0] == 3:
                    for k in range (len(list_3)):
                        if (list_3[k][1] == list_3[j][1]) & ((list_3[k][0] == 1)
| (list_3[k][0] == 2)):
                            list_3[k][3] = pri_max_3+1
                if list_3[j][0] == 2:
                    for k in range (len(list_3)):
                        if (list_3[k][1] == list_3[j][1]) & (list_3[k][0] == 1):
                            list_3[k][3] = pri_max_3+1
                if list_3[j][0] == 5:
                    for k in range (len(list_3)):
                        if (list_3[k][1] == list_3[j][1]) & ((list_3[k][0] == 6)
| (list_3[k][0] == 7)):
                            list_3[k][3] = pri_max_3+1
                if list_3[j][0] == 6:
                    for k in range (len(list_3)):
                        if (list_3[k][1] == list_3[j][1]) & (list_3[k][0] == 7):
                            list_3[k][3] = pri_max_3+1

    class agent_3:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_3 = []
    for i in range(len(list_3)):
        passenger_3.append(agent_3(list_3[i][0],list_3[i][1],list_3[i][2],list_3[
i][3]))

    class person:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    list_pass = [[person(0,0,0,0) for i in range(0,34)] for j in range(0,8)]

    class grid:
        def __init__(self,type,value,pass_char,pass_seat,pass_pri):
            self.type = type
            # 0 -> block
            # 1 -> queue
            # 2 -> aisle
            # 3 -> seat
            self.value = value
            # 0 -> avaliable
            # 1 -> passenger
            self.pass_char = pass_char
            self.pass_seat = pass_seat
            self.pass_pri = pass_pri

```

```
    plane = [[grid(0,0,0,0,0) for i in range(0,36)] for i in range(0,203)]

    for i in range(1,8):
        for j in range(1,34):
            plane[i][j].type = 3

    for i in range(5,8):
        plane[i][1].type = 0

    for i in range(0,36):
        plane[4][i].type = 2

    for i in range(5,203):
        plane[i][0].type = 1

    def C1(passenger_1):
        for i in range(len(passenger_1)):
            list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
            list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
            list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
            list_pass[passenger_1[i].char][passenger_1[i].seat].pri = passenger_1
[i].pri
            plane[passenger_1[i].char][passenger_1[i].seat].value = 1
            plane[passenger_1[i].char][passenger_1[i].seat].pass_char =
passenger_1[i].char
            plane[passenger_1[i].char][passenger_1[i].seat].pass_seat =
passenger_1[i].seat
            plane[passenger_1[i].char][passenger_1[i].seat].pass_pri =
passenger_1[i].pri

    def C2(passenger_2):
        for i in range(len(passenger_2)):
            list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
            list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
passenger_2[i].seat
            list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
[i].bag
            list_pass[passenger_2[i].char][passenger_2[i].seat].pri = passenger_2
[i].pri
            plane[passenger_2[i].char][passenger_2[i].seat].value = 1
            plane[passenger_2[i].char][passenger_2[i].seat].pass_char =
passenger_2[i].char
            plane[passenger_2[i].char][passenger_2[i].seat].pass_seat =
passenger_2[i].seat
            plane[passenger_2[i].char][passenger_2[i].seat].pass_pri =
passenger_2[i].pri

    def C3(passenger_3):
        for i in range(len(passenger_3)):
            list_pass[passenger_3[i].char][passenger_3[i].seat].char =
passenger_3[i].char
            list_pass[passenger_3[i].char][passenger_3[i].seat].seat =
passenger_3[i].seat
            list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3
[i].bag
            list_pass[passenger_3[i].char][passenger_3[i].seat].pri = passenger_3
```

```
[i].pri
269             plane[passenger_3[i].char][passenger_3[i].seat].value = 1
270             plane[passenger_3[i].char][passenger_3[i].seat].pass_char =
    passenger_3[i].char
271             plane[passenger_3[i].char][passenger_3[i].seat].pass_seat =
    passenger_3[i].seat
272             plane[passenger_3[i].char][passenger_3[i].seat].pass_pri =
    passenger_3[i].pri
273
274     if(case==1):
275         C1(passenger_1)
276     if(case==2):
277         C2(passenger_2)
278     if(case==3):
279         C3(passenger_3)
280
281     time = 0
282
283     a = [[0 for i in range(0,36)] for i in range(0,203)]
284
285     for i in range(0,203):
286             for j in range(0,36):
287                 #print(plane[i][j].value)
288                 a[i][j] = plane[i][j].value
289
290     plt.figure('time'+str(time))
291     im = plt.imshow(a[0:8])
292     ax = plt.gca()
293     ax.set_xticks(np.arange(-.5, 36, 1), minor=True)
294     ax.set_yticks(np.arange(-.5, 8, 1), minor=True)
295     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
296     plt.savefig('figure_1_d.png')
297     #plt.show()
298
299     def check_pass(plane):
300         check = 0
301         for w in range(0,8):
302             for z in range(0,36):
303                 if(plane[w][z].type >0 and plane[w][z].value == 1):
304                     check+=1
305         return check
306
307     def min_pri(plane):
308         x = 10
309         for w in range(0,8):
310             for z in range(0,36):
311                 if(plane[w][z].type>0):
312                     if(plane[w][z].value==1):
313                         if(plane[w][z].pass_pri< x):
314                             x = plane[w][z].pass_pri
315         return x
316
317     i = 3
318     for j in range(1,34):
319         if(plane[i][j].value==0 and plane[i-1][j].value==1):
320             p1r = plane[i-1][j].pass_char
321             p1c = plane[i-1][j].pass_seat
322             p1p = plane[i-1][j].pass_pri
323             plane[i][j].pass_char = p1r
324             plane[i][j].pass_seat = p1c
```

```
            plane[i][j].pass_pri = p1p
            plane[i][j].value = 1
            plane[i-1][j].pass_char = 0
            plane[i-1][j].pass_seat = 0
            plane[i-1][j].pass_pri = 0
            plane[i-1][j].value = 0

    i = 5
    for j in range(2,34):
        if(plane[i][j].value==0 and plane[i+1][j].value==1):
            p1r = plane[i+1][j].pass_char
            p1c = plane[i+1][j].pass_seat
            p1p = plane[i+1][j].pass_pri
            plane[i][j].pass_char = p1r
            plane[i][j].pass_seat = p1c
            plane[i][j].pass_pri = p1p
            plane[i][j].value = 1
            plane[i+1][j].pass_char = 0
            plane[i+1][j].pass_seat = 0
            plane[i+1][j].pass_pri = 0
            plane[i+1][j].value = 0

    i = 2
    for j in range(1,34):
        if(plane[i][j].value==0 and plane[i-1][j].value==1):
            p1r = plane[i-1][j].pass_char
            p1c = plane[i-1][j].pass_seat
            p1p = plane[i-1][j].pass_pri
            plane[i][j].pass_char = p1r
            plane[i][j].pass_seat = p1c
            plane[i][j].pass_pri = p1p
            plane[i][j].value = 1
            plane[i-1][j].pass_char = 0
            plane[i-1][j].pass_seat = 0
            plane[i-1][j].pass_pri = 0
            plane[i-1][j].value = 0

    i = 6
    for j in range(2,34):
        if(plane[i][j].value==0 and plane[i+1][j].value==1):
            p1r = plane[i+1][j].pass_char
            p1c = plane[i+1][j].pass_seat
            p1p = plane[i+1][j].pass_pri
            plane[i][j].pass_char = p1r
            plane[i][j].pass_seat = p1c
            plane[i][j].pass_pri = p1p
            plane[i][j].value = 1
            plane[i+1][j].pass_char = 0
            plane[i+1][j].pass_seat = 0
            plane[i+1][j].pass_pri = 0
            plane[i+1][j].value = 0

    time+=1

    """for i in range(0,203):
        for j in range(0,36):
            #print(plane[i][j].value)
            a[i][j] = plane[i][j].value
    plt.figure('time'+str(time))
    im = plt.imshow(a[0:8])
```

```
    ax = plt.gca()
    ax.set_xticks(np.arange(-.5, 36, 1), minor=True)
    ax.set_yticks(np.arange(-.5, 8, 1), minor=True)
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
    plt.savefig('time'+str(time)+'.png')"""

    while(1):
        x = min_pri(plane)
        #check
        check = check_pass(plane)
        if(check==0):
            break
        time+=1

        j = 0
        for i in reversed(range(4,202)):
            if(plane[i][j].value==1 and plane[i+1][j].value==0):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                p1p = plane[i][j].pass_pri
                plane[i+1][j].pass_char = p1r
                plane[i+1][j].pass_seat = p1c
                plane[i+1][j].pass_pri = p1p
                plane[i+1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].pass_pri = 0
                plane[i][j].value = 0

        i = 4
        for j in range(0,34):
            if(plane[i][j].value==1):
                if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag>0)
:
                    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag
-=1
                continue
            if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char][
plane[i][j+1].pass_seat].bag==0):
                p1r = plane[i][j+1].pass_char
                p1c = plane[i][j+1].pass_seat
                p1p = plane[i][j+1].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i][j+1].pass_char = 0
                plane[i][j+1].pass_seat = 0
                plane[i][j+1].pass_pri = 0
                plane[i][j+1].value = 0
                continue
            if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char][
plane[i][j+1].pass_seat].bag!=0):
                continue
            if(j==0):
                continue
            if(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1 and plane[i
+1][j].pass_pri==x and plane[i+1][j].value == 1):
                ch = random.choice([i-1,i+1])
                p1r = plane[ch][j].pass_char
```

```
                p1c = plane[ch][j].pass_seat
                p1p = plane[ch][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[ch][j].pass_char = 0
                plane[ch][j].pass_seat = 0
                plane[ch][j].pass_pri = 0
                plane[ch][j].value = 0
            elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
                p1r = plane[i-1][j].pass_char
                p1c = plane[i-1][j].pass_seat
                p1p = plane[i-1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i-1][j].pass_char = 0
                plane[i-1][j].pass_seat = 0
                plane[i-1][j].pass_pri = 0
                plane[i-1][j].value = 0
            elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                p1r = plane[i+1][j].pass_char
                p1c = plane[i+1][j].pass_seat
                p1p = plane[i+1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i+1][j].pass_char = 0
                plane[i+1][j].pass_seat = 0
                plane[i+1][j].pass_pri = 0
                plane[i+1][j].value = 0
            else:
                continue

        i = 3
        for j in range(1,34):
            if(plane[i][j].value==0 and plane[i-1][j].value==1):
                p1r = plane[i-1][j].pass_char
                p1c = plane[i-1][j].pass_seat
                p1p = plane[i-1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i-1][j].pass_char = 0
                plane[i-1][j].pass_seat = 0
                plane[i-1][j].pass_pri = 0
                plane[i-1][j].value = 0

        i = 5
        for j in range(2,34):
            if(plane[i][j].value==0 and plane[i+1][j].value==1):
                p1r = plane[i+1][j].pass_char
                p1c = plane[i+1][j].pass_seat
                p1p = plane[i+1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
```

```
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i+1][j].pass_char = 0
                plane[i+1][j].pass_seat = 0
                plane[i+1][j].pass_pri = 0
                plane[i+1][j].value = 0

        i = 2
        for j in range(1,34):
            if(plane[i][j].value==0 and plane[i-1][j].value==1):
                p1r = plane[i-1][j].pass_char
                p1c = plane[i-1][j].pass_seat
                p1p = plane[i-1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i-1][j].pass_char = 0
                plane[i-1][j].pass_seat = 0
                plane[i-1][j].pass_pri = 0
                plane[i-1][j].value = 0

        i = 6
        for j in range(2,34):
            if(plane[i][j].value==0 and plane[i+1][j].value==1):
                p1r = plane[i+1][j].pass_char
                p1c = plane[i+1][j].pass_seat
                p1p = plane[i+1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i+1][j].pass_char = 0
                plane[i+1][j].pass_seat = 0
                plane[i+1][j].pass_pri = 0
                plane[i+1][j].value = 0

        """j = 0
        i = 4
        if(plane[i][j].value==1 and plane[i+1][j].value==0):
            p1r = plane[i][j].pass_char
            p1c = plane[i][j].pass_seat
            p1p = plane[i][j].pass_pri
            plane[i+1][j].pass_char = p1r
            plane[i+1][j].pass_seat = p1c
            plane[i+1][j].pass_pri = p1p
            plane[i+1][j].value = 1
            plane[i][j].pass_char = 0
            plane[i][j].pass_seat = 0
            plane[i][j].pass_pri = 0
            plane[i][j].value = 0"""

        #print("time",time)
        #print("check",check)
        #print(plane[4][0].value,plane[4][0].pass_pri)
        #print(plane[5][0].value,plane[5][0].pass_pri)
        """i = 4
        for j in reversed(range(0,36)):
            if(plane[i][j].value==1):
                print("i = ",i,"j = ",j,"goal = ",plane[i][j].pass_char,plane[i][
```

```
    j].pass_seat,list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out,
    list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_1,list_pass[plane[i
    ][j].pass_char][plane[i][j].pass_seat].t_2,list_pass[plane[i][j].pass_char][
    plane[i][j].pass_seat].check)
560         print("\n")"""
561
562         """for i in range(0,203):
563             for j in range(0,36):
564                 #print(plane[i][j].value)
565                 a[i][j] = plane[i][j].value
566         plt.figure('time'+str(time))
567         im = plt.imshow(a[0:8])
568         ax = plt.gca()
569         ax.set_xticks(np.arange(-.5, 36, 1), minor=True)
570         ax.set_yticks(np.arange(-.5, 8, 1), minor=True)
571         ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
572         plt.savefig('time'+str(time)+'.png')"""
573         #plt.show()
574     return time
575
576 print(run1(1,0.1,0,195))
577 print(run1(2,0.1,0,195))
578 print(run1(3,0.1,0,195))
```

B.5 Codes of Disembarking Process, Airplane II

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import math
5  import statistics as st
6  import random
7  from statistics import stdev
8  from scipy.integrate import quad
9  import copy
10
11 def run2(case,RL,RJ,N):
12     # Create position of people in "Narrow Body" Passenger Aircraft
13     arr = np.arange(1,319)
14     list_0 = arr.tolist()
15     for i in range (33):
16         list_0[i] = [math.floor(list_0[i]/11)+1, (list_0[i]%11)+3]
17     for i in range (33,117):
18         list_0[i] = [math.floor((list_0[i]-33)/14)+5, ((list_0[i]-33)%14)]
19     for i in range (117,201):
20         list_0[i] = [math.floor((list_0[i]-117)/14)+12, ((list_0[i]-117)%14)]
21     for i in range (201,285):
22         list_0[i] = [math.floor((list_0[i]-201)/14)+19, ((list_0[i]-201)%14)]
23     for i in range (285,318):
24         list_0[i] = [math.floor((list_0[i]-285)/11)+26, ((list_0[i]-285)%11)+3]
25     for i in range (len(list_0)):
26         if list_0[i][1] == 13:
27             list_0[i+1] = [list_0[i][0], 14]
28
29     # Random luggage stow time of each people by experimental data and Weilbell
    distribution
30     luggage = [9.6, 8.2, 7.5, 9.2, 8.1, 7.8, 6.8, 5.5, 5.1, 6.3, 5.7, 6.2, 4.9,
    5.5, 6.1, 6.6, 8.1, 5.5, 6.8, 8.5, 9.0, 6.9, 9.2, 6.0, 5.9, 5.7, 7.3, 7.4,
    6.1, 3.3, 6.0, 8.2, 8.6, 8.7, 7.5, 7.4, 9.1, 7.4, 7.4, 4.8, 6.8, 3.7, 4.8,
    5.0, 5.7, 7.7, 7.6, 7.5, 6.5, 5.2, 9.1, 8.9, 9.0, 7.7, 6.8, 7.1, 9.2, 8.2,
```

```
    10.2, 10.0, 9.3, 8.9, 8.3, 7.7, 7.9, 7.2, 5.8, 6.1, 6.1, 10.0, 9.5, 9.6, 9.3,
    5.5, 5.0, 0.9, 2.7, 5.8, 7.3, 5.5, 7.8, 8.1, 6.2, 7.5, 6.1, 5.3, 9.8, 6.7,
    7.5]
     luggage_arr = np.array(luggage)
     mean = luggage_arr.mean()
     std = stdev(luggage_arr)
     k = (std/mean)**(-1.086)
     z = 1 + 1/k
     def f(x):
         return math.exp(-x)*(x**(z-1))
     gamma,err = quad(f, 0, math.inf)
     c = mean/gamma
     for i in range (len(list_0)):
         weibull = (c*(np.random.weibull(k, 1))).tolist()
         time = round((weibull[0]/1.42))
         list_0[i].append(time)

     #Case1
     list_1 = copy.deepcopy(list_0)
     pri_max_1 = 1

     for i in range (len(list_1)):
         list_1[i].append(1)

     list_late_1 = random.sample(list_1,round(N*RL))

     for i in range (len(list_late_1)):
         for j in range (len(list_1)):
             if list_late_1[i] == list_1[j]:
                 list_1[j][3] = pri_max_1+1
                 if list_1[j][0] == 3:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 1)
    | (list_1[k][0] == 2)):
                             list_1[k][3] = pri_max_1+1
                 if list_1[j][0] == 2:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 1):
                             list_1[k][3] = pri_max_1+1
                 if list_1[j][0] == 5:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 6)
    | (list_1[k][0] == 7)):
                             list_1[k][3] = pri_max_1+1
                 if list_1[j][0] == 6:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 7):
                             list_1[k][3] = pri_max_1+1
                 if list_1[j][0] == 10:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 9)
    | (list_1[k][0] == 8)):
                             list_1[k][3] = pri_max_1+1
                 if list_1[j][0] == 9:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 8):
                             list_1[k][3] = pri_max_1+1
                 if list_1[j][0] == 12:
                     for k in range (len(list_1)):
                         if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 13)
```

```
    | (list_1[k][0] == 14)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 13:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 14):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 17:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 16)
    | (list_1[k][0] == 15)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 16:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 15):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 19:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 20)
    | (list_1[k][0] == 21)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 20:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 21):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 24:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 23)
    | (list_1[k][0] == 22)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 23:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 22):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 26:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 27)
    | (list_1[k][0] == 28)):
                            list_1[k][3] = pri_max_1+1
                if list_1[j][0] == 27:
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 28):
                            list_1[k][3] = pri_max_1+1

    class agent_1:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_1 = []
    for i in range(len(list_1)):
        passenger_1.append(agent_1(list_1[i][0],list_1[i][1],list_1[i][2],list_1[
    i][3]))

    #Case2
    list_2 = copy.deepcopy(list_0)
    pri_max_2 = 3

    for i in range (len(list_2)):
```

```
        if (list_0[i][0] == 3) or (list_0[i][0] == 5) or (list_0[i][0] == 10) or (list_0[i][0] == 12) or (list_0[i][0] == 17) or (list_0[i][0] == 19) or (list_0[i][0] == 24) or (list_0[i][0] == 26):
            list_2[i].append(1)
        elif (list_0[i][0] == 2) or (list_0[i][0] == 6) or (list_0[i][0] == 9) or (list_0[i][0] == 13) or (list_0[i][0] == 16) or (list_0[i][0] == 20) or (list_0[i][0] == 23) or (list_0[i][0] == 27):
            list_2[i].append(2)
        elif (list_0[i][0] == 1) or (list_0[i][0] == 7) or (list_0[i][0] == 8) or (list_0[i][0] == 14) or (list_0[i][0] == 15) or (list_0[i][0] == 21) or (list_0[i][0] == 22) or (list_0[i][0] == 28):
            list_2[i].append(3)

    list_late_2 = random.sample(list_2,round(N*RL))

    for i in range (len(list_late_2)):
        for j in range (len(list_2)):
            if list_late_2[i] == list_2[j]:
                list_2[j][3] = pri_max_2+1
                if list_2[j][0] == 3:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 1) | (list_2[k][0] == 2)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 2:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 1):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 5:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 6) | (list_2[k][0] == 7)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 6:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 7):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 10:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 9) | (list_2[k][0] == 8)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 9:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 8):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 12:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 13) | (list_2[k][0] == 14)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 13:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 14):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 17:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 16) | (list_2[k][0] == 15)):
                            list_2[k][3] = pri_max_2+1
```

```
                if list_2[j][0] == 16:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 15):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 19:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 20)
    | (list_2[k][0] == 21)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 20:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 21):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 24:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 23)
    | (list_2[k][0] == 22)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 23:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 22):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 26:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 27)
    | (list_2[k][0] == 28)):
                            list_2[k][3] = pri_max_2+1
                if list_2[j][0] == 27:
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 28):
                            list_2[k][3] = pri_max_2+1

    class agent_2:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_2 = []
    for i in range(len(list_2)):
        passenger_2.append(agent_2(list_2[i][0],list_2[i][1],list_2[i][2],list_2[
    i][3]))


    class person:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    list_pass = [[person(0,0,0,0) for i in range(0,15)] for j in range(0,29)]

    class grid:
        def __init__(self,type,value,pass_char,pass_seat,pass_pri):
            self.type = type
            # 0 -> block
            # 1 -> queue
            # 2 -> aisle
```

```
            # 3 -> seat
            self.value = value
            # 0 -> avaliable
            # 1 -> passenger
            self.pass_char = pass_char
            self.pass_seat = pass_seat
            self.pass_pri = pass_pri

    plane = [[grid(0,0,0,0,0) for i in range(0,17)] for j in range(0,347)]

    for i in range(1,4):
        for j in range(4,15):
            plane[i][j].type = 3

    for i in range(5,11):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(12,18):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(19,25):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(26,29):
        for j in range(4,15):
            plane[i][j].type = 3

    for i in range(4,347):
        plane[i][0].type = 1

    q = [4,11,18,25]
    for i in q:
        for j in range(0,17):
            plane[i][j].type = 2

    def C1(passenger_1):
        for i in range(len(passenger_1)):
            list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
            list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
            list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
            list_pass[passenger_1[i].char][passenger_1[i].seat].pri = passenger_1
[i].pri
            plane[passenger_1[i].char][passenger_1[i].seat].value = 1
            plane[passenger_1[i].char][passenger_1[i].seat].pass_char =
passenger_1[i].char
            plane[passenger_1[i].char][passenger_1[i].seat].pass_seat =
passenger_1[i].seat
            plane[passenger_1[i].char][passenger_1[i].seat].pass_pri =
passenger_1[i].pri

    def C2(passenger_2):
        for i in range(len(passenger_2)):
            list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
```

```
            list_pass[passenger_2[i].char][passenger_2[i].seat].seat = passenger_2[i].seat
            list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2[i].bag
            list_pass[passenger_2[i].char][passenger_2[i].seat].pri = passenger_2[i].pri
            plane[passenger_2[i].char][passenger_2[i].seat].value = 1
            plane[passenger_2[i].char][passenger_2[i].seat].pass_char = passenger_2[i].char
            plane[passenger_2[i].char][passenger_2[i].seat].pass_seat = passenger_2[i].seat
            plane[passenger_2[i].char][passenger_2[i].seat].pass_pri = passenger_2[i].pri

    if(case==1):
        C1(passenger_1)
    if(case==2):
        C2(passenger_2)
    time = 0

    a = [[0 for i in range(0,17)] for i in range(0,347)]

    for i in range(0,347):
        for j in range(0,17):
            #print(plane[i][j].value)
            a[i][j] = plane[i][j].type

    time = 0

    """plt.figure('time'+str(time))
    im = plt.imshow(a[0:29])
    ax = plt.gca()
    ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
    ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
    plt.savefig("figure_2_d.png")
    plt.show()"""

    def check_pass(plane):
        check = 0
        for w in range(0,29):
            for z in range(0,17):
                if(plane[w][z].type>0 and plane[w][z].value == 1):
                    check+=1
        return check

    def min_pri(plane):
        x = 10
        for w in range(0,29):
            for z in range(0,17):
                if(plane[w][z].type>0):
                    if(plane[w][z].value==1):
                        if(plane[w][z].pass_pri<x):
                            x = plane[w][z].pass_pri
        return x

    left_1 = [3,10,17,24]
    for i in left_1:
        for j in range(1,15):
            if(plane[i][j].value==0 and plane[i-1][j].value==1):
```

```
                p1r = plane[i-1][j].pass_char
                p1c = plane[i-1][j].pass_seat
                p1p = plane[i-1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i-1][j].pass_char = 0
                plane[i-1][j].pass_seat = 0
                plane[i-1][j].pass_pri = 0
                plane[i-1][j].value = 0

    right_1 = [5,12,19,26]
    for i in right_1:
        for j in range(1,15):
            if(plane[i][j].value==0 and plane[i+1][j].value==1):
                p1r = plane[i+1][j].pass_char
                p1c = plane[i+1][j].pass_seat
                p1p = plane[i+1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i+1][j].pass_char = 0
                plane[i+1][j].pass_seat = 0
                plane[i+1][j].pass_pri = 0
                plane[i+1][j].value = 0

    left_2 = [2,9,16,23]
    for i in left_2:
        for j in range(1,15):
            if(plane[i][j].value==0 and plane[i-1][j].value==1):
                p1r = plane[i-1][j].pass_char
                p1c = plane[i-1][j].pass_seat
                p1p = plane[i-1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i-1][j].pass_char = 0
                plane[i-1][j].pass_seat = 0
                plane[i-1][j].pass_pri = 0
                plane[i-1][j].value = 0

    right_2 = [6,13,20,27]
    for i in right_2:
        for j in range(1,15):
            if(plane[i][j].value==0 and plane[i+1][j].value==1):
                p1r = plane[i+1][j].pass_char
                p1c = plane[i+1][j].pass_seat
                p1p = plane[i+1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i+1][j].pass_char = 0
                plane[i+1][j].pass_seat = 0
                plane[i+1][j].pass_pri = 0
                plane[i+1][j].value = 0

```

```
    time+=1

    for i in range(0,347):
        for j in range(0,17):
            #print(plane[i][j].value)
            a[i][j] = plane[i][j].value
    plt.figure('time'+str(time))
    im = plt.imshow(a[0:29])
    ax = plt.gca()
    ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
    ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
    plt.savefig('time'+str(time)+'.png')

    while(1):
        x = min_pri(plane)
        #check
        check = check_pass(plane)
        if(check==0):
            break
        time+=1

        j = 0
        for i in reversed(range(4,346)):
            if(plane[i][j].value==1 and plane[i+1][j].value==0):
                p1r = plane[i][j].pass_char
                p1c = plane[i][j].pass_seat
                p1p = plane[i][j].pass_pri
                plane[i+1][j].pass_char = p1r
                plane[i+1][j].pass_seat = p1c
                plane[i+1][j].pass_pri = p1p
                plane[i+1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].pass_pri = 0
                plane[i][j].value = 0

        aisle = [4,11,18,25]
        for i in aisle:
            for j in range(0,15):
                if(plane[i][j].value==1):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
                    continue
                if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
][plane[i][j+1].pass_seat].bag==0):
                    p1r = plane[i][j+1].pass_char
                    p1c = plane[i][j+1].pass_seat
                    p1p = plane[i][j+1].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i][j+1].pass_char = 0
                    plane[i][j+1].pass_seat = 0
                    plane[i][j+1].pass_pri = 0
                    plane[i][j+1].value = 0
                    continue
```

```
                if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
    ][plane[i][j+1].pass_seat].bag!=0):
                    continue
                if(j==0):
                    continue
                if(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1 and
    plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                    ch = random.choice([i-1,i+1])
                    p1r = plane[ch][j].pass_char
                    p1c = plane[ch][j].pass_seat
                    p1p = plane[ch][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[ch][j].pass_char = 0
                    plane[ch][j].pass_seat = 0
                    plane[ch][j].pass_pri = 0
                    plane[ch][j].value = 0
                elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
                    p1r = plane[i-1][j].pass_char
                    p1c = plane[i-1][j].pass_seat
                    p1p = plane[i-1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i-1][j].pass_char = 0
                    plane[i-1][j].pass_seat = 0
                    plane[i-1][j].pass_pri = 0
                    plane[i-1][j].value = 0
                elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                    p1r = plane[i+1][j].pass_char
                    p1c = plane[i+1][j].pass_seat
                    p1p = plane[i+1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i+1][j].pass_char = 0
                    plane[i+1][j].pass_seat = 0
                    plane[i+1][j].pass_pri = 0
                    plane[i+1][j].value = 0
                else:
                    continue

        left_1 = [3,10,17,24]
        for i in left_1:
            for j in range(1,15):
                if(plane[i][j].value==0 and plane[i-1][j].value==1):
                    p1r = plane[i-1][j].pass_char
                    p1c = plane[i-1][j].pass_seat
                    p1p = plane[i-1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i-1][j].pass_char = 0
                    plane[i-1][j].pass_seat = 0
                    plane[i-1][j].pass_pri = 0
```

```
                    plane[i-1][j].value = 0

        right_1 = [5,12,19,26]
        for i in right_1:
            for j in range(1,15):
                if(plane[i][j].value==0 and plane[i+1][j].value==1):
                    p1r = plane[i+1][j].pass_char
                    p1c = plane[i+1][j].pass_seat
                    p1p = plane[i+1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i+1][j].pass_char = 0
                    plane[i+1][j].pass_seat = 0
                    plane[i+1][j].pass_pri = 0
                    plane[i+1][j].value = 0

        left_2 = [2,9,16,23]
        for i in left_2:
            for j in range(1,15):
                if(plane[i][j].value==0 and plane[i-1][j].value==1):
                    p1r = plane[i-1][j].pass_char
                    p1c = plane[i-1][j].pass_seat
                    p1p = plane[i-1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i-1][j].pass_char = 0
                    plane[i-1][j].pass_seat = 0
                    plane[i-1][j].pass_pri = 0
                    plane[i-1][j].value = 0

        right_2 = [6,13,20,27]
        for i in right_2:
            for j in range(1,15):
                if(plane[i][j].value==0 and plane[i+1][j].value==1):
                    p1r = plane[i+1][j].pass_char
                    p1c = plane[i+1][j].pass_seat
                    p1p = plane[i+1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i+1][j].pass_char = 0
                    plane[i+1][j].pass_seat = 0
                    plane[i+1][j].pass_pri = 0
                    plane[i+1][j].value = 0

        """print("time",time)
        print("check",check)

        for i in range(0,347):
            for j in range(0,17):
                #print(plane[i][j].value)
                a[i][j] = plane[i][j].value
        plt.figure('time'+str(time))
        im = plt.imshow(a[0:29])
        ax = plt.gca()
```

```
        ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
        ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
        ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
        plt.savefig('time'+str(time)+'.png')"""
    return time

print(run2(1,0,0,318))
print(run2(2,0,0,318))
```

B.6 Codes of Disembarking Process, Airplane III

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import statistics as st
import random
from statistics import stdev
from scipy.integrate import quad
import copy

def run3(case,RL,RJ,N):
    # Create position of people in "Narrow Body" Passenger Aircraft
    arr = np.arange(1,243)
    list_0 = arr.tolist()
    for i in range (28):
        list_0[i] = [math.floor(list_0[i]/14)+1, (list_0[i]%14)]
    for i in range (28,67):
        list_0[i] = [math.floor((list_0[i]-28)/13)+4, ((list_0[i]-28)%13)]
    for i in range (67,95):
        list_0[i] = [math.floor((list_0[i]-67)/14)+8, ((list_0[i]-67)%14)]
    for i in range (95,137):
        list_0[i] = [math.floor((list_0[i]-95)/21)+1, ((list_0[i]-95)%21)+19]
    for i in range (137,200):
        list_0[i] = [math.floor((list_0[i]-137)/21)+4, ((list_0[i]-137)%21)+19]
    for i in range (200,242):
        list_0[i] = [math.floor((list_0[i]-200)/21)+8, ((list_0[i]-200)%21)+19]
    for i in range (242):
        if (list_0[i][1] == 0) | (list_0[i][1] == 19):
            list_0[i] = [list_0[i-1][0],list_0[i-1][1]+1]


    # Random luggage stow time of each people by experimental data and Weilbell distribution
    luggage = [9.6, 8.2, 7.5, 9.2, 8.1, 7.8, 6.8, 5.5, 5.1, 6.3, 5.7, 6.2, 4.9, 5.5, 6.1, 6.6, 8.1, 5.5, 6.8, 8.5, 9.0, 6.9, 9.2, 6.0, 5.9, 5.7, 7.3, 7.4, 6.1, 3.3, 6.0, 8.2, 8.6, 8.7, 7.5, 7.4, 9.1, 7.4, 7.4, 4.8, 6.8, 3.7, 4.8, 5.0, 5.7, 7.7, 7.6, 7.5, 6.5, 5.2, 9.1, 8.9, 9.0, 7.7, 6.8, 7.1, 9.2, 8.2, 10.2, 10.0, 9.3, 8.9, 8.3, 7.7, 7.9, 7.2, 5.8, 6.1, 6.1, 10.0, 9.5, 9.6, 9.3, 5.5, 5.0, 0.9, 2.7, 5.8, 7.3, 5.5, 7.8, 8.1, 6.2, 7.5, 6.1, 5.3, 9.8, 6.7, 7.5]
    luggage_arr = np.array(luggage)
    mean = luggage_arr.mean()
    std = stdev(luggage_arr)
    k = (std/mean)**(-1.086)
    z = 1 + 1/k
    def f(x):
        return math.exp(-x)*(x**(z-1))
    gamma,err = quad(f, 0, math.inf)
    c = mean/gamma
```

```
    for i in range (len(list_0)):
        weibull = (c*(np.random.weibull(k, 1))).tolist()
        time = round((weibull[0]/1.42))
        list_0[i].append(time)

    #Case1
    list_1 = copy.deepcopy(list_0)
    pri_max_1 = 1

    for i in range (len(list_1)):
        list_1[i].append(1)

    list_late_1 = random.sample(list_1,round(N*RL))

    for i in range (len(list_late_1)):
        for j in range (len(list_1)):
            if list_late_1[i] == list_1[j]:
                list_1[j][3] = pri_max_1+1
                if (list_1[j][0] == 4) or (list_1[j][0] == 6):
                    for k in range (len(list_1)):
                        if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 5):
                            list_1[k][3] = pri_max_1+1

    class agent_1:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_1 = []
    for i in range(len(list_1)):
        passenger_1.append(agent_1(list_1[i][0],list_1[i][1],list_1[i][2],list_1[
    i][3]))

    #Case2
    list_2 = copy.deepcopy(list_0)
    pri_max_2 = 4

    for i in range (len(list_2)):
        if (list_2[i][0] == 1) or (list_2[i][0] == 5):
            list_2[i].append(4)
        elif (list_2[i][0] == 2) or (list_2[i][0] == 4):
            list_2[i].append(3)
        elif (list_2[i][0] == 9):
            list_2[i].append(2)
        elif (list_2[i][0] == 6) or (list_2[i][0] == 8):
            list_2[i].append(1)

    list_late_2 = random.sample(list_2,round(N*RL))

    for i in range (len(list_late_2)):
        for j in range (len(list_2)):
            if list_late_2[i] == list_2[j]:
                list_2[j][3] = pri_max_2+1
                if (list_2[j][0] == 4) or (list_2[j][0] == 6):
                    for k in range (len(list_2)):
                        if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 5):
                            list_2[k][3] = pri_max_2+1
```

```
    class agent_2:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_2 = []
    for i in range(len(list_2)):
        passenger_2.append(agent_2(list_2[i][0],list_2[i][1],list_2[i][2],list_2[
    i][3]))

    #Case3
    list_3 = copy.deepcopy(list_0)
    pri_max_3 = 4

    for i in range (len(list_3)):
        if (list_3[i][0] == 1) | (list_3[i][0] == 5):
            list_3[i].append(4)
        if (list_3[i][0] == 9) & (1 <= list_3[i][1] <= 14):
            list_3[i].append(4)

        if (list_3[i][0] == 2) | (list_3[i][0] == 4):
            list_3[i].append(3)
        if (list_3[i][0] == 6) & (1 <= list_3[i][0] <= 13):
            list_3[i].append(3)

        if (list_3[i][0] == 8) & (1 <= list_3[i][0] <= 14):
            list_3[i].append(2)
        if (list_3[i][0] == 9) & (20 <= list_3[i][1] <= 40):
            list_3[i].append(2)

        if (list_3[i][0] == 6) & (20 <= list_3[i][0] <= 40):
            list_3[i].append(1)
        if (list_3[i][0] == 8) & (20 <= list_3[i][0] <= 40):
            list_3[i].append(1)

    list_late_3 = random.sample(list_3,round(N*RL))

    for i in range (len(list_late_3)):
        for j in range (len(list_3)):
            if list_late_3[i] == list_3[j]:
                list_3[j][3] = pri_max_3+1
                if (list_3[j][0] == 4) or (list_3[j][0] == 6):
                    for k in range (len(list_3)):
                        if (list_3[k][1] == list_3[j][1]) & (list_3[k][0] == 5):
                            list_3[k][3] = pri_max_3+1

    class agent_3:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_3 = []
    for i in range(len(list_3)):
        passenger_3.append(agent_3(list_3[i][0],list_3[i][1],list_3[i][2],list_3[
    i][3]))

```

```
    #Case4
    list_4 = copy.deepcopy(list_0)
    pri_max_4 = 4

    for i in range (len(list_4)):
        if (list_4[i][0] == 1) | (list_4[i][0] == 5):
            list_4[i].append(4)
        if (list_4[i][0] == 9) & (1 <= list_4[i][1] <= 14):
            list_4[i].append(2)
        if (list_4[i][0] == 2) | (list_4[i][0] == 4):
            list_4[i].append(3)
        if (list_4[i][0] == 6) & (1 <= list_4[i][0] <= 13):
            list_4[i].append(1)
        if (list_4[i][0] == 8) & (1 <= list_4[i][0] <= 14):
            list_4[i].append(1)
        if (list_4[i][0] == 9) & (20 <= list_4[i][1] <= 40):
            list_4[i].append(4)
        if (list_4[i][0] == 6) & (20 <= list_4[i][0] <= 40):
            list_4[i].append(3)
        if (list_4[i][0] == 8) & (20 <= list_4[i][0] <= 40):
            list_4[i].append(3)

    list_late_4 = random.sample(list_4,round(N*RL))

    for i in range (len(list_late_4)):
        for j in range (len(list_4)):
            if list_late_4[i] == list_4[j]:
                list_4[j][3] = pri_max_4+1
                if (list_4[j][0] == 4) or (list_4[j][0] == 6):
                    for k in range (len(list_4)):
                        if (list_4[k][1] == list_4[j][1]) & (list_4[k][0] == 5):
                            list_4[k][3] = pri_max_4+1

    class agent_4:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    passenger_4 = []
    for i in range(len(list_4)):
        passenger_4.append(agent_4(list_4[i][0],list_4[i][1],list_4[i][2],list_4[
    i][3]))

    class person:
        def __init__(self,char,seat,bag,pri):
            self.char = char
            self.seat = seat
            self.bag = bag
            self.pri = pri

    list_pass = [[person(0,0,0,0) for i in range(0,41)] for j in range(0,10)]

    class grid:
        def __init__(self,type,value,pass_char,pass_seat,pass_pri):
            self.type = type
            # -1 -> block
            # 0 -> cabin
            # 1 -> queue
```

```
            # 2 -> aisle
            # 3 -> seat
            self.value = value
            # 0 -> avaliable
            # 1 -> passenger
            self.pass_char = pass_char
            self.pass_seat = pass_seat
            self.pass_pri = pass_pri

    plane = [[grid(-1,0,0,0,0) for i in range(0,42)] for j in range(0,252)]

    for i in range(1,3):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(4,7):
        for j in range(1,14):
            plane[i][j].type = 3

    for i in range(8,10):
        for j in range(1,15):
            plane[i][j].type = 3

    for i in range(1,3):
        for j in range(20,41):
            plane[i][j].type = 3

    for i in range(4,7):
        for j in range(20,41):
            plane[i][j].type = 3

    for i in range(8,10):
        for j in range(20,41):
            plane[i][j].type = 3

    for i in range(3,252):
        plane[i][0].type = 1
        plane[i][41].type = 1

    q = [3,7]
    for i in q:
        for j in range(0,17):
            plane[i][j].type = 2
        for j in range(18,42):
            plane[i][j].type = 2

    for i in range(1,10):
        plane[i][17].type = 0

    def C1(passenger_1):
        for i in range(len(passenger_1)):
            list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
            list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
            list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
            list_pass[passenger_1[i].char][passenger_1[i].seat].pri = passenger_1
[i].pri
            plane[passenger_1[i].char][passenger_1[i].seat].value = 1
```

```
            plane[passenger_1[i].char][passenger_1[i].seat].pass_char = passenger_1[i].char
            plane[passenger_1[i].char][passenger_1[i].seat].pass_seat = passenger_1[i].seat
            plane[passenger_1[i].char][passenger_1[i].seat].pass_pri = passenger_1[i].pri
    def C2(passenger_2):
        for i in range(len(passenger_2)):
            list_pass[passenger_2[i].char][passenger_2[i].seat].char = passenger_2[i].char
            list_pass[passenger_2[i].char][passenger_2[i].seat].seat = passenger_2[i].seat
            list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2[i].bag
            list_pass[passenger_2[i].char][passenger_2[i].seat].pri = passenger_2[i].pri
            plane[passenger_2[i].char][passenger_2[i].seat].value = 1
            plane[passenger_2[i].char][passenger_2[i].seat].pass_char = passenger_2[i].char
            plane[passenger_2[i].char][passenger_2[i].seat].pass_seat = passenger_2[i].seat
            plane[passenger_2[i].char][passenger_2[i].seat].pass_pri = passenger_2[i].pri
    def C3(passenger_3):
        for i in range(len(passenger_3)):
            list_pass[passenger_3[i].char][passenger_3[i].seat].char = passenger_3[i].char
            list_pass[passenger_3[i].char][passenger_3[i].seat].seat = passenger_3[i].seat
            list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3[i].bag
            list_pass[passenger_3[i].char][passenger_3[i].seat].pri = passenger_3[i].pri
            plane[passenger_3[i].char][passenger_3[i].seat].value = 1
            plane[passenger_3[i].char][passenger_3[i].seat].pass_char = passenger_3[i].char
            plane[passenger_3[i].char][passenger_3[i].seat].pass_seat = passenger_3[i].seat
            plane[passenger_3[i].char][passenger_3[i].seat].pass_pri = passenger_3[i].pri
    def C4(passenger_4):
        for i in range(len(passenger_4)):
            list_pass[passenger_4[i].char][passenger_4[i].seat].char = passenger_4[i].char
            list_pass[passenger_4[i].char][passenger_4[i].seat].seat = passenger_4[i].seat
            list_pass[passenger_4[i].char][passenger_4[i].seat].bag = passenger_4[i].bag
            list_pass[passenger_4[i].char][passenger_4[i].seat].pri = passenger_4[i].pri
            plane[passenger_4[i].char][passenger_4[i].seat].value = 1
            plane[passenger_4[i].char][passenger_4[i].seat].pass_char = passenger_4[i].char
            plane[passenger_4[i].char][passenger_4[i].seat].pass_seat = passenger_4[i].seat
            plane[passenger_4[i].char][passenger_4[i].seat].pass_pri = passenger_4[i].pri

    if(case==1):
        C1(passenger_1)
```

```
    if(case==2):
        C1(passenger_2)
    if(case==3):
        C3(passenger_3)
    if(case==4):
        C4(passenger_4)

    time = 0

    a = [[0 for i in range(0,42)] for i in range(0,252)]

    for i in range(0,252):
        for j in range(0,42):
            #print(plane[i][j].value)
            a[i][j] = plane[i][j].value

    time = 0

    plt.figure('time'+str(time))
    im = plt.imshow(a[0:10])
    ax = plt.gca()
    ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
    ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
    ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
    plt.savefig("figure_3_d.png")
    #plt.show()

    def check_pass(plane):
        check = 0
        for w in range(0,10):
            for z in range(0,42):
                if(plane[w][z].type>0 and plane[w][z].value == 1):
                    check+=1
        return check

    def min_pri(plane):
        x = 10
        for w in range(0,10):
            for z in range(0,42):
                if(plane[w][z].type>0):
                    if(plane[w][z].value==1):
                        if(plane[w][z].pass_pri<x):
                            x = plane[w][z].pass_pri
        return x

    left_1 = [2,6]
    for i in left_1:
        for j in range(1,41):
            if(plane[i][j].value==0 and plane[i-1][j].value==1):
                p1r = plane[i-1][j].pass_char
                p1c = plane[i-1][j].pass_seat
                p1p = plane[i-1][j].pass_pri
                plane[i][j].pass_char = p1r
                plane[i][j].pass_seat = p1c
                plane[i][j].pass_pri = p1p
                plane[i][j].value = 1
                plane[i-1][j].pass_char = 0
                plane[i-1][j].pass_seat = 0
                plane[i-1][j].pass_pri = 0
                plane[i-1][j].value = 0
```

```
371
372     right_1 = [4,8]
373     for i in right_1:
374         for j in range(1,41):
375             if(plane[i][j].value==0 and plane[i+1][j].value==1):
376                 p1r = plane[i+1][j].pass_char
377                 p1c = plane[i+1][j].pass_seat
378                 p1p = plane[i+1][j].pass_pri
379                 plane[i][j].pass_char = p1r
380                 plane[i][j].pass_seat = p1c
381                 plane[i][j].pass_pri = p1p
382                 plane[i][j].value = 1
383                 plane[i+1][j].pass_char = 0
384                 plane[i+1][j].pass_seat = 0
385                 plane[i+1][j].pass_pri = 0
386                 plane[i+1][j].value = 0
387
388     time+=1
389
390     """for i in range(0,250):
391         for j in range(0,42):
392             #print(plane[i][j].value)
393             a[i][j] = plane[i][j].value
394     plt.figure('time'+str(time))
395     im = plt.imshow(a[0:10])
396     ax = plt.gca()
397     ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
398     ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
399     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
400     plt.savefig('time'+str(time)+'.png')"""
401
402     while(1):
403         x = min_pri(plane)
404         #check
405         check = check_pass(plane)
406         if(check==0):
407             break
408         time+=1
409
410         j = 0
411         for i in reversed(range(3,251)):
412             if(plane[i][j].value==1 and plane[i+1][j].value==0):
413                 p1r = plane[i][j].pass_char
414                 p1c = plane[i][j].pass_seat
415                 p1p = plane[i][j].pass_pri
416                 plane[i+1][j].pass_char = p1r
417                 plane[i+1][j].pass_seat = p1c
418                 plane[i+1][j].pass_pri = p1p
419                 plane[i+1][j].value = 1
420                 plane[i][j].pass_char = 0
421                 plane[i][j].pass_seat = 0
422                 plane[i][j].pass_pri = 0
423                 plane[i][j].value = 0
424
425         j = 41
426         for i in reversed(range(3,251)):
427             if(plane[i][j].value==1 and plane[i+1][j].value==0):
428                 p1r = plane[i][j].pass_char
429                 p1c = plane[i][j].pass_seat
430                 p1p = plane[i][j].pass_pri
```

```
                plane[i+1][j].pass_char = p1r
                plane[i+1][j].pass_seat = p1c
                plane[i+1][j].pass_pri = p1p
                plane[i+1][j].value = 1
                plane[i][j].pass_char = 0
                plane[i][j].pass_seat = 0
                plane[i][j].pass_pri = 0
                plane[i][j].value = 0

        aisle = [3,7]
        #front
        for i in aisle:
            for j in range(0,15):
                if(plane[i][j].value==1):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
                    continue
                if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
][plane[i][j+1].pass_seat].bag==0):
                    p1r = plane[i][j+1].pass_char
                    p1c = plane[i][j+1].pass_seat
                    p1p = plane[i][j+1].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i][j+1].pass_char = 0
                    plane[i][j+1].pass_seat = 0
                    plane[i][j+1].pass_pri = 0
                    plane[i][j+1].value = 0
                    continue
                if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
][plane[i][j+1].pass_seat].bag!=0):
                    continue
                if(j==0):
                    continue
                if(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1 and
plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                    ch = random.choice([i-1,i+1])
                    p1r = plane[ch][j].pass_char
                    p1c = plane[ch][j].pass_seat
                    p1p = plane[ch][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[ch][j].pass_char = 0
                    plane[ch][j].pass_seat = 0
                    plane[ch][j].pass_pri = 0
                    plane[ch][j].value = 0
                elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
                    p1r = plane[i-1][j].pass_char
                    p1c = plane[i-1][j].pass_seat
                    p1p = plane[i-1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
```

```
                    plane[i-1][j].pass_char = 0
                    plane[i-1][j].pass_seat = 0
                    plane[i-1][j].pass_pri = 0
                    plane[i-1][j].value = 0
                elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                    p1r = plane[i+1][j].pass_char
                    p1c = plane[i+1][j].pass_seat
                    p1p = plane[i+1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i+1][j].pass_char = 0
                    plane[i+1][j].pass_seat = 0
                    plane[i+1][j].pass_pri = 0
                    plane[i+1][j].value = 0
                else:
                    continue
        #back
        for i in aisle:
            for j in reversed(range(20,42)):
                if(plane[i][j].value==1):
                    if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
                        list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
                    continue
                if(plane[i][j-1].value == 1 and list_pass[plane[i][j-1].pass_char
][plane[i][j-1].pass_seat].bag==0):
                    p1r = plane[i][j-1].pass_char
                    p1c = plane[i][j-1].pass_seat
                    p1p = plane[i][j-1].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i][j-1].pass_char = 0
                    plane[i][j-1].pass_seat = 0
                    plane[i][j-1].pass_pri = 0
                    plane[i][j-1].value = 0
                    continue
                if(plane[i][j-1].value == 1 and list_pass[plane[i][j-1].pass_char
][plane[i][j-1].pass_seat].bag!=0):
                    continue
                if(j==41):
                    continue
                if(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1 and
plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                    ch = random.choice([i-1,i+1])
                    p1r = plane[ch][j].pass_char
                    p1c = plane[ch][j].pass_seat
                    p1p = plane[ch][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[ch][j].pass_char = 0
                    plane[ch][j].pass_seat = 0
                    plane[ch][j].pass_pri = 0
                    plane[ch][j].value = 0
```

```
                elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
                    p1r = plane[i-1][j].pass_char
                    p1c = plane[i-1][j].pass_seat
                    p1p = plane[i-1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i-1][j].pass_char = 0
                    plane[i-1][j].pass_seat = 0
                    plane[i-1][j].pass_pri = 0
                    plane[i-1][j].value = 0
                elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
                    p1r = plane[i+1][j].pass_char
                    p1c = plane[i+1][j].pass_seat
                    p1p = plane[i+1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i+1][j].pass_char = 0
                    plane[i+1][j].pass_seat = 0
                    plane[i+1][j].pass_pri = 0
                    plane[i+1][j].value = 0
                else:
                    continue
        left_1 = [2,6]
        for i in left_1:
            for j in range(1,41):
                if(plane[i][j].value==0 and plane[i-1][j].value==1):
                    p1r = plane[i-1][j].pass_char
                    p1c = plane[i-1][j].pass_seat
                    p1p = plane[i-1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i-1][j].pass_char = 0
                    plane[i-1][j].pass_seat = 0
                    plane[i-1][j].pass_pri = 0
                    plane[i-1][j].value = 0

        right_1 = [4,8]
        for i in right_1:
            for j in range(1,41):
                if(plane[i][j].value==0 and plane[i+1][j].value==1):
                    p1r = plane[i+1][j].pass_char
                    p1c = plane[i+1][j].pass_seat
                    p1p = plane[i+1][j].pass_pri
                    plane[i][j].pass_char = p1r
                    plane[i][j].pass_seat = p1c
                    plane[i][j].pass_pri = p1p
                    plane[i][j].value = 1
                    plane[i+1][j].pass_char = 0
                    plane[i+1][j].pass_seat = 0
                    plane[i+1][j].pass_pri = 0
                    plane[i+1][j].value = 0

        """print("time",time)
        print("check",check)
```

```
        print(x)
        for i in range(0,250):
            for j in range(0,42):
                #print(plane[i][j].value)
                a[i][j] = plane[i][j].value
        plt.figure('time'+str(time))
        im = plt.imshow(a[0:10])
        ax = plt.gca()
        ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
        ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
        ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
        plt.savefig('time'+str(time)+'.png')"""
    return time

print(run3(1,0,0,242))
print(run3(2,0,0,242))
print(run3(3,0,0,242))
print(run3(4,0,0,242))
```