

For office use only

Team Control Number

For office use only

T1 _____
T2 _____
T3 _____
T4 _____

2022031

F1 _____
F2 _____
F3 _____
F4 _____

2022

The International Mathematical Modeling Challenge (IM²C) 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	4
1.1 Background	4
1.2 Problem Restatement	4
2 Preliminaries	5
2.1 Definitions	5
2.2 General Assumptions	5
3 Model Construction	6
3.1 Parameters	6
3.2 Boarding Model	7
3.2.1 Summary of Boarding Model	7
3.2.2 Boarding Method and Ideal Boarding Sequence (IBS)	7
3.2.3 Boarding Method Complexity	9
3.2.4 Late-arriving passengers	9
3.2.5 Queue jumping and Real Boarding Sequence (RBS)	9
3.2.6 Aisle and Row Traverse Speed	10
3.2.7 Bag Stowing Time	10
3.2.8 Algorithm for Path Finding	11
3.2.9 Algorithm for Moving, Bag Stowing, and Unblocking	11
3.3 Disembarking Model	13
3.3.1 Summary of Disembarking Model	13
3.3.2 Disembarking Method	13
3.3.3 Late-disembarking Passengers	13
3.3.4 Bag Collection Time	13
3.3.5 Disembarking Simulation	13
4 Aircraft I: Narrow-body Aircraft	14
4.1 Boarding Methods	14
4.2 Optimal Boarding Method	15
4.3 Sensitivity Analysis	16
4.3.1 Variation of Bag Stowing Time	16
4.3.2 Variation of Percentage of Queue-jumping Passengers	17
4.4 Disembarking Methods	18
4.5 Optimal Disembarking Method	18
5 Aircraft II: Flying Wing Aircraft	19
5.1 Optimal Boarding Method	19
5.2 Optimal Disembarking Method	19
6 Aircraft III: Two-entrance Two-aisle Aircraft	20
6.1 Optimal Boarding Method	20
6.2 Optimal Disembarking Method	20
7 Adjustments for the Pandemic	21
7.1 Optimal Boarding Methods	21

8 Model Evaluation	22
8.1 Strengths	22
8.2 Limitations and Improvements	22
9 Conclusions	22
10 Letter to an Airline Executive	23
References	24
A Raw Data	26
A.1 Raw Data for Subsection 4.2	26
A.2 Raw Data for Subsection 5.1	27
A.3 Raw Data for Subsection 6.1	28
A.4 Raw Data for Subsection 7.1	29
B Codes	30
B.1 Codes of Boarding Process, Airplane I	30
B.2 Codes of Boarding Process, Airplane II	48
B.3 Codes of Boarding Process, Airplane III	63
B.4 Codes of Disembarking Process, Airplane I	90
B.5 Codes of Disembarking Process, Airplane II	101
B.6 Codes of Disembarking Process, Airplane III	112

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 $1, 2, \dots, M$. Each passenger is assigned a priority $p \in \{1, 2, \dots, M\}$. 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

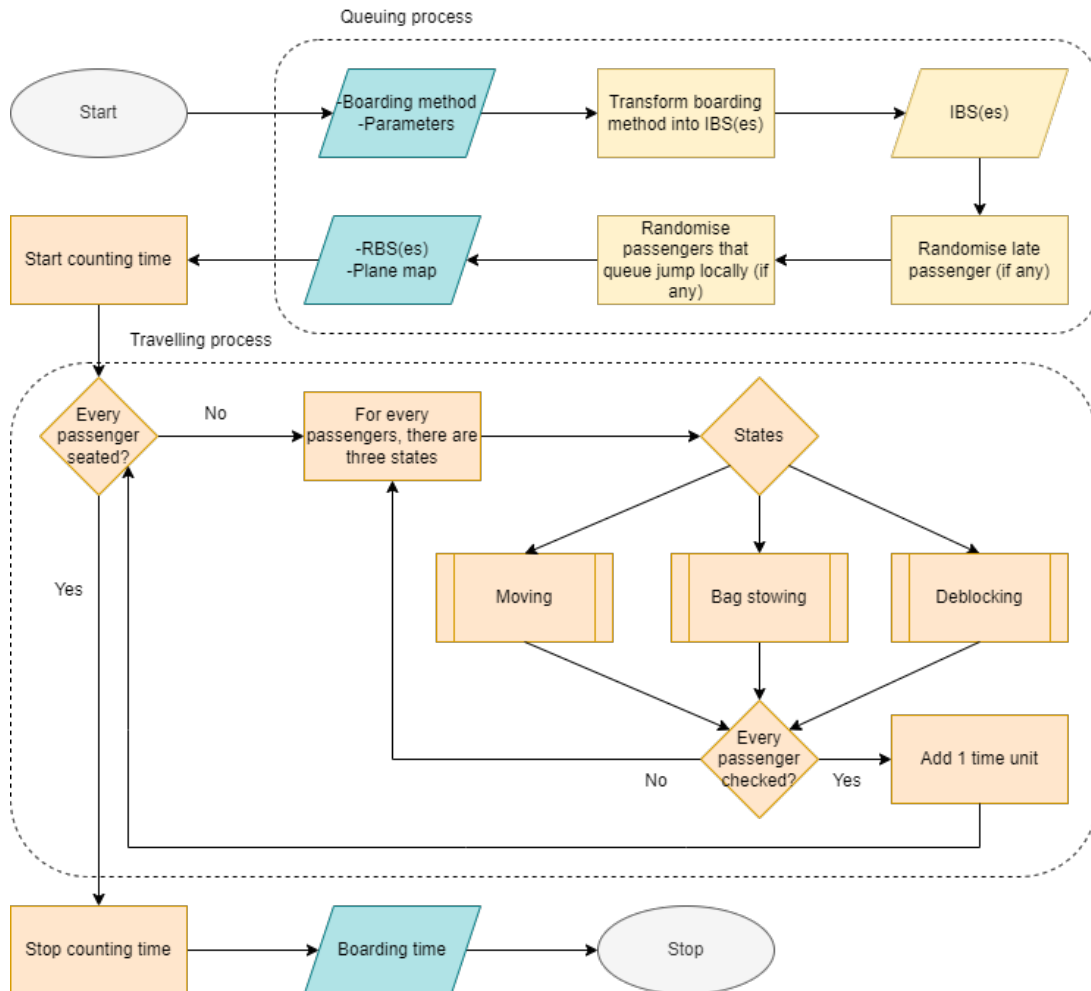


Figure 1: Flowchart of the boarding model

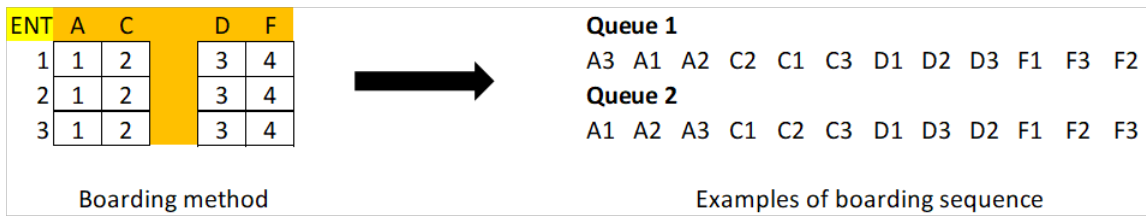


Figure 2: Turning a boarding method into boarding sequence.

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

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

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,

$$1 \text{ time unit} = 1.42 \text{ s}. \quad (4)$$

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

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

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]:

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

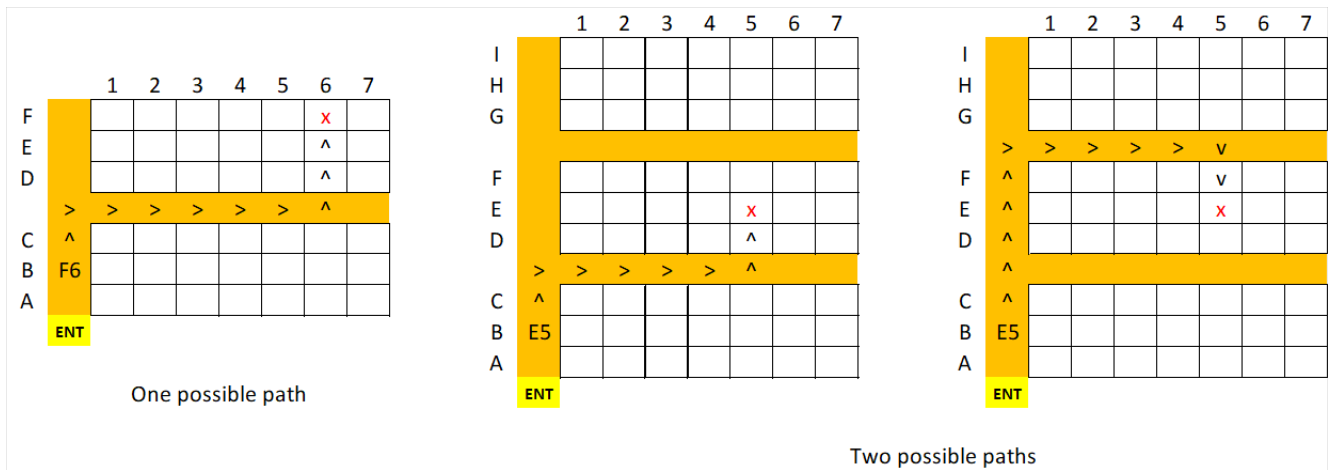


Figure 3: (Left) A seat with fixed simplest path, which almost always the case. (Middle & Right) A seat that has two simplest paths and the paths.

where $\Gamma(x)$ is the well-known gamma function $\Gamma(x) = \int_0^\infty e^{-u}u^{x-1}du$. 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.

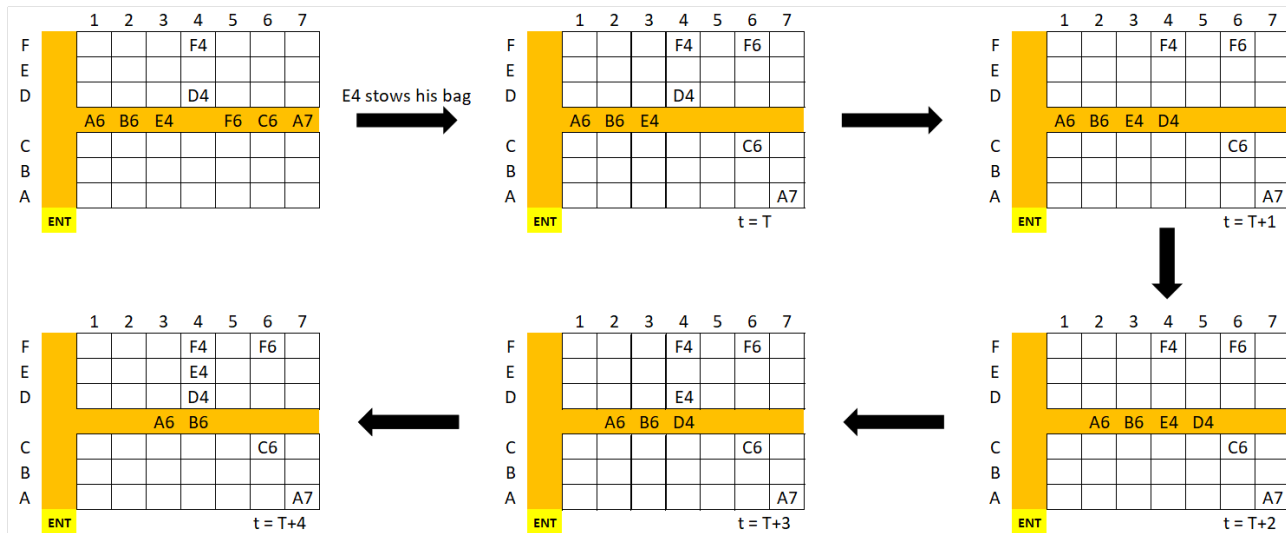


Figure 4: The unblocking process.

- **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.

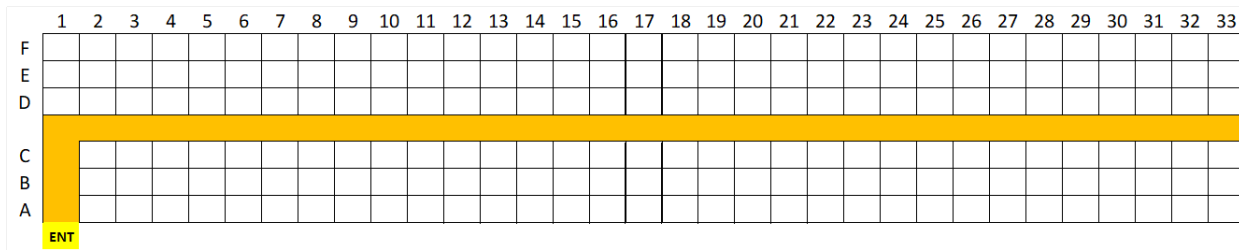


Figure 5: The grid map of Aircraft I.

- **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 i th group and j th 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 $C \approx 0.417$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	
F	17	49	16	48	15	47	14	46	13	45	12	44	11	43	10	42	9	41	8	40	7	39	6	38	5	37	4	36	3	35	2	34	1	
E	82		81		80		79		78		77		76		75		74		73		72		71		70		69		68		67		66	
D																																		
C																																		
B	...	98		97		96		95		94		93		92		91		90		89		88		87		86		85		84		83		
A	65	33	64	32	63	31	62	30	61	29	60	28	59	27	58	26	57	25	56	24	55	23	54	22	53	21	52	20	51	19	50	18		

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

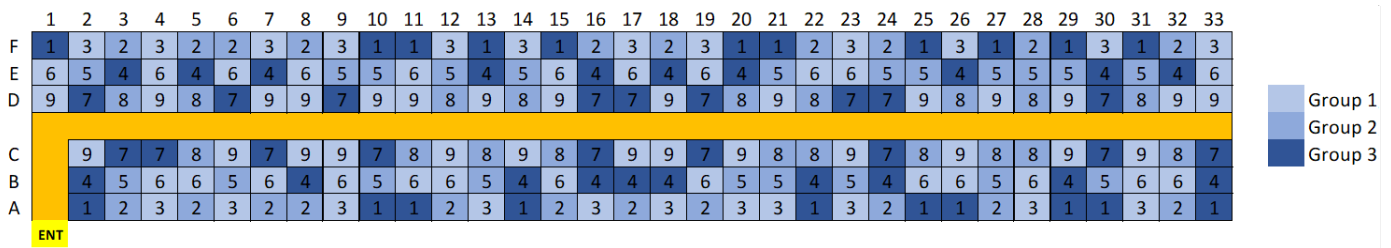


Figure 7: Boarding by luggage size method on Narrow-Body Passenger Aircraft

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

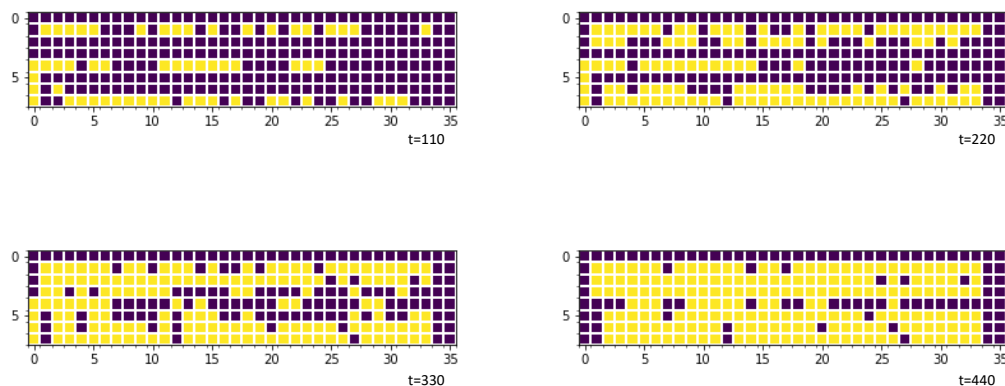


Figure 8: An example of the progression of a simulation using boarding by seat method. Note that $BT = 487$.

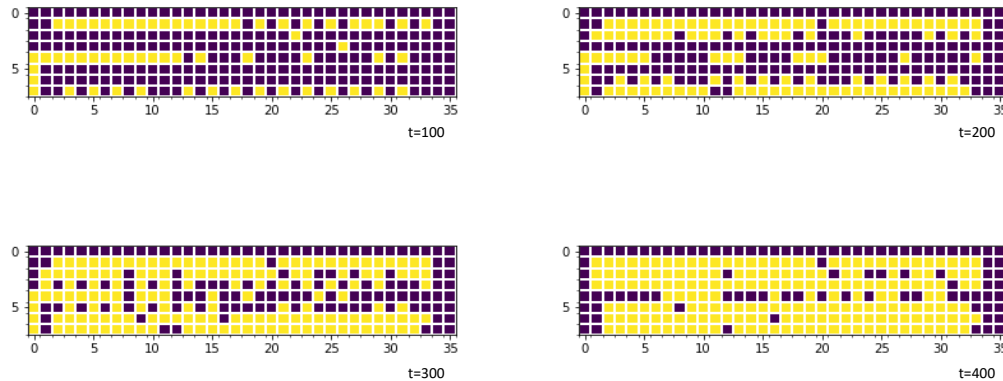


Figure 9: An example of the progression of a simulation using Steffen's method. Note that $BT = 436$.

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.

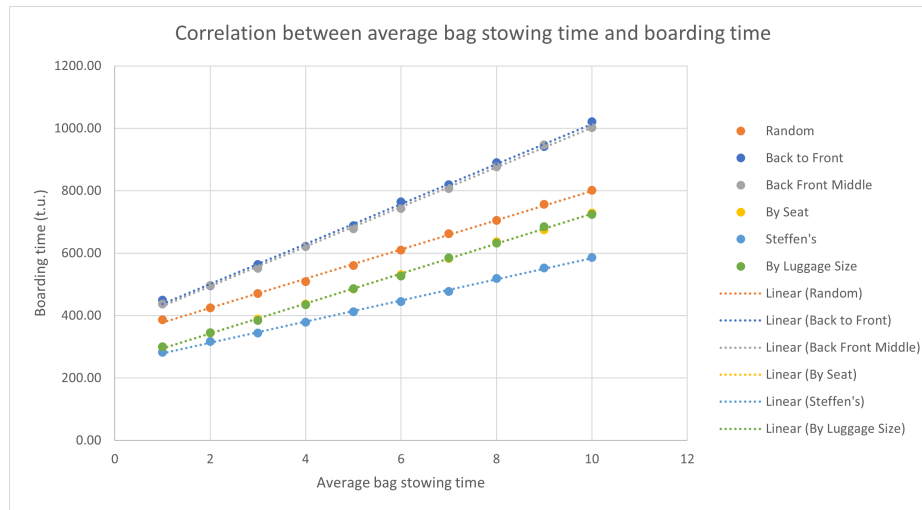


Figure 10: Graph between average bag stowing time and boarding time, along with the best-fit linear lines.

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.

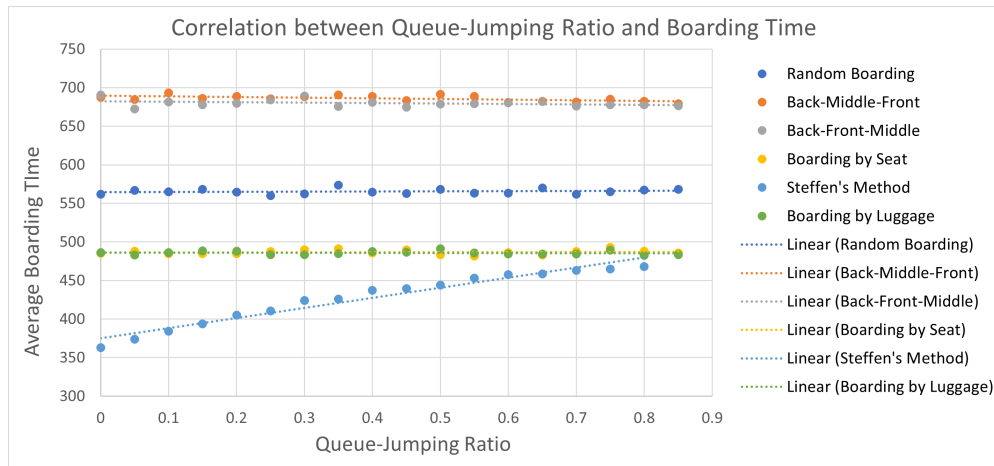


Figure 11: Graph between Queue-Jumping Ratio R_J and average boarding time .

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]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
F	3	3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
E	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C																																	
B		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
A		2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
		3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
ENT																																	

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).

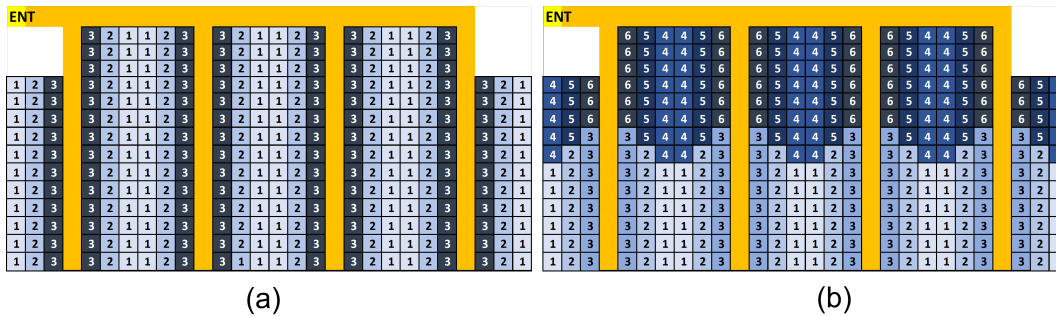


Figure 13: (a) Boarding by Seat ($C = 0.191$) and (b) Modified-Boarding by Seat ($C = 0.311$)

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

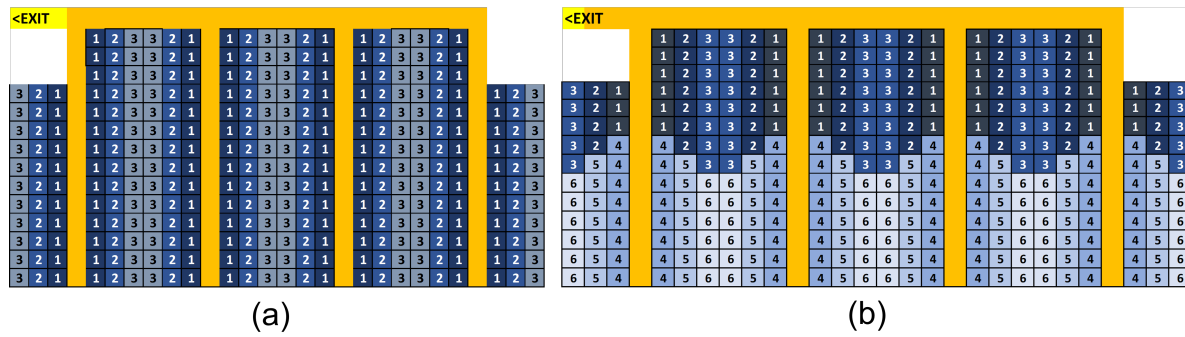


Figure 14: (a) Disembarking by Seat (b) Modified-Disembarking by Seat

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.

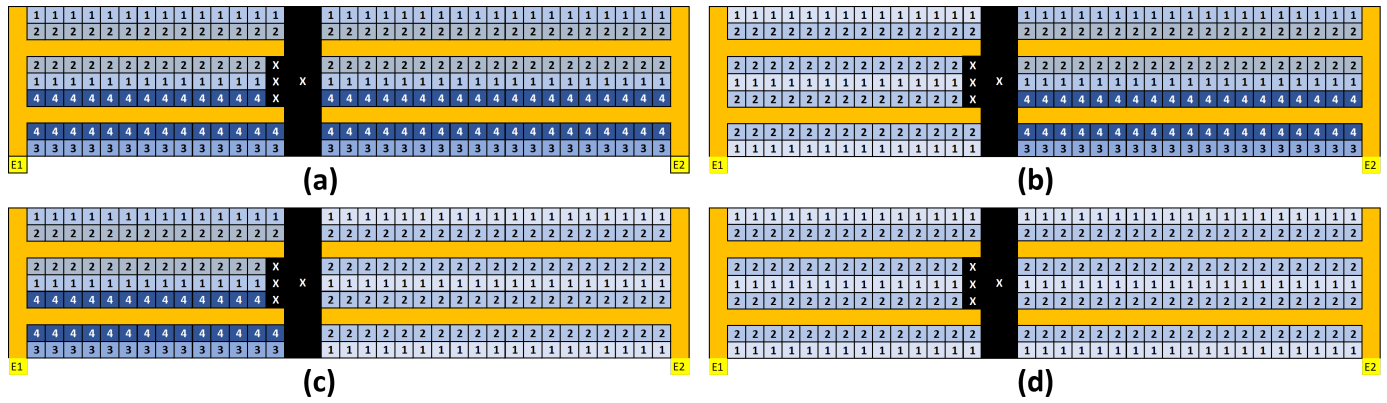


Figure 15: Four possible methods of boarding by seat, $C = 0.255$ for (a), $C = 0.330$ for (b) and (c) and $C = 0.255$ for (d).

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.

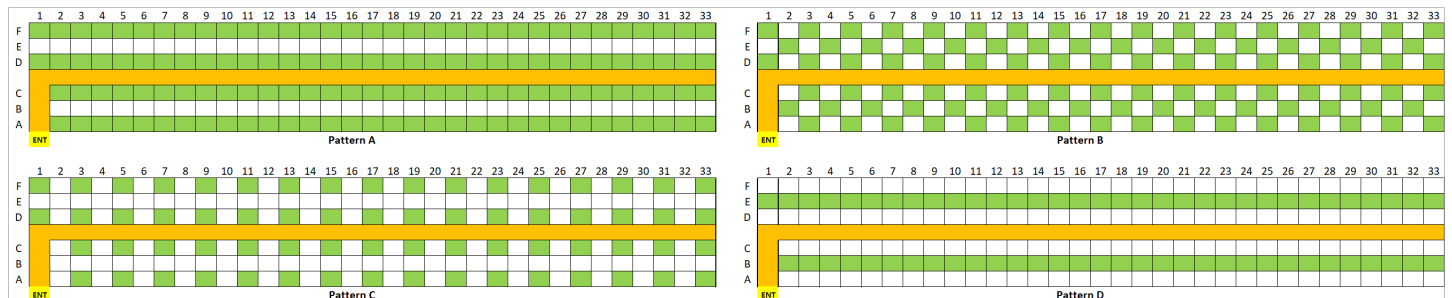


Figure 16: Four patterns of seating in Aircraft I in the pandemic.

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

```

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
10 def run1(case,RL,RJ,N):
11     # Create position of people in "Narrow Body" Passenger Aircraft
12     arr = np.arange(1,196)
13     list_0 = arr.tolist()
14     for i in range (99):
15         list_0[i] = [math.floor(list_0[i]/33)+1,list_0[i]%33]
16     for i in range (99,195):
17         list_0[i] = [math.floor((list_0[i]-99)/32)+5,(list_0[i]-99)%32+1]
18     list_0[32] = [1,33]
19     list_0[65] = [2,33]
20     list_0[98] = [3,33]
21     list_0[130] = [5,33]
22     list_0[162] = [6,33]
23     list_0[194] = [7,33]
24
25     # Random luggage stow time of each people by experimental data and Weibull
26     # distribution
27     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,
28     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,
29     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,
30     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,
31     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,
32     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]
33     luggage_arr = np.array(luggage)
34     mean = luggage_arr.mean()
35     std = stdev(luggage_arr)
36     k = (std/mean)**(-1.086)
37     z = 1 + 1/k
38     def f(x):
39         return math.exp(-x)*(x**(z-1))
40     gamma,err = quad(f, 0, math.inf)
41     c = mean/gamma
42     for i in range (len(list_0)):
43         weibull = (c*(np.random.weibull(k, 1))).tolist()
44         time = round((weibull[0]/1.42))
45         list_0[i].append(time)
46
47     #Case1
48     list_1 = random.sample(list_0,195)
49
50     class agent_1:
51         def __init__(self,char,seat,bag):
52             self.char = char
53             self.seat = seat
54             self.bag = bag

```

```

50 passenger_1 = []
51 for i in range(len(list_0)):
52     passenger_1.append(agent_1(list_1[i][0], list_1[i][1], list_1[i][2]))
53
54 #Case2
55 list_late_2 = random.sample(list_0, round(N*RL))
56 list_no_late_2 = [x for x in list_0 if x not in list_late_2]
57
58 list_bow = []
59 list_middle_2 = []
60 list_aft = []
61
62 for i in range (len(list_no_late_2)):
63     if 1 <= list_no_late_2[i][1] <= 11:
64         list_bow.append(list_no_late_2[i])
65
66 for i in range (len(list_no_late_2)):
67     if 12 <= list_no_late_2[i][1] <= 22:
68         list_middle_2.append(list_no_late_2[i])
69
70 for i in range (len(list_no_late_2)):
71     if 23 <= list_no_late_2[i][1] <= 33:
72         list_aft.append(list_no_late_2[i])
73
74 list_random_bow = random.sample(list_bow, len(list_bow))
75 list_random_middle_2 = random.sample(list_middle_2, len(list_middle_2))
76 list_random_aft = random.sample(list_aft, len(list_aft))
77 list_random_2 = list_random_aft + list_random_middle_2 + list_random_bow
78
79 list_bad_2 = random.sample(list_random_2, round(N*RJ))
80
81 random_2 = []
82 for i in range (len(list_random_2)):
83     for j in range (len(list_bad_2)):
84         if list_random_2[i] == list_bad_2[j]:
85             random_2.append(i)
86
87 list_bad_2 = []
88 for i in range (len(random_2)):
89     list_bad_2.append(list_random_2[random_2[i]])
90
91 list_no_bad_2 = [x for x in list_random_2 if x not in list_bad_2]
92
93 no_random_2 = []
94 for i in range (len(list_random_2)):
95     for j in range (len(list_no_bad_2)):
96         if list_random_2[i] == list_no_bad_2[j]:
97             no_random_2.append(i)
98
99 normal_2 = []
100 for i in range (len(random_2)):
101     normal_2.append(np.random.normal(random_2[i], 5))
102
103 for i in range (len(normal_2)):
104     list_bad_2[i].append(normal_2[i])
105
106 list_bad_2 = sorted(list_bad_2, key=lambda x: x[-1])
107
108 for i in range (len(normal_2)):
109     del list_bad_2[i][3]

```

```

110     normal_2 = sorted(normal_2)
111
112     list_shift_2 = []
113     j=0
114     for i in range (len(list_no_bad_2)+len(list_bad_2)):
115         if (len(list_bad_2) != 0) & (j < len(no_random_2)):
116             if normal_2[0] < no_random_2[j]:
117                 list_shift_2.append(list_bad_2[0])
118                 del list_bad_2[0]
119                 del normal_2[0]
120             elif normal_2[0] > no_random_2[j]:
121                 list_shift_2.append(list_no_bad_2[j])
122                 j=j+1
123             elif j >= len(no_random_2):
124                 list_shift_2.append(list_bad_2[0])
125                 del list_bad_2[0]
126                 del normal_2[0]
127             else:
128                 list_shift_2.append(list_no_bad_2[j])
129                 j=j+1
130
131     list_2= list_shift_2 + list_late_2
132
133     class agent_2:
134         def __init__(self, char, seat, bag, num_out, row, col):
135             self.char = char
136             self.seat = seat
137             self.bag = bag
138             self.num_out = num_out
139             self.row = row
140             self.col = col
141
142     passenger_2 = []
143     for i in range(len(list_0)):
144         passenger_2.append(agent_2(list_2[i][0], list_2[i][1], list_2[i][2], 0, 0, 0))
145
146     #Case3
147     list_late_3 = random.sample(list_0, round(N*RL))
148     list_no_late_3 = [x for x in list_0 if x not in list_late_3]
149
150     list_bow = []
151     list_middle_3 = []
152     list_aft = []
153
154     for i in range (len(list_no_late_3)):
155         if 1 <= list_no_late_3[i][1] <= 11:
156             list_bow.append(list_no_late_3[i])
157
158     for i in range (len(list_no_late_3)):
159         if 12 <= list_no_late_3[i][1] <= 22:
160             list_middle_3.append(list_no_late_3[i])
161
162     for i in range (len(list_no_late_3)):
163         if 23 <= list_no_late_3[i][1] <= 33:
164             list_aft.append(list_no_late_3[i])
165
166     list_random_bow = random.sample(list_bow, len(list_bow))
167     list_random_middle_3 = random.sample(list_middle_3, len(list_middle_3))
168     list_random_aft = random.sample(list_aft, len(list_aft))
169

```



```
170 list_random_3 = list_random_aft + list_random_bow + list_random_middle_3
171
172 list_bad_3 = random.sample(list_random_3, round(N*RJ))
173
174 random_3 = []
175 for i in range (len(list_random_3)):
176     for j in range (len(list_bad_3)):
177         if list_random_3[i] == list_bad_3[j]:
178             random_3.append(i)
179
180 list_bad_3 = []
181 for i in range (len(random_3)):
182     list_bad_3.append(list_random_3[random_3[i]])
183
184 list_no_bad_3 = [x for x in list_random_3 if x not in list_bad_3]
185
186 no_random_3 = []
187 for i in range (len(list_random_3)):
188     for j in range (len(list_no_bad_3)):
189         if list_random_3[i] == list_no_bad_3[j]:
190             no_random_3.append(i)
191
192 normal_3 = []
193 for i in range (len(random_3)):
194     normal_3.append(np.random.normal(random_3[i], 5))
195
196 for i in range (len(normal_3)):
197     list_bad_3[i].append(normal_3[i])
198
199 list_bad_3 = sorted(list_bad_3, key=lambda x: x[-1])
200
201 for i in range (len(normal_3)):
202     del list_bad_3[i][3]
203
204 normal_3 = sorted(normal_3)
205
206 list_shift_3 = []
207 j=0
208 for i in range (len(list_no_bad_3)+len(list_bad_3)):
209     if (len(list_bad_3) != 0) & (j < len(no_random_3)):
210         if normal_3[0] < no_random_3[j]:
211             list_shift_3.append(list_bad_3[0])
212             del list_bad_3[0]
213             del normal_3[0]
214         elif normal_3[0] > no_random_3[j]:
215             list_shift_3.append(list_no_bad_3[j])
216             j=j+1
217         elif j >= len(no_random_3):
218             list_shift_3.append(list_bad_3[0])
219             del list_bad_3[0]
220             del normal_3[0]
221         else:
222             list_shift_3.append(list_no_bad_3[j])
223             j=j+1
224
225 list_3 = list_shift_3 + list_late_3
226
227 class agent_3:
228     def __init__(self, char, seat, bag, num_out, row, col):
229         self.char = char
```

```
230         self.seat = seat
231         self.bag = bag
232         self.num_out = num_out
233         self.row = row
234         self.col = col
235
236     passenger_3 = []
237     for i in range(len(list_0)):
238         passenger_3.append(agent_3(list_3[i][0], list_3[i][1], list_3[i][2], 0, 0, 0))
239
240     #Case4
241     list_late_4 = random.sample(list_0, round(N*RL))
242     list_no_late_4 = [x for x in list_0 if x not in list_late_4]
243
244     list_window_4 = []
245     list_middle_4 = []
246     list_aisle_4 = []
247
248     for i in range(len(list_no_late_4)):
249         if (list_no_late_4[i][0] == 1) | (list_no_late_4[i][0] == 7):
250             list_window_4.append(list_no_late_4[i])
251     for i in range(len(list_no_late_4)):
252         if (list_no_late_4[i][0] == 2) | (list_no_late_4[i][0] == 6):
253             list_middle_4.append(list_no_late_4[i])
254
255     for i in range(len(list_no_late_4)):
256         if (list_no_late_4[i][0] == 3) | (list_no_late_4[i][0] == 5):
257             list_aisle_4.append(list_no_late_4[i])
258
259     list_random_window_4 = random.sample(list_window_4, len(list_window_4))
260     list_random_middle_4 = random.sample(list_middle_4, len(list_middle_4))
261     list_random_aisle_4 = random.sample(list_aisle_4, len(list_aisle_4))
262     list_random_4 = list_random_window_4 + list_random_middle_4 +
list_random_aisle_4
263
264     list_bad_4 = random.sample(list_random_4, round(N*RJ))
265
266     random_4 = []
267     for i in range(len(list_random_4)):
268         for j in range(len(list_bad_4)):
269             if list_random_4[i] == list_bad_4[j]:
270                 random_4.append(i)
271
272     list_bad_4 = []
273     for i in range(len(random_4)):
274         list_bad_4.append(list_random_4[random_4[i]])
275
276     list_no_bad_4 = [x for x in list_random_4 if x not in list_bad_4]
277
278     no_random_4 = []
279     for i in range(len(list_random_4)):
280         for j in range(len(list_no_bad_4)):
281             if list_random_4[i] == list_no_bad_4[j]:
282                 no_random_4.append(i)
283
284     normal_4 = []
285     for i in range(len(random_4)):
286         normal_4.append(np.random.normal(random_4[i], 5))
287
288     for i in range(len(normal_4)):
```

```

289         list_bad_4[i].append(normal_4[i])
290
291 list_bad_4 = sorted(list_bad_4, key=lambda x: x[-1])
292
293 for i in range (len(normal_4)):
294     del list_bad_4[i][3]
295
296 normal_4 = sorted(normal_4)
297
298 list_shift_4 = []
299 j=0
300 for i in range (len(list_no_bad_4)+len(list_bad_4)):
301     if (len(list_bad_4) != 0) & (j < len(no_random_4)):
302         if normal_4[0] < no_random_4[j]:
303             list_shift_4.append(list_bad_4[0])
304             del list_bad_4[0]
305             del normal_4[0]
306         elif normal_4[0] > no_random_4[j]:
307             list_shift_4.append(list_no_bad_4[j])
308             j=j+1
309         elif j >= len(no_random_4):
310             list_shift_4.append(list_bad_4[0])
311             del list_bad_4[0]
312             del normal_4[0]
313         else:
314             list_shift_4.append(list_no_bad_4[j])
315             j=j+1
316
317 list_4 = list_shift_4 + list_late_4
318
319 class agent_4:
320     def __init__(self, char, seat, bag, num_out, row, col):
321         self.char = char
322         self.seat = seat
323         self.bag = bag
324         self.num_out = num_out
325         self.row = row
326         self.col = col
327
328 passenger_4 = []
329 for i in range(len(list_0)):
330     passenger_4.append(agent_4(list_4[i][0], list_4[i][1], list_4[i][2], 0, 0, 0))
331
332 #Case5
333 arr_1_ABC = np.arange(33, 0, -2)
334 arr_1_DEF = np.arange(33, 1, -2)
335 arr_2_all = np.arange(32, 0, -2)
336 list_1_ABC = arr_1_ABC.tolist()
337 list_1_DEF = arr_1_DEF.tolist()
338 list_2_all = arr_2_all.tolist()
339
340 list_correct_5 = []
341
342 for i in range (len(list_1_ABC)):
343     for j in range (len(list_0)):
344         if (list_0[j][0] == 1) & (list_0[j][1] == list_1_ABC[i]):
345             list_correct_5.append(list_0[j])
346 for i in range (len(list_1_DEF)):
347     for j in range (len(list_0)):
348         if (list_0[j][0] == 7) & (list_0[j][1] == list_1_DEF[i]):

```

```
349         list_correct_5.append(list_0[j])
350     for i in range (len(list_2_all)):
351         for j in range (len(list_0)):
352             if (list_0[j][0] == 1) & (list_0[j][1] == list_2_all[i]):
353                 list_correct_5.append(list_0[j])
354     for i in range (len(list_2_all)):
355         for j in range (len(list_0)):
356             if (list_0[j][0] == 7) & (list_0[j][1] == list_2_all[i]):
357                 list_correct_5.append(list_0[j])
358     for i in range (len(list_1_ABC)):
359         for j in range (len(list_0)):
360             if (list_0[j][0] == 2) & (list_0[j][1] == list_1_ABC[i]):
361                 list_correct_5.append(list_0[j])
362     for i in range (len(list_1_DEF)):
363         for j in range (len(list_0)):
364             if (list_0[j][0] == 6) & (list_0[j][1] == list_1_DEF[i]):
365                 list_correct_5.append(list_0[j])
366     for i in range (len(list_2_all)):
367         for j in range (len(list_0)):
368             if (list_0[j][0] == 2) & (list_0[j][1] == list_2_all[i]):
369                 list_correct_5.append(list_0[j])
370     for i in range (len(list_2_all)):
371         for j in range (len(list_0)):
372             if (list_0[j][0] == 6) & (list_0[j][1] == list_2_all[i]):
373                 list_correct_5.append(list_0[j])
374     for i in range (len(list_1_ABC)):
375         for j in range (len(list_0)):
376             if (list_0[j][0] == 3) & (list_0[j][1] == list_1_ABC[i]):
377                 list_correct_5.append(list_0[j])
378     for i in range (len(list_1_DEF)):
379         for j in range (len(list_0)):
380             if (list_0[j][0] == 5) & (list_0[j][1] == list_1_DEF[i]):
381                 list_correct_5.append(list_0[j])
382     for i in range (len(list_2_all)):
383         for j in range (len(list_0)):
384             if (list_0[j][0] == 3) & (list_0[j][1] == list_2_all[i]):
385                 list_correct_5.append(list_0[j])
386     for i in range (len(list_2_all)):
387         for j in range (len(list_0)):
388             if (list_0[j][0] == 5) & (list_0[j][1] == list_2_all[i]):
389                 list_correct_5.append(list_0[j])
390
391     list_late_5 = random.sample(list_correct_5, round(N*RL))
392     list_no_late_5 = [x for x in list_correct_5 if x not in list_late_5]
393
394     list_bad_5 = random.sample(list_no_late_5, round(N*RJ))
395
396     random_5 = []
397     for i in range (len(list_no_late_5)):
398         for j in range (len(list_bad_5)):
399             if list_no_late_5[i] == list_bad_5[j]:
400                 random_5.append(i)
401
402     list_bad_5 = []
403     for i in range (len(random_5)):
404         list_bad_5.append(list_no_late_5[random_5[i]])
405
406     list_no_bad_5 = [x for x in list_no_late_5 if x not in list_bad_5]
407
408     no_random_5 = []
```

```

409     for i in range (len(list_no_late_5)):
410         for j in range (len(list_no_bad_5)):
411             if list_no_late_5[i] == list_no_bad_5[j]:
412                 no_random_5.append(i)
413
414     normal_5 = []
415     for i in range (len(random_5)):
416         normal_5.append(np.random.normal(random_5[i], 5))
417
418     for i in range (len(normal_5)):
419         list_bad_5[i].append(normal_5[i])
420
421     list_bad_5 = sorted(list_bad_5, key=lambda x: x[-1])
422
423     for i in range (len(normal_5)):
424         del list_bad_5[i][3]
425
426     normal_5 = sorted(normal_5)
427
428     list_shift_5 = []
429     j=0
430     for i in range (len(list_no_bad_5)+len(list_bad_5)):
431         if (len(list_bad_5) != 0) & (j < len(no_random_5)):
432             if normal_5[0] < no_random_5[j]:
433                 list_shift_5.append(list_bad_5[0])
434                 del list_bad_5[0]
435                 del normal_5[0]
436             elif normal_5[0] > no_random_5[j]:
437                 list_shift_5.append(list_no_bad_5[j])
438                 j=j+1
439             elif j >= len(no_random_5):
440                 list_shift_5.append(list_bad_5[0])
441                 del list_bad_5[0]
442                 del normal_5[0]
443             else:
444                 list_shift_5.append(list_no_bad_5[j])
445                 j=j+1
446
447     list_5 = list_shift_5 + list_late_5
448
449     class agent_5:
450         def __init__(self, char, seat, bag, num_out, row, col):
451             self.char = char
452             self.seat = seat
453             self.bag = bag
454             self.num_out = num_out
455             self.row = row
456             self.col = col
457
458     passenger_5 = []
459     for i in range(len(list_0)):
460         passenger_5.append(agent_5(list_5[i][0], list_5[i][1], list_5[i][2], 0, 0, 0))
461
462     #Case6
463     list_late_6 = random.sample(list_0, round(N*RL))
464     list_no_late_6 = [x for x in list_0 if x not in list_late_6]
465
466     list_window_6 = []
467     list_middle_6 = []
468     list_aisle_6 = []

```

```
469 list_window_6_0 = []
470 list_window_6_4 = []
471 list_window_6_7 = []
472 list_middle_6_0 = []
473 list_middle_6_4 = []
474 list_middle_6_7 = []
475 list_aisle_6_0 = []
476 list_aisle_6_4 = []
477 list_aisle_6_7 = []
478
479 for i in range (len(list_no_late_6)):
480     if (list_no_late_6[i][0] == 1) | (list_no_late_6[i][0] == 7):
481         list_window_6.append(list_no_late_6[i])
482
483 for i in range (len(list_no_late_6)):
484     if (list_no_late_6[i][0] == 2) | (list_no_late_6[i][0] == 6):
485         list_middle_6.append(list_no_late_6[i])
486
487 for i in range (len(list_no_late_6)):
488     if (list_no_late_6[i][0] == 3) | (list_no_late_6[i][0] == 5):
489         list_aisle_6.append(list_no_late_6[i])
490
491 for i in range (len(list_window_6)):
492     if 0 <= list_window_6[i][2] <= 3:
493         list_window_6_0.append(list_window_6[i])
494
495 for i in range (len(list_window_6)):
496     if 4 <= list_window_6[i][2] <= 6:
497         list_window_6_4.append(list_window_6[i])
498
499 for i in range (len(list_window_6)):
500     if 7 <= list_window_6[i][2]:
501         list_window_6_7.append(list_window_6[i])
502
503 for i in range (len(list_middle_6)):
504     if 0 <= list_middle_6[i][2] <=3:
505         list_middle_6_0.append(list_middle_6[i])
506
507 for i in range (len(list_middle_6)):
508     if 4 <= list_middle_6[i][2] <=6:
509         list_middle_6_4.append(list_middle_6[i])
510
511 for i in range (len(list_middle_6)):
512     if 7 <= list_middle_6[i][2]:
513         list_middle_6_7.append(list_middle_6[i])
514
515 for i in range (len(list_aisle_6)):
516     if 0 <= list_aisle_6[i][2] <=3:
517         list_aisle_6_0.append(list_aisle_6[i])
518
519 for i in range (len(list_aisle_6)):
520     if 4 <= list_aisle_6[i][2] <=6:
521         list_aisle_6_4.append(list_aisle_6[i])
522
523 for i in range (len(list_aisle_6)):
524     if 7 <= list_aisle_6[i][2]:
525         list_aisle_6_7.append(list_aisle_6[i])
526
527 list_random_window_6_0 = random.sample(list_window_6_0, len(list_window_6_0))
528 list_random_window_6_4 = random.sample(list_window_6_4, len(list_window_6_4))
```

```

529 list_random_window_6_7 = random.sample(list_window_6_7, len(list_window_6_7))
530 list_random_middle_6_0 = random.sample(list_middle_6_0, len(list_middle_6_0))
531 list_random_middle_6_4 = random.sample(list_middle_6_4, len(list_middle_6_4))
532 list_random_middle_6_7 = random.sample(list_middle_6_7, len(list_middle_6_7))
533 list_random_aisle_6_0 = random.sample(list_aisle_6_0, len(list_aisle_6_0))
534 list_random_aisle_6_4 = random.sample(list_aisle_6_4, len(list_aisle_6_4))
535 list_random_aisle_6_7 = random.sample(list_aisle_6_7, len(list_aisle_6_7))
536 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

537
538 list_bad_6 = random.sample(list_random_6, round(N*RJ))
539
540 random_6 = []
541 for i in range (len(list_random_6)):
542     for j in range (len(list_bad_6)):
543         if list_random_6[i] == list_bad_6[j]:
544             random_6.append(i)
545
546 list_bad_6 = []
547 for i in range (len(random_6)):
548     list_bad_6.append(list_random_6[random_6[i]])
549
550 list_no_bad_6 = [x for x in list_random_6 if x not in list_bad_6]
551
552 no_random_6 = []
553 for i in range (len(list_random_6)):
554     for j in range (len(list_no_bad_6)):
555         if list_random_6[i] == list_no_bad_6[j]:
556             no_random_6.append(i)
557
558 normal_6 = []
559 for i in range (len(random_6)):
560     normal_6.append(np.random.normal(random_6[i], 5))
561
562 for i in range (len(normal_6)):
563     list_bad_6[i].append(normal_6[i])
564
565 list_bad_6 = sorted(list_bad_6, key=lambda x: x[-1])
566
567 for i in range (len(normal_6)):
568     del list_bad_6[i][3]
569
570 normal_6 = sorted(normal_6)
571
572 list_shift_6 = []
573 j=0
574 for i in range (len(list_no_bad_6)+len(list_bad_6)):
575     if (len(list_bad_6) != 0) & (j < len(no_random_6)):
576         if normal_6[0] < no_random_6[j]:
577             list_shift_6.append(list_bad_6[0])
578             del list_bad_6[0]
579             del normal_6[0]
580         elif normal_6[0] > no_random_6[j]:
581             list_shift_6.append(list_no_bad_6[j])
582             j=j+1
583         elif j >= len(no_random_6):
584             list_shift_6.append(list_bad_6[0])
585             del list_bad_6[0]

```

```
586         del normal_6[0]
587     else:
588         list_shift_6.append(list_no_bad_6[j])
589         j=j+1
590
591 list_6 = list_shift_6 + list_late_6
592
593 class agent_6:
594     def __init__(self, char, seat, bag, num_out, row, col):
595         self.char = char
596         self.seat = seat
597         self.bag = bag
598         self.num_out = num_out
599         self.row = row
600         self.col = col
601
602 passenger_6 = []
603 for i in range(len(list_0)):
604     passenger_6.append(agent_6(list_6[i][0], list_6[i][1], list_6[i][2], 0, 0, 0))
605
606 class person:
607     def __init__(self, char, seat, bag, num_out, t_1, t_2, check):
608         self.char = char
609         self.seat = seat
610         self.bag = bag
611         self.num_out = num_out
612         self.t_1 = t_1
613         self.t_2 = t_2
614         self.check = check
615
616 list_pass = [[person(0,0,0,0,0,0,0) for i in range(0,34)] for j in range(0,8)
617 ]
618
619 class grid:
620     def __init__(self, type, value, pass_char, pass_seat):
621         self.type = type
622         # 0 -> block
623         # 1 -> queue
624         # 2 -> aisle
625         # 3 -> seat
626         self.value = value
627         # 0 -> available
628         # 1 -> passenger
629         self.pass_char = pass_char
630         self.pass_seat = pass_seat
631
632 plane = [[grid(0,0,0,0) for i in range(0,36)] for i in range(0,200)]
633
634 for i in range(1,8):
635     for j in range(1,34):
636         plane[i][j].type = 3
637
638 for i in range(5,8):
639     plane[i][1].type = 0
640
641 for i in range(0,36):
642     plane[4][i].type = 2
643
644 for i in range(5,200):
645     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
```

```
693     list_pass[passenger_4[i].char][passenger_4[i].seat].num_out = -1
694     list_pass[passenger_4[i].char][passenger_4[i].seat].t_1 = -1
695     list_pass[passenger_4[i].char][passenger_4[i].seat].t_2 = -1
696     list_pass[passenger_4[i].char][passenger_4[i].seat].check = 0
697     for i in range(0, len(passenger_4)):
698         plane[5+i][0].value = 1
699         plane[5+i][0].pass_char = passenger_4[i].char
700         plane[5+i][0].pass_seat = passenger_4[i].seat
701
702     def C5(passenger_5):
703         for i in range(len(passenger_5)):
704             list_pass[passenger_5[i].char][passenger_5[i].seat].char =
passenger_5[i].char
705             list_pass[passenger_5[i].char][passenger_5[i].seat].seat =
passenger_5[i].seat
706             list_pass[passenger_5[i].char][passenger_5[i].seat].bag = passenger_5
[i].bag
707             list_pass[passenger_5[i].char][passenger_5[i].seat].num_out = -1
708             list_pass[passenger_5[i].char][passenger_5[i].seat].t_1 = -1
709             list_pass[passenger_5[i].char][passenger_5[i].seat].t_2 = -1
710             list_pass[passenger_5[i].char][passenger_5[i].seat].check = 0
711         for i in range(0, len(passenger_5)):
712             plane[5+i][0].value = 1
713             plane[5+i][0].pass_char = passenger_5[i].char
714             plane[5+i][0].pass_seat = passenger_5[i].seat
715
716     def C6(passenger_6):
717         for i in range(len(passenger_6)):
718             list_pass[passenger_6[i].char][passenger_6[i].seat].char =
passenger_6[i].char
719             list_pass[passenger_6[i].char][passenger_6[i].seat].seat =
passenger_6[i].seat
720             list_pass[passenger_6[i].char][passenger_6[i].seat].bag = passenger_6
[i].bag
721             list_pass[passenger_6[i].char][passenger_6[i].seat].num_out = -1
722             list_pass[passenger_6[i].char][passenger_6[i].seat].t_1 = -1
723             list_pass[passenger_6[i].char][passenger_6[i].seat].t_2 = -1
724             list_pass[passenger_6[i].char][passenger_6[i].seat].check = 0
725         for i in range(0, len(passenger_6)):
726             plane[5+i][0].value = 1
727             plane[5+i][0].pass_char = passenger_6[i].char
728             plane[5+i][0].pass_seat = passenger_6[i].seat
729
730     if(case==1):
731         C1(passenger_1)
732     if(case==2):
733         C2(passenger_2)
734     if(case==3):
735         C3(passenger_3)
736     if(case==4):
737         C4(passenger_4)
738     if(case==5):
739         C5(passenger_5)
740     if(case==6):
741         C6(passenger_6)
742
743     a = [[0 for i in range(0,36)] for i in range(0,200)]
744
745     for i in range(0,200):
746         for j in range(0,36):
```

```
747         #print(plane[i][j].value)
748         a[i][j] = plane[i][j].type
749
750     time = 0
751
752     def check_pass(plane):
753         check = 0
754         for w in range(0,200):
755             for z in range(0,36):
756                 if(plane[w][z].type == 3 and plane[w][z].value == 1):
757                     check+=1
758         return check
759
760     time = 0
761     while(1):
762         #check
763         check = check_pass(plane)
764         if(check==N):
765             #print(time)
766             break
767         time+=1
768
769     i=2
770     for j in range(1,34):
771         if(plane[i][j].pass_char == 1 and plane[i][j].value == 1 and plane[i
-1][j].value == 0):
772             p1r = plane[i][j].pass_char
773             p1c = plane[i][j].pass_seat
774             plane[i-1][j].pass_char = p1r
775             plane[i-1][j].pass_seat = p1c
776             plane[i-1][j].value = 1
777             plane[i][j].pass_char = 0
778             plane[i][j].pass_seat = 0
779             plane[i][j].value = 0
780
781     i=6
782     for j in range(2,34):
783         if(plane[i][j].pass_char==7 and plane[i][j].value == 1 and plane[i
+1][j].value == 0):
784             p1r = plane[i][j].pass_char
785             p1c = plane[i][j].pass_seat
786             plane[i+1][j].pass_char = p1r
787             plane[i+1][j].pass_seat = p1c
788             plane[i+1][j].value = 1
789             plane[i][j].pass_char = 0
790             plane[i][j].pass_seat = 0
791             plane[i][j].value = 0
792
793     i=3
794     for j in range(1,34):
795         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):
796             p1r = plane[i][j].pass_char
797             p1c = plane[i][j].pass_seat
798             plane[i-1][j].pass_char = p1r
799             plane[i-1][j].pass_seat = p1c
800             plane[i-1][j].value = 1
801             plane[i][j].pass_char = 0
802             plane[i][j].pass_seat = 0
803             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
808 ] [j].value == 1 and plane[i+1][j].value == 0):
809             p1r = plane[i][j].pass_char
810             p1c = plane[i][j].pass_seat
811             plane[i+1][j].pass_char = p1r
812             plane[i+1][j].pass_seat = p1c
813             plane[i+1][j].value = 1
814             plane[i][j].pass_char = 0
815             plane[i][j].pass_seat = 0
816             plane[i][j].value = 0
817
818     i = 4
819     for j in reversed(range(0,36)):
820         if(plane[i][j].value==0):
821             continue
822         if(plane[i][j].pass_seat==j):
823             if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2>0)
:
824                 list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].t_2
825                 -=1
826                 continue
827             else:
828                 p1r = plane[i][j].pass_char
829                 p1c = plane[i][j].pass_seat
830                 if(plane[i][j].pass_char<4):
831                     if(plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
832 pass_seat].num_out-1][j].value == 1):
833                         p2r = plane[i-list_pass[plane[i][j].pass_char][plane[
834 i][j].pass_seat].num_out-1][j].pass_char
835                         p2c = plane[i-list_pass[plane[i][j].pass_char][plane[
836 i][j].pass_seat].num_out-1][j].pass_seat
837                         plane[i-1][j].pass_char = p2r
838                         plane[i-1][j].pass_seat = p2c
839                         plane[i-1][j].value = 1
840                         plane[i-2][j].pass_char = p1r
841                         plane[i-2][j].pass_seat = p1c
842                         plane[i-2][j].value = 1
843                         plane[i][j].pass_char = 0
844                         plane[i][j].pass_seat = 0
845                         plane[i][j].value = 0
846                         continue
847                     plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
848 pass_seat].num_out-1][j].pass_char = p1r
849                     plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
850 pass_seat].num_out-1][j].pass_seat = p1c
851                     plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
852 pass_seat].num_out-1][j].value = 1
853                 else:
854                     if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
855 pass_seat].num_out+1][j].value == 1):
856                         p2r = plane[i+list_pass[plane[i][j].pass_char][plane[
857 i][j].pass_seat].num_out+1][j].pass_char
858                         p2c = plane[i+list_pass[plane[i][j].pass_char][plane[
859 i][j].pass_seat].num_out+1][j].pass_seat
860                         plane[i+1][j].pass_char = p2r
861                         plane[i+1][j].pass_seat = p2c
862                         plane[i+1][j].value = 1

```

```

852         plane[i+2][j].pass_char = p1r
853         plane[i+2][j].pass_seat = p1c
854         plane[i+2][j].value = 1
855         plane[i][j].pass_char = 0
856         plane[i][j].pass_seat = 0
857         plane[i][j].value = 0
858         continue
859         if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value == 1):
860             continue
861             plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_char = p1r
862             plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_seat = p1c
863             plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value = 1
864             plane[i][j].pass_char = 0
865             plane[i][j].pass_seat = 0
866             plane[i][j].value = 0
867         if(plane[i][j].pass_seat>j):
868             if(plane[i][j].pass_seat-j==1):
869                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
870                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
871                     continue
872                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
check==0):
873                     sum = 0
874                     if(plane[i][j].pass_char<4):
875                         for m in range(1,4):
876                             if(plane[m][j+1].value == 1 and plane[m][j+1].
pass_char!=m):
877                                 sum+=1
878                     else:
879                         for m in range(5,8):
880                             if(plane[m][j+1].value == 1 and plane[m][j+1].
pass_char!=m):
881                                 sum+=1
882                     if(sum!=0):
883                         continue
884                     sum = 0
885                     if(plane[i][j].pass_char<4):
886                         for m in reversed(range(plane[i][j].pass_char+1,4)):
887                             if(plane[m][j+1].value == 1):
888                                 sum+=1
889                     else:
890                         for m in range(5,plane[i][j].pass_char):
891                             if(plane[m][j+1].value == 1):
892                                 sum+=1
893                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
num_out = sum
894                     mov = 0
895                     for n in range(0,list_pass[plane[i][j].pass_char][plane[i]
][j].pass_seat].num_out+1):
896                         if(plane[i][j+1+n].value==1):
897                             mov = 1
898                     if(mov == 1):
899                         continue
900                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].

```

```

check = 1
901         if(plane[i][j].pass_char<4):
902             if(plane[i][j].pass_char==3):
903                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
904                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
905             if(plane[i][j].pass_char==2):
906                 if(plane[i-1][j+1].value==1):
907                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 1
908                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 2
909                 else:
910                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
911                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
912             if(plane[i][j].pass_char==1):
913                 if(plane[i-1][j+1].value==1 and plane[i-2][j+1].
value==1):
914                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 2
915                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 3
916                 if(plane[i-1][j+1].value==1 and plane[i-2][j+1].
value==0):
917                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 1
918                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 2
919                 if(plane[i-1][j+1].value==0 and plane[i-2][j+1].
value==1):
920                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 2
921                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 2
922                 if(plane[i-1][j+1].value==0 and plane[i-2][j+1].
value==0):
923                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
924                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
925             else:
926                 if(plane[i][j].pass_char==5):
927                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
928                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
929                 if(plane[i][j].pass_char==6):
930                     if(plane[i+1][j+1].value==1):
931                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 1
932                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 2
933                     else:
934                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
935                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0

```

```

936         if(plane[i][j].pass_char==7):
937             if(plane[i+1][j+1].value==1 and plane[i+2][j+1].
value==1):
938                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 2
939                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 3
940             if(plane[i+1][j+1].value==1 and plane[i+2][j+1].
value==0):
941                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 1
942                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 2
943             if(plane[i+1][j+1].value==0 and plane[i+2][j+1].
value==1):
944                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 2
945                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 2
946             if(plane[i+1][j+1].value==0 and plane[i+2][j+1].
value==0):
947                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
948                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
949             if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
check == 1):
950                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].t_1>0):
951                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1-=1
952             else:
953                 if(plane[i][j+1].value==1):
954                     continue
955                 p1r = plane[i][j].pass_char
956                 p1c = plane[i][j].pass_seat
957                 plane[i][j+1].pass_char = p1r
958                 plane[i][j+1].pass_seat = p1c
959                 plane[i][j+1].value = 1
960                 plane[i][j].pass_char = 0
961                 plane[i][j].pass_seat = 0
962                 plane[i][j].value = 0
963             else:
964                 if(plane[i][j+1].value==0):
965                     p1r = plane[i][j].pass_char
966                     p1c = plane[i][j].pass_seat
967                     plane[i][j+1].pass_char = p1r
968                     plane[i][j+1].pass_seat = p1c
969                     plane[i][j+1].value = 1
970                     plane[i][j].pass_char = 0
971                     plane[i][j].pass_seat = 0
972                     plane[i][j].value = 0
973
974         j = 0
975         for i in range(5,200):
976             if(plane[i-1][j].value==0 and plane[i][j].value==1):
977                 p1r = plane[i][j].pass_char
978                 p1c = plane[i][j].pass_seat
979                 plane[i-1][j].pass_char = p1r
980                 plane[i-1][j].pass_seat = p1c

```

```

981         plane[i-1][j].value = 1
982         plane[i][j].pass_char = 0
983         plane[i][j].pass_seat = 0
984         plane[i][j].value = 0
985
986         #print("time",time)
987         #print("check",check)
988         """i = 4
989         for j in reversed(range(0,36)):
990             if(plane[i][j].value==1):
991                 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)
992                 print("\n")"""
993         """for i in range(0,200):
994             for j in range(0,36):
995                 #print(plane[i][j].value)
996                 a[i][j] = plane[i][j].value
997         plt.figure('time'+str(time))
998         im = plt.imshow(a[0:8])
999         ax= plt.gca()
1000         ax.set_xticks(np.arange(-.5, 36, 1), minor=True)
1001         ax.set_yticks(np.arange(-.5, 8, 1), minor=True)
1002         ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
1003         plt.savefig('time'+str(time)+'.png')
1004         #plt.show()"""
1005     return time
1006
1007 print(run1(1,0,0.3,195))

```

B.2 Codes of Boarding 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
10 def run2(case,RL,RJ,N):
11     # Create position of people in "Narrow Body" Passenger Aircraft
12     arr = np.arange(1,319)
13     list_0 = arr.tolist()
14     for i in range (33):
15         list_0[i] = [math.floor(list_0[i]/11)+1, (list_0[i]%11)+3]
16     for i in range (33,117):
17         list_0[i] = [math.floor((list_0[i]-33)/14)+5, ((list_0[i]-33)%14)]
18     for i in range (117,201):
19         list_0[i] = [math.floor((list_0[i]-117)/14)+12, ((list_0[i]-117)%14)]
20     for i in range (201,285):
21         list_0[i] = [math.floor((list_0[i]-201)/14)+19, ((list_0[i]-201)%14)]
22     for i in range (285,318):
23         list_0[i] = [math.floor((list_0[i]-285)/11)+26, ((list_0[i]-285)%11)+3]
24     for i in range (len(list_0)):
25         if list_0[i][1] == 13:
26             list_0[i+1] = [list_0[i][0], 14]

```



```

27
28 # Random luggage stow time of each people by experimental data and Weibull
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))

```

```
75 list_correct_2 = list_random_in_2 + list_random_mid_2 + list_random_out_2
76
77 list_late_2 = random.sample(list_correct_2, round(N*RL))
78 list_no_late_2 = [x for x in list_correct_2 if x not in list_late_2]
79
80 list_bad_2 = random.sample(list_no_late_2, round(N*RJ))
81
82 random_2 = []
83 for i in range (len(list_no_late_2)):
84     for j in range (len(list_bad_2)):
85         if list_no_late_2[i] == list_bad_2[j]:
86             random_2.append(i)
87
88 list_bad_2 = []
89 for i in range (len(random_2)):
90     list_bad_2.append(list_no_late_2[random_2[i]])
91
92 list_no_bad_2 = [x for x in list_no_late_2 if x not in list_bad_2]
93
94 no_random_2 = []
95 for i in range (len(list_no_late_2)):
96     for j in range (len(list_no_bad_2)):
97         if list_no_late_2[i] == list_no_bad_2[j]:
98             no_random_2.append(i)
99
100 normal_2 = []
101 for i in range (len(random_2)):
102     normal_2.append(np.random.normal(random_2[i], 5))
103
104 for i in range (len(normal_2)):
105     list_bad_2[i].append(normal_2[i])
106
107 list_bad_2 = sorted(list_bad_2, key=lambda x: x[-1])
108
109 for i in range (len(normal_2)):
110     del list_bad_2[i][3]
111
112 normal_2 = sorted(normal_2)
113
114 list_shift_2 = []
115 j=0
116 for i in range (len(list_no_bad_2)+len(list_bad_2)):
117     if (len(list_bad_2) != 0) & (j < len(no_random_2)):
118         if normal_2[0] < no_random_2[j]:
119             list_shift_2.append(list_bad_2[0])
120             del list_bad_2[0]
121             del normal_2[0]
122         elif normal_2[0] > no_random_2[j]:
123             list_shift_2.append(list_no_bad_2[j])
124             j=j+1
125     elif j >= len(no_random_2):
126         list_shift_2.append(list_bad_2[0])
127         del list_bad_2[0]
128         del normal_2[0]
129     else:
130         list_shift_2.append(list_no_bad_2[j])
131         j=j+1
132
133 list_2 = list_shift_2 + list_late_2
134
```

```
135 class agent_2:
136     def __init__(self, char, seat, bag, num_out, row, col):
137         self.char = char
138         self.seat = seat
139         self.bag = bag
140         self.num_out = num_out
141         self.row = row
142         self.col = col
143
144 passenger_2 = []
145 for i in range(len(list_0)):
146     passenger_2.append(agent_2(list_2[i][0], list_2[i][1], list_2[i][2], 0, 0, 0))
147
148 #Case3
149 list_1_3 = []
150 list_2_3 = []
151 list_3_3 = []
152 list_4_3 = []
153 list_5_3 = []
154 list_6_3 = []
155 list_7_3 = []
156 list_8_3 = []
157 list_9_3 = []
158 list_10_3 = []
159 list_11_3 = []
160 list_12_3 = []
161
162 for i in range(len(list_0)):
163     if (list_0[i][0] == 1) or (list_0[i][0] == 7):
164         list_1_3.append(list_0[i])
165 for i in range(len(list_0)):
166     if (list_0[i][0] == 8) or (list_0[i][0] == 14):
167         list_2_3.append(list_0[i])
168 for i in range(len(list_0)):
169     if (list_0[i][0] == 15) or (list_0[i][0] == 21):
170         list_3_3.append(list_0[i])
171 for i in range(len(list_0)):
172     if (list_0[i][0] == 22) or (list_0[i][0] == 28):
173         list_4_3.append(list_0[i])
174 for i in range(len(list_0)):
175     if (list_0[i][0] == 2) or (list_0[i][0] == 6):
176         list_5_3.append(list_0[i])
177 for i in range(len(list_0)):
178     if (list_0[i][0] == 9) or (list_0[i][0] == 13):
179         list_6_3.append(list_0[i])
180 for i in range(len(list_0)):
181     if (list_0[i][0] == 16) or (list_0[i][0] == 20):
182         list_7_3.append(list_0[i])
183 for i in range(len(list_0)):
184     if (list_0[i][0] == 23) or (list_0[i][0] == 27):
185         list_8_3.append(list_0[i])
186 for i in range(len(list_0)):
187     if (list_0[i][0] == 3) or (list_0[i][0] == 5):
188         list_9_3.append(list_0[i])
189 for i in range(len(list_0)):
190     if (list_0[i][0] == 10) or (list_0[i][0] == 12):
191         list_10_3.append(list_0[i])
192 for i in range(len(list_0)):
193     if (list_0[i][0] == 17) or (list_0[i][0] == 19):
194         list_11_3.append(list_0[i])
```

```

195     for i in range (len(list_0)):
196         if (list_0[i][0] == 24) or (list_0[i][0] == 26):
197             list_12_3.append(list_0[i])
198
199     list_random_1_3 = random.sample(list_1_3, len(list_1_3))
200     list_random_2_3 = random.sample(list_2_3, len(list_2_3))
201     list_random_3_3 = random.sample(list_3_3, len(list_3_3))
202     list_random_4_3 = random.sample(list_4_3, len(list_4_3))
203     list_random_5_3 = random.sample(list_5_3, len(list_5_3))
204     list_random_6_3 = random.sample(list_6_3, len(list_6_3))
205     list_random_7_3 = random.sample(list_7_3, len(list_7_3))
206     list_random_8_3 = random.sample(list_8_3, len(list_8_3))
207     list_random_9_3 = random.sample(list_9_3, len(list_9_3))
208     list_random_10_3 = random.sample(list_10_3, len(list_10_3))
209     list_random_11_3 = random.sample(list_11_3, len(list_11_3))
210     list_random_12_3 = random.sample(list_12_3, len(list_12_3))
211     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
212
213     list_late_3 = random.sample(list_correct_3, round(N*RL))
214     list_no_late_3 = [x for x in list_correct_3 if x not in list_late_3]
215
216     list_bad_3 = random.sample(list_no_late_3, round(N*RJ))
217
218     random_3 = []
219     for i in range (len(list_no_late_3)):
220         for j in range (len(list_bad_3)):
221             if list_no_late_3[i] == list_bad_3[j]:
222                 random_3.append(i)
223
224     list_bad_3 = []
225     for i in range (len(random_3)):
226         list_bad_3.append(list_no_late_3[random_3[i]])
227
228     list_no_bad_3 = [x for x in list_no_late_3 if x not in list_bad_3]
229
230     no_random_3 = []
231     for i in range (len(list_no_late_3)):
232         for j in range (len(list_no_bad_3)):
233             if list_no_late_3[i] == list_no_bad_3[j]:
234                 no_random_3.append(i)
235
236     normal_3 = []
237     for i in range (len(random_3)):
238         normal_3.append(np.random.normal(random_3[i], 5))
239
240     for i in range (len(normal_3)):
241         list_bad_3[i].append(normal_3[i])
242
243     list_bad_3 = sorted(list_bad_3, key=lambda x: x[-1])
244
245     for i in range (len(normal_3)):
246         del list_bad_3[i][3]
247
248     normal_3 = sorted(normal_3)
249
250     list_shift_3 = []
251     j=0

```

```

252     for i in range (len(list_no_bad_3)+len(list_bad_3)):
253         if (len(list_bad_3) != 0) & (j < len(no_random_3)):
254             if normal_3[0] < no_random_3[j]:
255                 list_shift_3.append(list_bad_3[0])
256                 del list_bad_3[0]
257                 del normal_3[0]
258             elif normal_3[0] > no_random_3[j]:
259                 list_shift_3.append(list_no_bad_3[j])
260                 j=j+1
261         elif j >= len(no_random_3):
262             list_shift_3.append(list_bad_3[0])
263             del list_bad_3[0]
264             del normal_3[0]
265         else:
266             list_shift_3.append(list_no_bad_3[j])
267             j=j+1
268
269     list_3 = list_shift_3 + list_late_3
270
271     class agent_3:
272         def __init__(self, char, seat, bag, num_out, row, col):
273             self.char = char
274             self.seat = seat
275             self.bag = bag
276             self.num_out = num_out
277             self.row = row
278             self.col = col
279
280     passenger_3 = []
281     for i in range(len(list_0)):
282         passenger_3.append(agent_3(list_3[i][0], list_3[i][1], list_3[i][2], 0, 0, 0))
283
284     #Case4
285
286     list_1_4 = []
287     list_2_4 = []
288     list_3_4 = []
289     list_4_4 = []
290     list_5_4 = []
291     list_6_4 = []
292
293     for i in range (len(list_0)):
294         if ((list_0[i][0] == 1) and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0]
295 == 7) and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 8) and (9 <= list_0[
296 i][1] <= 14)) or ((list_0[i][0] == 14) and (9 <= list_0[i][1] <= 14)) or ((
297 list_0[i][0] == 15) and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 21)
298 and (9 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 22) and (9 <= list_0[i][1]
299 <= 14)) or ((list_0[i][0] == 28) and (9 <= list_0[i][1] <= 14)):
300             list_1_4.append(list_0[i])
301         for i in range (len(list_0)):
302             if ((list_0[i][0] == 2) and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0]
303 == 6) and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 9) and (8 <= list_0[
304 i][1] <= 14)) or ((list_0[i][0] == 13) and (8 <= list_0[i][1] <= 14)) or ((
305 list_0[i][0] == 16) and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 20)
306 and (8 <= list_0[i][1] <= 14)) or ((list_0[i][0] == 23) and (8 <= list_0[i][1]
307 <= 14)) or ((list_0[i][0] == 27) and (8 <= list_0[i][1] <= 14)):
308                 list_2_4.append(list_0[i])
309         for i in range (len(list_0)):
310             if ((list_0[i][0] == 3) and (7 <= list_0[i][1] <= 14)) or ((list_0[i][0]
311 == 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]:

```

```
341         no_random_4.append(i)
342
343     normal_4 = []
344     for i in range (len(random_4)):
345         normal_4.append(np.random.normal(random_4[i], 5))
346
347     for i in range (len(normal_4)):
348         list_bad_4[i].append(normal_4[i])
349
350     list_bad_4 = sorted(list_bad_4, key=lambda x: x[-1])
351
352     for i in range (len(normal_4)):
353         del list_bad_4[i][3]
354
355     normal_4 = sorted(normal_4)
356
357     list_shift_4 = []
358     j=0;
359     for i in range (len(list_no_bad_4)+len(list_bad_4)):
360         if (len(list_bad_4) != 0) & (j < len(no_random_4)):
361             if normal_4[0] < no_random_4[j]:
362                 list_shift_4.append(list_bad_4[0])
363                 del list_bad_4[0]
364                 del normal_4[0]
365             elif normal_4[0] > no_random_4[j]:
366                 list_shift_4.append(list_no_bad_4[j])
367                 j=j+1
368             elif j >= len(no_random_4):
369                 list_shift_4.append(list_bad_4[0])
370                 del list_bad_4[0]
371                 del normal_4[0]
372             else:
373                 list_shift_4.append(list_no_bad_4[j])
374                 j=j+1
375
376     list_4 = list_shift_4 + list_late_4
377
378     class agent_4:
379         def __init__(self, char, seat, bag, num_out, row, col):
380             self.char = char
381             self.seat = seat
382             self.bag = bag
383             self.num_out = num_out
384             self.row = row
385             self.col = col
386
387     passenger_4 = []
388     for i in range(len(list_0)):
389         passenger_4.append(agent_4(list_4[i][0], list_4[i][1], list_4[i][2], 0, 0, 0))
390
391     class person:
392         def __init__(self, char, seat, bag, num_out, t_1, t_2, check):
393             self.char = char
394             self.seat = seat
395             self.bag = bag
396             self.num_out = num_out
397             self.t_1 = t_1
398             self.t_2 = t_2
399             self.check = check
400
```

```
401 list_pass = [[person(0,0,0,0,0,0,0) for i in range(0,15)] for j in range
(0,29)]
402
403 class grid:
404     def __init__(self, type, value, pass_char, pass_seat):
405         self.type = type
406         # 0 -> block
407         # 1 -> queue
408         # 2 -> aisle
409         # 3 -> seat
410         self.value = value
411         # 0 -> available
412         # 1 -> passenger
413         self.pass_char = pass_char
414         self.pass_seat = pass_seat
415
416 plane = [[grid(0,0,0,0) for i in range(0,17)] for j in range(0,344)]
417
418 for i in range(1,4):
419     for j in range(4,15):
420         plane[i][j].type = 3
421
422 for i in range(5,11):
423     for j in range(1,15):
424         plane[i][j].type = 3
425
426 for i in range(12,18):
427     for j in range(1,15):
428         plane[i][j].type = 3
429
430 for i in range(19,25):
431     for j in range(1,15):
432         plane[i][j].type = 3
433
434 for i in range(26,29):
435     for j in range(4,15):
436         plane[i][j].type = 3
437
438 for i in range(4,344):
439     plane[i][0].type = 1
440
441 q = [4,11,18,25]
442 for i in q:
443     for j in range(0,17):
444         plane[i][j].type = 2
445
446 def C1(passenger_1):
447     for i in range(len(passenger_1)):
448         list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
449         list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
450         list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
451         list_pass[passenger_1[i].char][passenger_1[i].seat].num_out = -1
452         list_pass[passenger_1[i].char][passenger_1[i].seat].t_1 = -1
453         list_pass[passenger_1[i].char][passenger_1[i].seat].t_2 = -1
454         list_pass[passenger_1[i].char][passenger_1[i].seat].check = 0
455     for i in range(0, len(passenger_1)):
456         plane[26+i][0].value = 1
```



```
457     plane[26+i][0].pass_char = passenger_1[i].char
458     plane[26+i][0].pass_seat = passenger_1[i].seat
459
460     def C2(passenger_2):
461         for i in range(len(passenger_2)):
462             list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
463             list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
passenger_2[i].seat
464             list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
[i].bag
465             list_pass[passenger_2[i].char][passenger_2[i].seat].num_out = -1
466             list_pass[passenger_2[i].char][passenger_2[i].seat].t_1 = -1
467             list_pass[passenger_2[i].char][passenger_2[i].seat].t_2 = -1
468             list_pass[passenger_2[i].char][passenger_2[i].seat].check = 0
469         for i in range(0, len(passenger_2)):
470             plane[26+i][0].value = 1
471             plane[26+i][0].pass_char = passenger_2[i].char
472             plane[26+i][0].pass_seat = passenger_2[i].seat
473
474     def C3(passenger_3):
475         for i in range(len(passenger_3)):
476             list_pass[passenger_3[i].char][passenger_3[i].seat].char =
passenger_3[i].char
477             list_pass[passenger_3[i].char][passenger_3[i].seat].seat =
passenger_3[i].seat
478             list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3
[i].bag
479             list_pass[passenger_3[i].char][passenger_3[i].seat].num_out = -1
480             list_pass[passenger_3[i].char][passenger_3[i].seat].t_1 = -1
481             list_pass[passenger_3[i].char][passenger_3[i].seat].t_2 = -1
482             list_pass[passenger_3[i].char][passenger_3[i].seat].check = 0
483         for i in range(0, len(passenger_3)):
484             plane[26+i][0].value = 1
485             plane[26+i][0].pass_char = passenger_3[i].char
486             plane[26+i][0].pass_seat = passenger_3[i].seat
487
488     def C4(passenger_4):
489         for i in range(len(passenger_4)):
490             list_pass[passenger_4[i].char][passenger_4[i].seat].char =
passenger_4[i].char
491             list_pass[passenger_4[i].char][passenger_4[i].seat].seat =
passenger_4[i].seat
492             list_pass[passenger_4[i].char][passenger_4[i].seat].bag = passenger_4
[i].bag
493             list_pass[passenger_4[i].char][passenger_4[i].seat].num_out = -1
494             list_pass[passenger_4[i].char][passenger_4[i].seat].t_1 = -1
495             list_pass[passenger_4[i].char][passenger_4[i].seat].t_2 = -1
496             list_pass[passenger_4[i].char][passenger_4[i].seat].check = 0
497         for i in range(0, len(passenger_4)):
498             plane[26+i][0].value = 1
499             plane[26+i][0].pass_char = passenger_4[i].char
500             plane[26+i][0].pass_seat = passenger_4[i].seat
501
502     if(case==1):
503         C1(passenger_1)
504     if(case==2):
505         C2(passenger_2)
506     if(case==3):
507         C3(passenger_3)
```



```

617         plane[i-1][j].pass_seat = p2c
618         plane[i-1][j].value = 1
619         plane[i-2][j].pass_char = p1r
620         plane[i-2][j].pass_seat = p1c
621         plane[i-2][j].value = 1
622         plane[i][j].pass_char = 0
623         plane[i][j].pass_seat = 0
624         plane[i][j].value = 0
625         continue
626         plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].pass_char = p1r
627         plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].pass_seat = p1c
628         plane[i-list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out-1][j].value = 1
629     else:
630         if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value == 1):
631             p2r = plane[i+list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1][j].pass_char
632             p2c = plane[i+list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1][j].pass_seat
633             plane[i+1][j].pass_char = p2r
634             plane[i+1][j].pass_seat = p2c
635             plane[i+1][j].value = 1
636             plane[i+2][j].pass_char = p1r
637             plane[i+2][j].pass_seat = p1c
638             plane[i+2][j].value = 1
639             plane[i][j].pass_char = 0
640             plane[i][j].pass_seat = 0
641             plane[i][j].value = 0
642             continue
643             if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value == 1):
644                 continue
645                 plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_char = p1r
646                 plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].pass_seat = p1c
647                 plane[i+list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out+1][j].value = 1
648                 plane[i][j].pass_char = 0
649                 plane[i][j].pass_seat = 0
650                 plane[i][j].value = 0
651             if(plane[i][j].pass_seat>j):
652                 if(plane[i][j].pass_seat-j==1):
653                     if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].bag>0):
654                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].bag-=1
655                         continue
656                     if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].check==0):
657                         sum = 0
658                         if(plane[i][j].pass_char<i):
659                             for m in range(i-3,i):
660                                 if(plane[m][j+1].value == 1 and plane[m][j
+1].pass_char!=m):
661                                     sum+=1
662                         else:

```

```

663         for m in range(i+1,i+4):
664             if(plane[m][j+1].value == 1 and plane[m][j
+1].pass_char!=m):
665                 sum+=1
666                 if(sum!=0):
667                     continue
668                 sum = 0
669                 if(plane[i][j].pass_char<i):
670                     for m in reversed(range(plane[i][j].pass_char+1,i
)):
671                         if(plane[m][j+1].value == 1):
672                             sum+=1
673                     else:
674                         for m in range(i+1,plane[i][j].pass_char):
675                             if(plane[m][j+1].value == 1):
676                                 sum+=1
677                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out = sum
678                 mov = 0
679                 for n in range(0,list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1):
680                     if(plane[i][j+1+n].value==1):
681                         mov = 1
682                 if(mov == 1):
683                     continue
684                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].check = 1
685                 if(plane[i][j].pass_char<i):
686                     if(plane[i][j].pass_char==i-1):
687                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
688                         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
689                     if(plane[i][j].pass_char==i-2):
690                         if(plane[i-1][j+1].value==1):
691                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
692                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
693                         else:
694                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
695                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
696                     if(plane[i][j].pass_char==i-3):
697                         if(plane[i-1][j+1].value==1 and plane[i-2][j
+1].value==1):
698                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 2
699                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 3
700                         if(plane[i-1][j+1].value==1 and plane[i-2][j
+1].value==0):
701                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
702                             list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
703                         if(plane[i-1][j+1].value==0 and plane[i-2][j
+1].value==1):
704                             list_pass[plane[i][j].pass_char][plane[i

```

```

] [j].pass_seat).t_1 = 2
705         list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 2
706         if (plane [i-1] [j+1].value==0 and plane [i-2] [j
+1].value==0):
707             list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 0
708             list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 0
709         else:
710             if (plane [i] [j].pass_char==i+1):
711                 list_pass [plane [i] [j].pass_char] [plane [i] [j].
pass_seat).t_1 = 0
712                 list_pass [plane [i] [j].pass_char] [plane [i] [j].
pass_seat).t_2 = 0
713             if (plane [i] [j].pass_char==i+2):
714                 if (plane [i+1] [j+1].value==1):
715                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 1
716                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 2
717                 else:
718                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 0
719                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 0
720             if (plane [i] [j].pass_char==i+3):
721                 if (plane [i+1] [j+1].value==1 and plane [i+2] [j
+1].value==1):
722                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 2
723                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 3
724                 if (plane [i+1] [j+1].value==1 and plane [i+2] [j
+1].value==0):
725                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 1
726                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 2
727                 if (plane [i+1] [j+1].value==0 and plane [i+2] [j
+1].value==1):
728                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 2
729                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 2
730                 if (plane [i+1] [j+1].value==0 and plane [i+2] [j
+1].value==0):
731                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_1 = 0
732                     list_pass [plane [i] [j].pass_char] [plane [i
] [j].pass_seat).t_2 = 0
733             if (list_pass [plane [i] [j].pass_char] [plane [i] [j].pass_seat
].check == 1):
734                 if (list_pass [plane [i] [j].pass_char] [plane [i] [j].
pass_seat).t_1>0):
735                     list_pass [plane [i] [j].pass_char] [plane [i] [j].
pass_seat).t_1-=1
736             else:
737                 if (plane [i] [j+1].value==1):
738                     continue

```

```

739     p1r = plane[i][j].pass_char
740     p1c = plane[i][j].pass_seat
741     plane[i][j+1].pass_char = p1r
742     plane[i][j+1].pass_seat = p1c
743     plane[i][j+1].value = 1
744     plane[i][j].pass_char = 0
745     plane[i][j].pass_seat = 0
746     plane[i][j].value = 0
747
748     else:
749         if(plane[i][j+1].value==0):
750             p1r = plane[i][j].pass_char
751             p1c = plane[i][j].pass_seat
752             plane[i][j+1].pass_char = p1r
753             plane[i][j+1].pass_seat = p1c
754             plane[i][j+1].value = 1
755             plane[i][j].pass_char = 0
756             plane[i][j].pass_seat = 0
757             plane[i][j].value = 0
758
759     j = 0
760     for i in range(5,344):
761         if(plane[i-1][j].value==1):
762             continue
763         else:
764             if(plane[i][j].value==1):
765                 if(math.ceil(plane[i][j].pass_char/7)*7-3<i):
766                     p1r = plane[i][j].pass_char
767                     p1c = plane[i][j].pass_seat
768                     plane[i-1][j].pass_char = p1r
769                     plane[i-1][j].pass_seat = p1c
770                     plane[i-1][j].value = 1
771                     plane[i][j].pass_char = 0
772                     plane[i][j].pass_seat = 0
773                     plane[i][j].value = 0
774
775     """print("time",time)
776     print("check",check)
777
778     for i in range(0,344):
779         for j in range(0,17):
780             #print(plane[i][j].value)
781             a[i][j] = plane[i][j].value
782     plt.figure('time'+str(time))
783     im = plt.imshow(a[0:29])
784     ax = plt.gca()
785     ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
786     ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
787     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
788     plt.savefig('time'+str(time)+'.png')
789     plt.show()"""
790     return time
791 print(run2(3,0,0,318))

```

B.3 Codes of Boarding Process, Airplane III

```

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
10 def run3(case,RL,RJ,N):
11     N1 = 95
12     N2 = 147
13     # Create position of people in "Narrow Body" Passenger Aircraft
14     arr = np.arange(1,243)
15     list_0 = arr.tolist()
16     for i in range (28):
17         list_0[i] = [math.floor(list_0[i]/14)+1, (list_0[i]%14)]
18     for i in range (28,67):
19         list_0[i] = [math.floor((list_0[i]-28)/13)+4, ((list_0[i]-28)%13)]
20     for i in range (67,95):
21         list_0[i] = [math.floor((list_0[i]-67)/14)+8, ((list_0[i]-67)%14)]
22     for i in range (95,137):
23         list_0[i] = [math.floor((list_0[i]-95)/21)+1, ((list_0[i]-95)%21)+19]
24     for i in range (137,200):
25         list_0[i] = [math.floor((list_0[i]-137)/21)+4, ((list_0[i]-137)%21)+19]
26     for i in range (200,242):
27         list_0[i] = [math.floor((list_0[i]-200)/21)+8, ((list_0[i]-200)%21)+19]
28     for i in range (242):
29         if (list_0[i][1] == 0) | (list_0[i][1] == 19):
30             list_0[i] = [list_0[i-1][0],list_0[i-1][1]+1]
31
32     # Random luggage stow time of each people by experimental data and Weibull
33     # distribution
34     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,
35     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,
36     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,
37     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,
38     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,
39     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]
40     luggage_arr = np.array(luggage)
41     mean = luggage_arr.mean()
42     std = stdev(luggage_arr)
43     k = (std/mean)**(-1.086)
44     z = 1 + 1/k
45     def f(x):
46         return math.exp(-x)*(x**(z-1))
47     gamma,err = quad(f, 0, math.inf)
48     c = mean/gamma
49     for i in range (len(list_0)):
50         weibull = (c*(np.random.weibull(k, 1))).tolist()
51         time = round((weibull[0]/1.42))
52         list_0[i].append(time)
53
54     #Separate 2 entrances
55     list_0_e1 = []
56     list_0_e2 = []
57     for i in range (len(list_0)):
58         if list_0[i][1] < 17:
59             list_0_e1.append(list_0[i])
60     for i in range (len(list_0)):
61         if list_0[i][1] > 17:
62             list_0_e2.append(list_0[i])
63
64     #Case1
65     list_1_e1 = random.sample(list_0_e1,len(list_0_e1))

```



```
60 list_1_e2 = random.sample(list_0_e2, len(list_0_e2))
61
62 class agent_1_e1:
63     def __init__(self, char, seat, bag):
64         self.char = char
65         self.seat = seat
66         self.bag = bag
67
68 class agent_1_e2:
69     def __init__(self, char, seat, bag):
70         self.char = char
71         self.seat = seat
72         self.bag = bag
73
74 passenger_1_e1 = []
75 for i in range(len(list_0_e1)):
76     passenger_1_e1.append(agent_1_e1(list_1_e1[i][0], list_1_e1[i][1],
list_1_e1[i][2]))
77
78 passenger_1_e2 = []
79 for i in range(len(list_0_e2)):
80     passenger_1_e2.append(agent_1_e2(list_1_e2[i][0], list_1_e2[i][1],
list_1_e2[i][2]))
81
82 #Case2
83 list_1_2_e1 = []
84 list_2_2_e1 = []
85 list_3_2_e1 = []
86 list_4_2_e1 = []
87
88 for i in range (len(list_0_e1)):
89     if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5):
90         list_1_2_e1.append(list_0_e1[i])
91 for i in range (len(list_0_e1)):
92     if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4):
93         list_2_2_e1.append(list_0_e1[i])
94 for i in range (len(list_0_e1)):
95     if (list_0_e1[i][0] == 9):
96         list_3_2_e1.append(list_0_e1[i])
97 for i in range (len(list_0_e1)):
98     if (list_0_e1[i][0] == 6) | (list_0_e1[i][0] == 8):
99         list_4_2_e1.append(list_0_e1[i])
100
101 list_random_1_2_e1 = random.sample(list_1_2_e1, len(list_1_2_e1))
102 list_random_2_2_e1 = random.sample(list_2_2_e1, len(list_2_2_e1))
103 list_random_3_2_e1 = random.sample(list_3_2_e1, len(list_3_2_e1))
104 list_random_4_2_e1 = random.sample(list_4_2_e1, len(list_4_2_e1))
105 list_correct_2_e1 = list_random_1_2_e1 + list_random_2_2_e1 +
list_random_3_2_e1 + list_random_4_2_e1
106
107 list_late_2_e1 = random.sample(list_correct_2_e1, round(N1*RL))
108 list_no_late_2_e1 = [x for x in list_correct_2_e1 if x not in list_late_2_e1]
109
110 list_bad_2_e1 = random.sample(list_no_late_2_e1, round(N1*RJ))
111
112 random_2_e1 = []
113 for i in range (len(list_no_late_2_e1)):
114     for j in range (len(list_bad_2_e1)):
115         if list_no_late_2_e1[i] == list_bad_2_e1[j]:
116             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))
177
178 list_1_2_e2 = []
179 list_2_2_e2 = []
180 list_3_2_e2 = []
181 list_4_2_e2 = []
182
183 for i in range (len(list_0_e2)):
184     if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5):
185         list_1_2_e2.append(list_0_e2[i])
186 for i in range (len(list_0_e2)):
187     if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4):
188         list_2_2_e2.append(list_0_e2[i])
189 for i in range (len(list_0_e2)):
190     if (list_0_e2[i][0] == 9):
191         list_3_2_e2.append(list_0_e2[i])
192 for i in range (len(list_0_e2)):
193     if (list_0_e2[i][0] == 6) | (list_0_e2[i][0] == 8):
194         list_4_2_e2.append(list_0_e2[i])
195
196 list_random_1_2_e2 = random.sample(list_1_2_e2, len(list_1_2_e2))
197 list_random_2_2_e2 = random.sample(list_2_2_e2, len(list_2_2_e2))
198 list_random_3_2_e2 = random.sample(list_3_2_e2, len(list_3_2_e2))
199 list_random_4_2_e2 = random.sample(list_4_2_e2, len(list_4_2_e2))
200 list_correct_2_e2 = list_random_1_2_e2 + list_random_2_2_e2 +
list_random_3_2_e2 + list_random_4_2_e2
201
202 list_late_2_e2 = random.sample(list_correct_2_e2, round(N2*RL))
203 list_no_late_2_e2 = [x for x in list_correct_2_e2 if x not in list_late_2_e2]
204
205 list_bad_2_e2 = random.sample(list_no_late_2_e2, round(N2*RJ))
206
207 random_2_e2 = []
208 for i in range (len(list_no_late_2_e2)):
209     for j in range (len(list_bad_2_e2)):
210         if list_no_late_2_e2[i] == list_bad_2_e2[j]:
211             random_2_e2.append(i)
212
213 list_bad_2_e2 = []
214 for i in range (len(random_2_e2)):
215     list_bad_2_e2.append(list_no_late_2_e2[random_2_e2[i]])
216
217 list_no_bad_2_e2 = [x for x in list_no_late_2_e2 if x not in list_bad_2_e2]
218
219 no_random_2_e2 = []
220 for i in range (len(list_no_late_2_e2)):
221     for j in range (len(list_no_bad_2_e2)):
222         if list_no_late_2_e2[i] == list_no_bad_2_e2[j]:
223             no_random_2_e2.append(i)
224
225 normal_2_e2 = []
226 for i in range (len(random_2_e2)):
227     normal_2_e2.append(np.random.normal(random_2_e2[i], 5))
228
229 for i in range (len(normal_2_e2)):
230     list_bad_2_e2[i].append(normal_2_e2[i])
231
232 list_bad_2_e2 = sorted(list_bad_2_e2, key=lambda x: x[-1])
233
234 for i in range (len(normal_2_e2)):

```

```

235         del list_bad_2_e2[i][3]
236
237     normal_2_e2 = sorted(normal_2_e2)
238
239     list_shift_2_e2 = []
240     j=0;
241     for i in range (len(list_no_bad_2_e2)+len(list_bad_2_e2)):
242         if (len(list_bad_2_e2) != 0) & (j < len(no_random_2_e2)):
243             if normal_2_e2[0] < no_random_2_e2[j]:
244                 list_shift_2_e2.append(list_bad_2_e2[0])
245                 del list_bad_2_e2[0]
246                 del normal_2_e2[0]
247             elif normal_2_e2[0] > no_random_2_e2[j]:
248                 list_shift_2_e2.append(list_no_bad_2_e2[j])
249                 j=j+1
250         elif j >= len(no_random_2_e2):
251             list_shift_2_e2.append(list_bad_2_e2[0])
252             del list_bad_2_e2[0]
253             del normal_2_e2[0]
254         else:
255             list_shift_2_e2.append(list_no_bad_2_e2[j])
256             j=j+1
257
258     list_2_e2 = list_shift_2_e2 + list_late_2_e2
259
260     class agent_2_e2:
261         def __init__(self, char, seat, bag, num_out, row, col):
262             self.char = char
263             self.seat = seat
264             self.bag = bag
265             self.num_out = num_out
266             self.row = row
267             self.col = col
268
269     passenger_2_e2 = []
270     for i in range(len(list_0_e2)):
271         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))
272
273     #Case3
274     list_1_3_e1 = []
275     list_2_3_e1 = []
276
277     for i in range (len(list_0_e1)):
278         if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5):
279             list_1_3_e1.append(list_0_e1[i])
280     for i in range (len(list_0_e1)):
281         if (list_0_e1[i][0] == 9) & (1 <= list_0_e1[i][1] <= 14):
282             list_1_3_e1.append(list_0_e1[i])
283     for i in range (len(list_0_e1)):
284         if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4):
285             list_2_3_e1.append(list_0_e1[i])
286     for i in range (len(list_0_e1)):
287         if (list_0_e1[i][0] == 6) & (1 <= list_0_e1[i][0] <= 13):
288             list_2_3_e1.append(list_0_e1[i])
289     for i in range (len(list_0_e1)):
290         if (list_0_e1[i][0] == 8) & (1 <= list_0_e1[i][0] <= 14):
291             list_2_3_e1.append(list_0_e1[i])
292
293     list_random_1_3_e1 = random.sample(list_1_3_e1, len(list_1_3_e1))

```

```
294 list_random_2_3_e1 = random.sample(list_2_3_e1, len(list_2_3_e1))
295 list_correct_3_e1 = list_random_1_3_e1 + list_random_2_3_e1
296
297 list_late_3_e1 = random.sample(list_correct_3_e1, round(N1*RL))
298 list_no_late_3_e1 = [x for x in list_correct_3_e1 if x not in list_late_3_e1]
299
300 list_bad_3_e1 = random.sample(list_no_late_3_e1, round(N1*RJ))
301
302 random_3_e1 = []
303 for i in range (len(list_no_late_3_e1)):
304     for j in range (len(list_bad_3_e1)):
305         if list_no_late_3_e1[i] == list_bad_3_e1[j]:
306             random_3_e1.append(i)
307
308 list_bad_3_e1 = []
309 for i in range (len(random_3_e1)):
310     list_bad_3_e1.append(list_no_late_3_e1[random_3_e1[i]])
311
312 list_no_bad_3_e1 = [x for x in list_no_late_3_e1 if x not in list_bad_3_e1]
313
314 no_random_3_e1 = []
315 for i in range (len(list_no_late_3_e1)):
316     for j in range (len(list_no_bad_3_e1)):
317         if list_no_late_3_e1[i] == list_no_bad_3_e1[j]:
318             no_random_3_e1.append(i)
319
320 normal_3_e1 = []
321 for i in range (len(random_3_e1)):
322     normal_3_e1.append(np.random.normal(random_3_e1[i], 5))
323
324 for i in range (len(normal_3_e1)):
325     list_bad_3_e1[i].append(normal_3_e1[i])
326
327 list_bad_3_e1 = sorted(list_bad_3_e1, key=lambda x: x[-1])
328
329 for i in range (len(normal_3_e1)):
330     del list_bad_3_e1[i][3]
331
332 normal_3_e1 = sorted(normal_3_e1)
333
334 list_shift_3_e1 = []
335 j=0;
336 for i in range (len(list_no_bad_3_e1)+len(list_bad_3_e1)):
337     if (len(list_bad_3_e1) != 0) & (j < len(no_random_3_e1)):
338         if normal_3_e1[0] < no_random_3_e1[j]:
339             list_shift_3_e1.append(list_bad_3_e1[0])
340             del list_bad_3_e1[0]
341             del normal_3_e1[0]
342         elif normal_3_e1[0] > no_random_3_e1[j]:
343             list_shift_3_e1.append(list_no_bad_3_e1[j])
344             j=j+1
345         elif j >= len(no_random_3_e1):
346             list_shift_3_e1.append(list_bad_3_e1[0])
347             del list_bad_3_e1[0]
348             del normal_3_e1[0]
349         else:
350             list_shift_3_e1.append(list_no_bad_3_e1[j])
351             j=j+1
352
353 list_3_e1 = list_shift_3_e1 + list_late_3_e1
```

```
354
355 class agent_3_e1:
356     def __init__(self, char, seat, bag, num_out, row, col):
357         self.char = char
358         self.seat = seat
359         self.bag = bag
360         self.num_out = num_out
361         self.row = row
362         self.col = col
363
364 passenger_3_e1 = []
365 for i in range(len(list_0_e1)):
366     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))
367
368 list_1_3_e2 = []
369 list_2_3_e2 = []
370 list_3_3_e2 = []
371 list_4_3_e2 = []
372
373 for i in range(len(list_0_e2)):
374     if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5):
375         list_1_3_e2.append(list_0_e2[i])
376 for i in range(len(list_0_e2)):
377     if (list_0_e2[i][0] == 9) & (1 <= list_0_e2[i][1] <= 14):
378         list_1_3_e2.append(list_0_e2[i])
379 for i in range(len(list_0_e2)):
380     if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4):
381         list_2_3_e2.append(list_0_e2[i])
382 for i in range(len(list_0_e2)):
383     if (list_0_e2[i][0] == 6) & (1 <= list_0_e2[i][0] <= 13):
384         list_2_3_e2.append(list_0_e2[i])
385 for i in range(len(list_0_e2)):
386     if (list_0_e2[i][0] == 8) & (1 <= list_0_e2[i][0] <= 14):
387         list_2_3_e2.append(list_0_e2[i])
388 for i in range(len(list_0_e2)):
389     if (list_0_e2[i][0] == 9) & (20 <= list_0_e2[i][1] <= 40):
390         list_3_3_e2.append(list_0_e2[i])
391 for i in range(len(list_0_e2)):
392     if (list_0_e2[i][0] == 6) & (20 <= list_0_e2[i][0] <= 40):
393         list_4_3_e2.append(list_0_e2[i])
394 for i in range(len(list_0_e2)):
395     if (list_0_e2[i][0] == 8) & (20 <= list_0_e2[i][0] <= 40):
396         list_4_3_e2.append(list_0_e2[i])
397
398 list_random_1_3_e2 = random.sample(list_1_3_e2, len(list_1_3_e2))
399 list_random_2_3_e2 = random.sample(list_2_3_e2, len(list_2_3_e2))
400 list_random_3_3_e2 = random.sample(list_3_3_e2, len(list_3_3_e2))
401 list_random_4_3_e2 = random.sample(list_4_3_e2, len(list_4_3_e2))
402 list_correct_3_e2 = list_random_1_3_e2 + list_random_2_3_e2 +
list_random_3_3_e2 + list_random_4_3_e2
403
404 list_late_3_e2 = random.sample(list_correct_3_e2, round(N2*RL))
405 list_no_late_3_e2 = [x for x in list_correct_3_e2 if x not in list_late_3_e2]
406
407 list_bad_3_e2 = random.sample(list_no_late_3_e2, round(N2*RJ))
408
409 random_3_e2 = []
410 for i in range(len(list_no_late_3_e2)):
411     for j in range(len(list_bad_3_e2)):
```

```
412         if list_no_late_3_e2[i] == list_bad_3_e2[j]:
413             random_3_e2.append(i)
414
415     list_bad_3_e2 = []
416     for i in range (len(random_3_e2)):
417         list_bad_3_e2.append(list_no_late_3_e2[random_3_e2[i]])
418
419     list_no_bad_3_e2 = [x for x in list_no_late_3_e2 if x not in list_bad_3_e2]
420
421     no_random_3_e2 = []
422     for i in range (len(list_no_late_3_e2)):
423         for j in range (len(list_no_bad_3_e2)):
424             if list_no_late_3_e2[i] == list_no_bad_3_e2[j]:
425                 no_random_3_e2.append(i)
426
427     normal_3_e2 = []
428     for i in range (len(random_3_e2)):
429         normal_3_e2.append(np.random.normal(random_3_e2[i], 5))
430
431     for i in range (len(normal_3_e2)):
432         list_bad_3_e2[i].append(normal_3_e2[i])
433
434     list_bad_3_e2 = sorted(list_bad_3_e2, key=lambda x: x[-1])
435
436     for i in range (len(normal_3_e2)):
437         del list_bad_3_e2[i][3]
438
439     normal_3_e2 = sorted(normal_3_e2)
440
441     list_shift_3_e2 = []
442     j=0;
443     for i in range (len(list_no_bad_3_e2)+len(list_bad_3_e2)):
444         if (len(list_bad_3_e2) != 0) & (j < len(no_random_3_e2)):
445             if normal_3_e2[0] < no_random_3_e2[j]:
446                 list_shift_3_e2.append(list_bad_3_e2[0])
447                 del list_bad_3_e2[0]
448                 del normal_3_e2[0]
449             elif normal_3_e2[0] > no_random_3_e2[j]:
450                 list_shift_3_e2.append(list_no_bad_3_e2[j])
451                 j=j+1
452             elif j >= len(no_random_3_e2):
453                 list_shift_3_e2.append(list_bad_3_e2[0])
454                 del list_bad_3_e2[0]
455                 del normal_3_e2[0]
456             else:
457                 list_shift_3_e2.append(list_no_bad_3_e2[j])
458                 j=j+1
459
460     list_3_e2 = list_shift_3_e2 + list_late_3_e2
461
462     class agent_3_e2:
463         def __init__(self, char, seat, bag, num_out, row, col):
464             self.char = char
465             self.seat = seat
466             self.bag = bag
467             self.num_out = num_out
468             self.row = row
469             self.col = col
470
471     passenger_3_e2 = []
```

```

472     for i in range(len(list_0_e2)):
473         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))
474
475     #Case4
476     list_1_4_e1 = []
477     list_2_4_e1 = []
478     list_3_4_e1 = []
479     list_4_4_e1 = []
480
481     for i in range (len(list_0_e1)):
482         if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5):
483             list_1_4_e1.append(list_0_e1[i])
484     for i in range (len(list_0_e1)):
485         if (list_0_e1[i][0] == 9) & (1 <= list_0_e1[i][1] <= 14):
486             list_3_4_e1.append(list_0_e1[i])
487     for i in range (len(list_0_e1)):
488         if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4):
489             list_2_4_e1.append(list_0_e1[i])
490     for i in range (len(list_0_e1)):
491         if (list_0_e1[i][0] == 6) & (1 <= list_0_e1[i][0] <= 13):
492             list_4_4_e1.append(list_0_e1[i])
493     for i in range (len(list_0_e1)):
494         if (list_0_e1[i][0] == 8) & (1 <= list_0_e1[i][0] <= 14):
495             list_4_4_e1.append(list_0_e1[i])
496     for i in range (len(list_0_e1)):
497         if (list_0_e1[i][0] == 9) & (20 <= list_0_e1[i][1] <= 40):
498             list_1_4_e1.append(list_0_e1[i])
499     for i in range (len(list_0_e1)):
500         if (list_0_e1[i][0] == 6) & (20 <= list_0_e1[i][0] <= 40):
501             list_2_4_e1.append(list_0_e1[i])
502     for i in range (len(list_0_e1)):
503         if (list_0_e1[i][0] == 8) & (20 <= list_0_e1[i][0] <= 40):
504             list_2_4_e1.append(list_0_e1[i])
505
506     list_random_1_4_e1 = random.sample(list_1_4_e1, len(list_1_4_e1))
507     list_random_2_4_e1 = random.sample(list_2_4_e1, len(list_2_4_e1))
508     list_random_3_4_e1 = random.sample(list_3_4_e1, len(list_3_4_e1))
509     list_random_4_4_e1 = random.sample(list_4_4_e1, len(list_4_4_e1))
510     list_correct_4_e1 = list_random_1_4_e1 + list_random_2_4_e1 +
list_random_3_4_e1 + list_random_4_4_e1
511
512     list_late_4_e1 = random.sample(list_correct_4_e1, round(N1*RL))
513     list_no_late_4_e1 = [x for x in list_correct_4_e1 if x not in list_late_4_e1]
514
515     list_bad_4_e1 = random.sample(list_no_late_4_e1, round(N1*RJ))
516
517     random_4_e1 = []
518     for i in range (len(list_no_late_4_e1)):
519         for j in range (len(list_bad_4_e1)):
520             if list_no_late_4_e1[i] == list_bad_4_e1[j]:
521                 random_4_e1.append(i)
522
523     list_bad_4_e1 = []
524     for i in range (len(random_4_e1)):
525         list_bad_4_e1.append(list_no_late_4_e1[random_4_e1[i]])
526
527     list_no_bad_4_e1 = [x for x in list_no_late_4_e1 if x not in list_bad_4_e1]
528
529     no_random_4_e1 = []

```



```
530     for i in range (len(list_no_late_4_e1)):
531         for j in range (len(list_no_bad_4_e1)):
532             if list_no_late_4_e1[i] == list_no_bad_4_e1[j]:
533                 no_random_4_e1.append(i)
534
535     normal_4_e1 = []
536     for i in range (len(random_4_e1)):
537         normal_4_e1.append(np.random.normal(random_4_e1[i], 5))
538
539     for i in range (len(normal_4_e1)):
540         list_bad_4_e1[i].append(normal_4_e1[i])
541
542     list_bad_4_e1 = sorted(list_bad_4_e1, key=lambda x: x[-1])
543
544     for i in range (len(normal_4_e1)):
545         del list_bad_4_e1[i][3]
546
547     normal_4_e1 = sorted(normal_4_e1)
548
549     list_shift_4_e1 = []
550     j=0;
551     for i in range (len(list_no_bad_4_e1)+len(list_bad_4_e1)):
552         if (len(list_bad_4_e1) != 0) & (j < len(no_random_4_e1)):
553             if normal_4_e1[0] < no_random_4_e1[j]:
554                 list_shift_4_e1.append(list_bad_4_e1[0])
555                 del list_bad_4_e1[0]
556                 del normal_4_e1[0]
557             elif normal_4_e1[0] > no_random_4_e1[j]:
558                 list_shift_4_e1.append(list_no_bad_4_e1[j])
559                 j=j+1
560             elif j >= len(no_random_4_e1):
561                 list_shift_4_e1.append(list_bad_4_e1[0])
562                 del list_bad_4_e1[0]
563                 del normal_4_e1[0]
564             else:
565                 list_shift_4_e1.append(list_no_bad_4_e1[j])
566                 j=j+1
567
568     list_4_e1 = list_shift_4_e1 + list_late_4_e1
569
570     class agent_4_e1:
571         def __init__(self, char, seat, bag, num_out, row, col):
572             self.char = char
573             self.seat = seat
574             self.bag = bag
575             self.num_out = num_out
576             self.row = row
577             self.col = col
578
579     passenger_4_e1 = []
580     for i in range(len(list_0_e1)):
581         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))
582
583     list_1_4_e2 = []
584     list_2_4_e2 = []
585     list_3_4_e2 = []
586     list_4_4_e2 = []
587
588     for i in range (len(list_0_e2)):
```

```

589         if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5):
590             list_1_4_e2.append(list_0_e2[i])
591     for i in range (len(list_0_e2)):
592         if (list_0_e2[i][0] == 9) & (1 <= list_0_e2[i][1] <= 14):
593             list_3_4_e2.append(list_0_e2[i])
594     for i in range (len(list_0_e2)):
595         if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4):
596             list_2_4_e2.append(list_0_e2[i])
597     for i in range (len(list_0_e2)):
598         if (list_0_e2[i][0] == 6) & (1 <= list_0_e2[i][0] <= 13):
599             list_4_4_e2.append(list_0_e2[i])
600     for i in range (len(list_0_e2)):
601         if (list_0_e2[i][0] == 8) & (1 <= list_0_e2[i][0] <= 14):
602             list_4_4_e2.append(list_0_e2[i])
603     for i in range (len(list_0_e2)):
604         if (list_0_e2[i][0] == 9) & (20 <= list_0_e2[i][1] <= 40):
605             list_1_4_e2.append(list_0_e2[i])
606     for i in range (len(list_0_e2)):
607         if (list_0_e2[i][0] == 6) & (20 <= list_0_e2[i][0] <= 40):
608             list_2_4_e2.append(list_0_e2[i])
609     for i in range (len(list_0_e2)):
610         if (list_0_e2[i][0] == 8) & (20 <= list_0_e2[i][0] <= 40):
611             list_2_4_e2.append(list_0_e2[i])
612
613     list_random_1_4_e2 = random.sample(list_1_4_e2, len(list_1_4_e2))
614     list_random_2_4_e2 = random.sample(list_2_4_e2, len(list_2_4_e2))
615     list_random_3_4_e2 = random.sample(list_3_4_e2, len(list_3_4_e2))
616     list_random_4_4_e2 = random.sample(list_4_4_e2, len(list_4_4_e2))
617     list_correct_4_e2 = list_random_1_4_e2 + list_random_2_4_e2 +
list_random_3_4_e2 + list_random_4_4_e2
618
619     list_late_4_e2 = random.sample(list_correct_4_e2, round(N2*RL))
620     list_no_late_4_e2 = [x for x in list_correct_4_e2 if x not in list_late_4_e2]
621
622     list_bad_4_e2 = random.sample(list_no_late_4_e2, round(N2*RJ))
623
624     random_4_e2 = []
625     for i in range (len(list_no_late_4_e2)):
626         for j in range (len(list_bad_4_e2)):
627             if list_no_late_4_e2[i] == list_bad_4_e2[j]:
628                 random_4_e2.append(i)
629
630     list_bad_4_e2 = []
631     for i in range (len(random_4_e2)):
632         list_bad_4_e2.append(list_no_late_4_e2[random_4_e2[i]])
633
634     list_no_bad_4_e2 = [x for x in list_no_late_4_e2 if x not in list_bad_4_e2]
635
636     no_random_4_e2 = []
637     for i in range (len(list_no_late_4_e2)):
638         for j in range (len(list_no_bad_4_e2)):
639             if list_no_late_4_e2[i] == list_no_bad_4_e2[j]:
640                 no_random_4_e2.append(i)
641
642     normal_4_e2 = []
643     for i in range (len(random_4_e2)):
644         normal_4_e2.append(np.random.normal(random_4_e2[i], 5))
645
646     for i in range (len(normal_4_e2)):
647         list_bad_4_e2[i].append(normal_4_e2[i])

```

```
648
649 list_bad_4_e2 = sorted(list_bad_4_e2, key=lambda x: x[-1])
650
651 for i in range (len(normal_4_e2)):
652     del list_bad_4_e2[i][3]
653
654 normal_4_e2 = sorted(normal_4_e2)
655
656 list_shift_4_e2 = []
657 j=0;
658 for i in range (len(list_no_bad_4_e2)+len(list_bad_4_e2)):
659     if (len(list_bad_4_e2) != 0) & (j < len(no_random_4_e2)):
660         if normal_4_e2[0] < no_random_4_e2[j]:
661             list_shift_4_e2.append(list_bad_4_e2[0])
662             del list_bad_4_e2[0]
663             del normal_4_e2[0]
664         elif normal_4_e2[0] > no_random_4_e2[j]:
665             list_shift_4_e2.append(list_no_bad_4_e2[j])
666             j=j+1
667     elif j >= len(no_random_4_e2):
668         list_shift_4_e2.append(list_bad_4_e2[0])
669         del list_bad_4_e2[0]
670         del normal_4_e2[0]
671     else:
672         list_shift_4_e2.append(list_no_bad_4_e2[j])
673         j=j+1
674
675 list_4_e2 = list_shift_4_e2 + list_late_4_e2
676
677 class agent_4_e2:
678     def __init__(self, char, seat, bag, num_out, row, col):
679         self.char = char
680         self.seat = seat
681         self.bag = bag
682         self.num_out = num_out
683         self.row = row
684         self.col = col
685
686 passenger_4_e2 = []
687 for i in range(len(list_0_e2)):
688     passenger_4_e2.append(agent_4_e2(list_4_e2[i][0], list_4_e2[i][1],
689 list_4_e2[i][2], 0, 0, 0))
690
691 #Case5
692 list_1_5_e1 = []
693 list_2_5_e1 = []
694
695 for i in range (len(list_0_e1)):
696     if (list_0_e1[i][0] == 1) | (list_0_e1[i][0] == 5) | (list_0_e1[i][0] ==
697 9):
698         list_1_5_e1.append(list_0_e1[i])
699     for i in range (len(list_0_e1)):
700         if (list_0_e1[i][0] == 2) | (list_0_e1[i][0] == 4) | (list_0_e1[i][0] ==
701 6) | (list_0_e1[i][0] == 8):
702             list_2_5_e1.append(list_0_e1[i])
703
704 list_random_1_5_e1 = random.sample(list_1_5_e1, len(list_1_5_e1))
705 list_random_2_5_e1 = random.sample(list_2_5_e1, len(list_2_5_e1))
706 list_correct_5_e1 = list_random_1_5_e1 + list_random_2_5_e1
707
```

```

705 list_late_5_e1 = random.sample(list_correct_5_e1, round(N1*RL))
706 list_no_late_5_e1 = [x for x in list_correct_5_e1 if x not in list_late_5_e1]
707
708 list_bad_5_e1 = random.sample(list_no_late_5_e1, round(N1*RJ))
709
710 random_5_e1 = []
711 for i in range (len(list_no_late_5_e1)):
712     for j in range (len(list_bad_5_e1)):
713         if list_no_late_5_e1[i] == list_bad_5_e1[j]:
714             random_5_e1.append(i)
715
716 list_bad_5_e1 = []
717 for i in range (len(random_5_e1)):
718     list_bad_5_e1.append(list_no_late_5_e1[random_5_e1[i]])
719
720 list_no_bad_5_e1 = [x for x in list_no_late_5_e1 if x not in list_bad_5_e1]
721
722 no_random_5_e1 = []
723 for i in range (len(list_no_late_5_e1)):
724     for j in range (len(list_no_bad_5_e1)):
725         if list_no_late_5_e1[i] == list_no_bad_5_e1[j]:
726             no_random_5_e1.append(i)
727
728 normal_5_e1 = []
729 for i in range (len(random_5_e1)):
730     normal_5_e1.append(np.random.normal(random_5_e1[i], 5))
731
732 for i in range (len(normal_5_e1)):
733     list_bad_5_e1[i].append(normal_5_e1[i])
734
735 list_bad_5_e1 = sorted(list_bad_5_e1, key=lambda x: x[-1])
736
737 for i in range (len(normal_5_e1)):
738     del list_bad_5_e1[i][3]
739
740 normal_5_e1 = sorted(normal_5_e1)
741
742 list_shift_5_e1 = []
743 j=0;
744 for i in range (len(list_no_bad_5_e1)+len(list_bad_5_e1)):
745     if (len(list_bad_5_e1) != 0) & (j < len(no_random_5_e1)):
746         if normal_5_e1[0] < no_random_5_e1[j]:
747             list_shift_5_e1.append(list_bad_5_e1[0])
748             del list_bad_5_e1[0]
749             del normal_5_e1[0]
750         elif normal_5_e1[0] > no_random_5_e1[j]:
751             list_shift_5_e1.append(list_no_bad_5_e1[j])
752             j=j+1
753         elif j >= len(no_random_5_e1):
754             list_shift_5_e1.append(list_bad_5_e1[0])
755             del list_bad_5_e1[0]
756             del normal_5_e1[0]
757         else:
758             list_shift_5_e1.append(list_no_bad_5_e1[j])
759             j=j+1
760
761 list_5_e1 = list_shift_5_e1 + list_late_5_e1
762
763 class agent_5_e1:
764     def __init__(self, char, seat, bag, num_out, row, col):

```

```
765         self.char = char
766         self.seat = seat
767         self.bag = bag
768         self.num_out = num_out
769         self.row = row
770         self.col = col
771
772     passenger_5_e1 = []
773     for i in range(len(list_0_e1)):
774         passenger_5_e1.append(agent_5_e1(list_5_e1[i][0], list_5_e1[i][1],
775         list_5_e1[i][2], 0, 0, 0))
776
777     list_1_5_e2 = []
778     list_2_5_e2 = []
779
780     for i in range(len(list_0_e2)):
781         if (list_0_e2[i][0] == 1) | (list_0_e2[i][0] == 5) | (list_0_e2[i][0] ==
782         9):
783             list_1_5_e2.append(list_0_e2[i])
784     for i in range(len(list_0_e2)):
785         if (list_0_e2[i][0] == 2) | (list_0_e2[i][0] == 4) | (list_0_e2[i][0] ==
786         6) | (list_0_e2[i][0] == 8):
787             list_2_5_e2.append(list_0_e2[i])
788
789     list_random_1_5_e2 = random.sample(list_1_5_e2, len(list_1_5_e2))
790     list_random_2_5_e2 = random.sample(list_2_5_e2, len(list_2_5_e2))
791     list_correct_5_e2 = list_random_1_5_e2 + list_random_2_5_e2
792
793     list_late_5_e2 = random.sample(list_correct_5_e2, round(N2*RL))
794     list_no_late_5_e2 = [x for x in list_correct_5_e2 if x not in list_late_5_e2]
795
796     list_bad_5_e2 = random.sample(list_no_late_5_e2, round(N2*RJ))
797
798     random_5_e2 = []
799     for i in range(len(list_no_late_5_e2)):
800         for j in range(len(list_bad_5_e2)):
801             if list_no_late_5_e2[i] == list_bad_5_e2[j]:
802                 random_5_e2.append(i)
803
804     list_bad_5_e2 = []
805     for i in range(len(random_5_e2)):
806         list_bad_5_e2.append(list_no_late_5_e2[random_5_e2[i]])
807
808     list_no_bad_5_e2 = [x for x in list_no_late_5_e2 if x not in list_bad_5_e2]
809
810     no_random_5_e2 = []
811     for i in range(len(list_no_late_5_e2)):
812         for j in range(len(list_no_bad_5_e2)):
813             if list_no_late_5_e2[i] == list_no_bad_5_e2[j]:
814                 no_random_5_e2.append(i)
815
816     normal_5_e2 = []
817     for i in range(len(random_5_e2)):
818         normal_5_e2.append(np.random.normal(random_5_e2[i], 5))
819
820     for i in range(len(normal_5_e2)):
821         list_bad_5_e2[i].append(normal_5_e2[i])
822
823     list_bad_5_e2 = sorted(list_bad_5_e2, key=lambda x: x[-1])
```

```
822     for i in range (len(normal_5_e2)):
823         del list_bad_5_e2[i][3]
824
825     normal_5_e2 = sorted(normal_5_e2)
826
827     list_shift_5_e2 = []
828     j=0;
829     for i in range (len(list_no_bad_5_e2)+len(list_bad_5_e2)):
830         if (len(list_bad_5_e2) != 0) & (j < len(no_random_5_e2)):
831             if normal_5_e2[0] < no_random_5_e2[j]:
832                 list_shift_5_e2.append(list_bad_5_e2[0])
833                 del list_bad_5_e2[0]
834                 del normal_5_e2[0]
835             elif normal_5_e2[0] > no_random_5_e2[j]:
836                 list_shift_5_e2.append(list_no_bad_5_e2[j])
837                 j=j+1
838             elif j >= len(no_random_5_e2):
839                 list_shift_5_e2.append(list_bad_5_e2[0])
840                 del list_bad_5_e2[0]
841                 del normal_5_e2[0]
842             else:
843                 list_shift_5_e2.append(list_no_bad_5_e2[j])
844                 j=j+1
845
846     list_5_e2 = list_shift_5_e2 + list_late_5_e2
847
848     class agent_5_e2:
849         def __init__(self, char, seat, bag, num_out, row, col):
850             self.char = char
851             self.seat = seat
852             self.bag = bag
853             self.num_out = num_out
854             self.row = row
855             self.col = col
856
857     passenger_5_e2 = []
858     for i in range(len(list_0_e2)):
859         passenger_5_e2.append(agent_5_e2(list_5_e2[i][0], list_5_e2[i][1],
860 list_5_e2[i][2], 0, 0, 0))
861
862     class person:
863         def __init__(self, char, seat, bag, num_out, t_1, t_2, check):
864             self.char = char
865             self.seat = seat
866             self.bag = bag
867             self.num_out = num_out
868             self.t_1 = t_1
869             self.t_2 = t_2
870             self.check = check
871
872     list_pass = [[person(0,0,0,0,0,0,0) for i in range(0,41)] for j in range
873 (0,10)]
874
875     class grid:
876         def __init__(self, type, value, pass_char, pass_seat):
877             self.type = type
878             # -1 -> block
879             # 0 -> cabin
880             # 1 -> queue
881             # 2 -> aisle
```

```
880     # 3 -> seat
881     self.value = value
882     # 0 -> available
883     # 1 -> passenger
884     self.pass_char = pass_char
885     self.pass_seat = pass_seat
886
887 plane = [[grid(-1,0,0,0) for i in range(0,42)] for j in range(0,250)]
888
889 for i in range(1,3):
890     for j in range(1,15):
891         plane[i][j].type = 3
892
893 for i in range(4,7):
894     for j in range(1,14):
895         plane[i][j].type = 3
896
897 for i in range(8,10):
898     for j in range(1,15):
899         plane[i][j].type = 3
900
901 for i in range(1,3):
902     for j in range(20,41):
903         plane[i][j].type = 3
904
905 for i in range(4,7):
906     for j in range(20,41):
907         plane[i][j].type = 3
908
909 for i in range(8,10):
910     for j in range(20,41):
911         plane[i][j].type = 3
912
913 for i in range(3,250):
914     plane[i][0].type = 1
915     plane[i][41].type = 1
916
917 q = [3,7]
918 for i in q:
919     for j in range(0,17):
920         plane[i][j].type = 2
921     for j in range(18,42):
922         plane[i][j].type = 2
923
924 for i in range(1,10):
925     plane[i][17].type = 0
926
927 def C1(passenger_1_e1,passenger_1_e2):
928     for i in range(len(passenger_1_e1)):
929         list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].char =
passenger_1_e1[i].char
930         list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].seat =
passenger_1_e1[i].seat
931         list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].bag =
passenger_1_e1[i].bag
932         list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].num_out =
-1
933         list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].t_1 = -1
934         list_pass[passenger_1_e1[i].char][passenger_1_e1[i].seat].t_2 = -1
935         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
```

```

1028         plane[8+i][41].pass_char = passenger_4_e2[i].char
1029         plane[8+i][41].pass_seat = passenger_4_e2[i].seat
1030
1031     def C5(passenger_5_e1,passenger_5_e2):
1032         for i in range(len(passenger_5_e1)):
1033             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].char =
passenger_5_e1[i].char
1034             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].seat =
passenger_5_e1[i].seat
1035             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].bag =
passenger_5_e1[i].bag
1036             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].num_out =
-1
1037             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].t_1 = -1
1038             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].t_2 = -1
1039             list_pass[passenger_5_e1[i].char][passenger_5_e1[i].seat].check = 0
1040         for i in range(0,len(passenger_5_e1)):
1041             plane[8+i][0].value = 1
1042             plane[8+i][0].pass_char = passenger_5_e1[i].char
1043             plane[8+i][0].pass_seat = passenger_5_e1[i].seat
1044         for i in range(len(passenger_5_e2)):
1045             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].char =
passenger_5_e2[i].char
1046             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].seat =
passenger_5_e2[i].seat
1047             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].bag =
passenger_5_e2[i].bag
1048             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].num_out =
-1
1049             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].t_1 = -1
1050             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].t_2 = -1
1051             list_pass[passenger_5_e2[i].char][passenger_5_e2[i].seat].check = 0
1052         for i in range(0,len(passenger_5_e2)):
1053             plane[8+i][41].value = 1
1054             plane[8+i][41].pass_char = passenger_5_e2[i].char
1055             plane[8+i][41].pass_seat = passenger_5_e2[i].seat
1056
1057     if(case==1):
1058         C1(passenger_1_e1,passenger_1_e2)
1059     if(case==2):
1060         C2(passenger_2_e1,passenger_2_e2)
1061     if(case==3):
1062         C3(passenger_3_e1,passenger_3_e2)
1063     if(case==4):
1064         C4(passenger_4_e1,passenger_4_e2)
1065     if(case==5):
1066         C5(passenger_5_e1,passenger_5_e2)
1067
1068     a = [[0 for i in range(0,42)] for i in range(0,250)]
1069
1070     for i in range(0,250):
1071         for j in range(0,42):
1072             #print(plane[i][j].value)
1073             a[i][j] = plane[i][j].type
1074
1075     time = 0
1076
1077     """print(len(passenger_1_e1))
1078     print(len(passenger_1_e2))
1079

```

```

1080 plt.figure('time'+str(time))
1081 im = plt.imshow(a)
1082 ax = plt.gca()
1083 #ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
1084 #ax.set_yticks(np.arange(-.5, 250, 1), minor=True)
1085 ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
1086 plt.savefig("figure_3.png")
1087 plt.show()"""
1088
1089 def check_pass(plane):
1090     check = 0
1091     for w in range(1,10):
1092         for z in range(1,41):
1093             if(plane[w][z].type == 3 and plane[w][z].value == 1):
1094                 check+=1
1095     return check
1096
1097 time = 0
1098 while(1):
1099     #check
1100     check = check_pass(plane)
1101     if(check==N):
1102         #print(time)
1103         break
1104     time+=1
1105
1106
1107 left_1 = [2,6]
1108 for i in left_1:
1109     for j in range(1,41):
1110         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):
1111             p1r = plane[i][j].pass_char
1112             p1c = plane[i][j].pass_seat
1113             plane[i-1][j].pass_char = p1r
1114             plane[i-1][j].pass_seat = p1c
1115             plane[i-1][j].value = 1
1116             plane[i][j].pass_char = 0
1117             plane[i][j].pass_seat = 0
1118             plane[i][j].value = 0
1119
1120     i = 8
1121     for j in range(1,41):
1122         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):
1123             p1r = plane[i][j].pass_char
1124             p1c = plane[i][j].pass_seat
1125             plane[i+1][j].pass_char = p1r
1126             plane[i+1][j].pass_seat = p1c
1127             plane[i+1][j].value = 1
1128             plane[i][j].pass_char = 0
1129             plane[i][j].pass_seat = 0
1130             plane[i][j].value = 0
1131
1132 #front
1133 aisle = [3,7]
1134 for i in aisle:
1135     for j in reversed(range(0,17)):
1136         if(plane[i][j].pass_char<5 and i==7):
1137             continue

```

```

1138         if(plane[i][j].value==0):
1139             continue
1140         if(plane[i][j].pass_seat==j):
1141             if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2>0):
1142                 list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
t_2-=1
1143                 continue
1144             else:
1145                 p1r = plane[i][j].pass_char
1146                 p1c = plane[i][j].pass_seat
1147                 if(plane[i][j].pass_char<i):
1148                     if(plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].value == 1):
1149                         p2r = plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_char
1150                         p2c = plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_seat
1151                         plane[i-1][j].pass_char = p2r
1152                         plane[i-1][j].pass_seat = p2c
1153                         plane[i-1][j].value = 1
1154                         plane[i-2][j].pass_char = p1r
1155                         plane[i-2][j].pass_seat = p1c
1156                         plane[i-2][j].value = 1
1157                         plane[i][j].pass_char = 0
1158                         plane[i][j].pass_seat = 0
1159                         plane[i][j].value = 0
1160                         continue
1161                     plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_char = p1r
1162                     plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_seat = p1c
1163                     plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].value = 1
1164                 else:
1165                     if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value == 1):
1166                         p2r = plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_char
1167                         p2c = plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_seat
1168                         plane[i+1][j].pass_char = p2r
1169                         plane[i+1][j].pass_seat = p2c
1170                         plane[i+1][j].value = 1
1171                         plane[i+2][j].pass_char = p1r
1172                         plane[i+2][j].pass_seat = p1c
1173                         plane[i+2][j].value = 1
1174                         plane[i][j].pass_char = 0
1175                         plane[i][j].pass_seat = 0
1176                         plane[i][j].value = 0
1177                         continue
1178                     if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value == 1):
1179                         continue
1180                     plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_char = p1r
1181                     plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_seat = p1c
1182                     plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value = 1

```

```

1183         plane[i][j].pass_char = 0
1184         plane[i][j].pass_seat = 0
1185         plane[i][j].value = 0
1186         if(plane[i][j].pass_seat>j):
1187             if(plane[i][j].pass_seat-j==1):
1188                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
1189 ] .bag>0):
1190                     list_pass[plane[i][j].pass_char][plane[i][j].
1191 pass_seat].bag-=1
1192                     continue
1193                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
1194 ] .check==0):
1195                     sum = 0
1196                     if(plane[i][j].pass_char<i):
1197                         for m in range(i-2,i):
1198                             if(plane[m][j+1].value == 1 and plane[m][j
1199 +1].pass_char!=m):
1200                                 sum+=1
1201                             else:
1202                                 for m in range(i+1,i+3):
1203                                     if(plane[m][j+1].value == 1 and plane[m][j
1204 +1].pass_char!=m):
1205                                         sum+=1
1206                                     if(sum!=0):
1207                                         continue
1208                                     sum = 0
1209                                     if(plane[i][j].pass_char<i):
1210                                         for m in reversed(range(plane[i][j].pass_char+1,i
1211 ))):
1212                                             if(plane[m][j+1].value == 1):
1213                                                 sum+=1
1214                                             else:
1215                                                 for m in range(i+1,plane[i][j].pass_char):
1216                                                     if(plane[m][j+1].value == 1):
1217                                                         sum+=1
1218                                                     list_pass[plane[i][j].pass_char][plane[i][j].
1219 pass_seat].num_out = sum
1220                                     mov = 0
1221                                     for n in range(0,list_pass[plane[i][j].pass_char][
1222 plane[i][j].pass_seat].num_out+1):
1223                                         if(plane[i][j+1+n].value==1):
1224                                             mov = 1
1225                                         if(mov == 1):
1226                                             continue
1227                                         list_pass[plane[i][j].pass_char][plane[i][j].
1228 pass_seat].check = 1
1229                                         if(plane[i][j].pass_char<i):
1230                                             if(plane[i][j].pass_char==i-1):
1231                                                 list_pass[plane[i][j].pass_char][plane[i][j].
1232 pass_seat].t_1 = 0
1233                                                 list_pass[plane[i][j].pass_char][plane[i][j].
1234 pass_seat].t_2 = 0
1235                                             if(plane[i][j].pass_char==i-2):
1236                                                 if(plane[i-1][j+1].value==1):
1237                                                     list_pass[plane[i][j].pass_char][plane[i
1238 ][j].pass_seat].t_1 = 1
1239                                                     list_pass[plane[i][j].pass_char][plane[i
1240 ][j].pass_seat].t_2 = 2
1241                                             else:
1242                                                 list_pass[plane[i][j].pass_char][plane[i

```

```

1230 ][j].pass_seat).t_1 = 0
1231                                     list_pass[plane[i][j].pass_char][plane[i]
1232 ][j].pass_seat).t_2 = 0
1233                                     else:
1234                                     if(plane[i][j].pass_char==i+1):
1235                                     list_pass[plane[i][j].pass_char][plane[i][j].
1236 pass_seat].t_1 = 0
1237                                     list_pass[plane[i][j].pass_char][plane[i][j].
1238 pass_seat].t_2 = 0
1239                                     if(plane[i][j].pass_char==i+2):
1240                                     if(plane[i+1][j+1].value==1):
1241                                     list_pass[plane[i][j].pass_char][plane[i]
1242 ][j].pass_seat).t_1 = 1
1243                                     list_pass[plane[i][j].pass_char][plane[i]
1244 ][j].pass_seat).t_2 = 2
1245                                     else:
1246                                     list_pass[plane[i][j].pass_char][plane[i]
1247 ][j].pass_seat).t_1 = 0
1248                                     list_pass[plane[i][j].pass_char][plane[i]
1249 ][j].pass_seat).t_2 = 0
1250                                     if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
1251 ].check == 1):
1252                                     if(list_pass[plane[i][j].pass_char][plane[i][j].
1253 pass_seat].t_1>0):
1254                                     list_pass[plane[i][j].pass_char][plane[i][j].
1255 pass_seat].t_1--1
1256                                     else:
1257                                     if(plane[i][j+1].value==1):
1258                                     continue
1259                                     p1r = plane[i][j].pass_char
1260                                     p1c = plane[i][j].pass_seat
1261                                     plane[i][j+1].pass_char = p1r
1262                                     plane[i][j+1].pass_seat = p1c
1263                                     plane[i][j+1].value = 1
1264                                     plane[i][j].pass_char = 0
1265                                     plane[i][j].pass_seat = 0
1266                                     plane[i][j].value = 0
1267                                     else:
1268                                     if(plane[i][j+1].value==0):
1269                                     p1r = plane[i][j].pass_char
1270                                     p1c = plane[i][j].pass_seat
1271                                     plane[i][j+1].pass_char = p1r
1272                                     plane[i][j+1].pass_seat = p1c
1273                                     plane[i][j+1].value = 1
1274                                     plane[i][j].pass_char = 0
1275                                     plane[i][j].pass_seat = 0
1276                                     plane[i][j].value = 0
1277
1278 #back
1279 for i in aisle:
1280     for j in range(18,42):
1281         if(plane[i][j].pass_char<5 and i==7):
1282             continue
1283         if(plane[i][j].value==0):
1284             continue
1285         if(plane[i][j].pass_seat==j):
1286             if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
1287 t_2>0):
1288                 list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
1289 t_2--1

```

```

1277         continue
1278     else:
1279         p1r = plane[i][j].pass_char
1280         p1c = plane[i][j].pass_seat
1281         if(plane[i][j].pass_char<i):
1282             if(plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].value == 1):
1283                 p2r = plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_char
1284                 p2c = plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_seat
1285                 plane[i-1][j].pass_char = p2r
1286                 plane[i-1][j].pass_seat = p2c
1287                 plane[i-1][j].value = 1
1288                 plane[i-2][j].pass_char = p1r
1289                 plane[i-2][j].pass_seat = p1c
1290                 plane[i-2][j].value = 1
1291                 plane[i][j].pass_char = 0
1292                 plane[i][j].pass_seat = 0
1293                 plane[i][j].value = 0
1294                 continue
1295                 plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_char = p1r
1296                 plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].pass_seat = p1c
1297                 plane[i-list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out-1][j].value = 1
1298             else:
1299                 if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value == 1):
1300                     p2r = plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_char
1301                     p2c = plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_seat
1302                     plane[i+1][j].pass_char = p2r
1303                     plane[i+1][j].pass_seat = p2c
1304                     plane[i+1][j].value = 1
1305                     plane[i+2][j].pass_char = p1r
1306                     plane[i+2][j].pass_seat = p1c
1307                     plane[i+2][j].value = 1
1308                     plane[i][j].pass_char = 0
1309                     plane[i][j].pass_seat = 0
1310                     plane[i][j].value = 0
1311                     continue
1312                 if(plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value == 1):
1313                     continue
1314                 plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_char = p1r
1315                 plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].pass_seat = p1c
1316                 plane[i+list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].num_out+1][j].value = 1
1317                 plane[i][j].pass_char = 0
1318                 plane[i][j].pass_seat = 0
1319                 plane[i][j].value = 0
1320             if(plane[i][j].pass_seat<j):
1321                 if(j-plane[i][j].pass_seat-j==1):
1322                     if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag>0):

```

```

1323         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].bag -= 1
1324         continue
1325     if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].check==0):
1326         sum = 0
1327         if(plane[i][j].pass_char<i):
1328             for m in range(i-2,i):
1329                 if(plane[m][j-1].value == 1 and plane[m][j
-1].pass_char!=m):
1330                     sum+=1
1331         else:
1332             for m in range(i+1,i+3):
1333                 if(plane[m][j-1].value == 1 and plane[m][j
-1].pass_char!=m):
1334                     sum+=1
1335         if(sum!=0):
1336             continue
1337         sum = 0
1338         if(plane[i][j].pass_char<i):
1339             for m in reversed(range(plane[i][j].pass_char+1,i
)):
1340                 if(plane[m][j-1].value == 1):
1341                     sum+=1
1342         else:
1343             for m in range(i+1,plane[i][j].pass_char):
1344                 if(plane[m][j-1].value == 1):
1345                     sum+=1
1346         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].num_out = sum
1347         mov = 0
1348         for n in range(0,list_pass[plane[i][j].pass_char][
plane[i][j].pass_seat].num_out+1):
1349             if(plane[i][j-1-n].value==1):
1350                 mov = 1
1351         if(mov == 1):
1352             continue
1353         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].check = 1
1354         if(plane[i][j].pass_char<i):
1355             if(plane[i][j].pass_char==i-1):
1356                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0
1357                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
1358             if(plane[i][j].pass_char==i-2):
1359                 if(plane[i-1][j-1].value==1):
1360                     list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
1361                     list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
1362             else:
1363                 list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
1364                 list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
1365         else:
1366             if(plane[i][j].pass_char==i+1):
1367                 list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1 = 0

```



```

1368         list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_2 = 0
1369         if(plane[i][j].pass_char==i+2):
1370             if(plane[i+1][j-1].value==1):
1371                 list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 1
1372                 list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 2
1373             else:
1374                 list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_1 = 0
1375                 list_pass[plane[i][j].pass_char][plane[i
][j].pass_seat].t_2 = 0
1376             if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat
].check == 1):
1377                 if(list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1>0):
1378                     list_pass[plane[i][j].pass_char][plane[i][j].
pass_seat].t_1-=1
1379                 else:
1380                     if(plane[i][j-1].value==1):
1381                         continue
1382                     p1r = plane[i][j].pass_char
1383                     p1c = plane[i][j].pass_seat
1384                     plane[i][j-1].pass_char = p1r
1385                     plane[i][j-1].pass_seat = p1c
1386                     plane[i][j-1].value = 1
1387                     plane[i][j].pass_char = 0
1388                     plane[i][j].pass_seat = 0
1389                     plane[i][j].value = 0
1390             else:
1391                 if(plane[i][j-1].value==0):
1392                     p1r = plane[i][j].pass_char
1393                     p1c = plane[i][j].pass_seat
1394                     plane[i][j-1].pass_char = p1r
1395                     plane[i][j-1].pass_seat = p1c
1396                     plane[i][j-1].value = 1
1397                     plane[i][j].pass_char = 0
1398                     plane[i][j].pass_seat = 0
1399                     plane[i][j].value = 0
1400
1401     j = 0
1402     for i in range(4,250):
1403         if(plane[i-1][j].value==1):
1404             continue
1405         else:
1406             if(plane[i][j].value==1):
1407                 if(plane[i][j].pass_char<5):
1408                     if(i>3):
1409                         p1r = plane[i][j].pass_char
1410                         p1c = plane[i][j].pass_seat
1411                         plane[i-1][j].pass_char = p1r
1412                         plane[i-1][j].pass_seat = p1c
1413                         plane[i-1][j].value = 1
1414                         plane[i][j].pass_char = 0
1415                         plane[i][j].pass_seat = 0
1416                         plane[i][j].value = 0
1417                 else:
1418                     if(i>7):
1419                         p1r = plane[i][j].pass_char

```

```

1420         p1c = plane[i][j].pass_seat
1421         plane[i-1][j].pass_char = p1r
1422         plane[i-1][j].pass_seat = p1c
1423         plane[i-1][j].value = 1
1424         plane[i][j].pass_char = 0
1425         plane[i][j].pass_seat = 0
1426         plane[i][j].value = 0
1427
1428     j = 41
1429     for i in range(4,250):
1430         if(plane[i-1][j].value==1):
1431             continue
1432         else:
1433             if(plane[i][j].value==1):
1434                 if(plane[i][j].pass_char<5):
1435                     if(i>3):
1436                         p1r = plane[i][j].pass_char
1437                         p1c = plane[i][j].pass_seat
1438                         plane[i-1][j].pass_char = p1r
1439                         plane[i-1][j].pass_seat = p1c
1440                         plane[i-1][j].value = 1
1441                         plane[i][j].pass_char = 0
1442                         plane[i][j].pass_seat = 0
1443                         plane[i][j].value = 0
1444                 else:
1445                     if(i>7):
1446                         p1r = plane[i][j].pass_char
1447                         p1c = plane[i][j].pass_seat
1448                         plane[i-1][j].pass_char = p1r
1449                         plane[i-1][j].pass_seat = p1c
1450                         plane[i-1][j].value = 1
1451                         plane[i][j].pass_char = 0
1452                         plane[i][j].pass_seat = 0
1453                         plane[i][j].value = 0
1454
1455     #print("time",time)
1456     #print("check",check)
1457
1458
1459     """print(plane[8][0].pass_char,plane[8][0].pass_seat)
1460     print(plane[8][41].pass_char,plane[8][41].pass_seat)
1461     for i in range(0,250):
1462         for j in range(0,42):
1463             #print(plane[i][j].value)
1464             a[i][j] = plane[i][j].value
1465     plt.figure('time'+str(time))
1466     im = plt.imshow(a[0:10])
1467     ax = plt.gca()
1468     ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
1469     ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
1470     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
1471     plt.savefig('time'+str(time)+'.png')
1472     #plt.show()"""
1473     return time
1474
1475 print(run3(2,0.3,0.5,242))

```

B.4 Codes of Disembarking Process, Airplane I

```
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 run1(case,RL,RJ,N):
12     # Create position of people in "Narrow Body" Passenger Aircraft
13     arr = np.arange(1,196)
14     list_0 = arr.tolist()
15     for i in range (99):
16         list_0[i] = [math.floor(list_0[i]/33)+1,list_0[i]%33]
17     for i in range (99,195):
18         list_0[i] = [math.floor((list_0[i]-99)/32)+5,(list_0[i]-99)%32+1]
19     list_0[32] = [1,33]
20     list_0[65] = [2,33]
21     list_0[98] = [3,33]
22     list_0[130] = [5,33]
23     list_0[162] = [6,33]
24     list_0[194] = [7,33]
25
26     # Random luggage stow time of each people by experimental data and Weibull
distribution
27     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]
28     luggage_arr = np.array(luggage)
29     mean = luggage_arr.mean()
30     std = stdev(luggage_arr)
31     k = (std/mean)**(-1.086)
32     z = 1 + 1/k
33     def f(x):
34         return math.exp(-x)*(x**(z-1))
35     gamma,err = quad(f, 0, math.inf)
36     c = mean/gamma
37     for i in range (len(list_0)):
38         weibull = (c*(np.random.weibull(k, 1))).tolist()
39         time = round((weibull[0]/1.42))
40         list_0[i].append(time)
41
42     #Case1
43     list_1 = copy.deepcopy(list_0)
44     pri_max_1 = 1
45
46     for i in range (len(list_1)):
47         list_1[i].append(1)
48
49     list_late_1 = random.sample(list_1,round(N*RL))
50
51     for i in range (len(list_late_1)):
52         for j in range (len(list_1)):
53             if list_late_1[i] == list_1[j]:
54                 list_1[j][3] = pri_max_1+1

```

```

55         if list_1[j][0] == 3:
56             for k in range (len(list_1)):
57                 if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 1)
| (list_1[k][0] == 2)):
58                     list_1[k][3] = pri_max_1+1
59         if list_1[j][0] == 2:
60             for k in range (len(list_1)):
61                 if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 1):
62                     list_1[k][3] = pri_max_1+1
63         if list_1[j][0] == 5:
64             for k in range (len(list_1)):
65                 if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 6)
| (list_1[k][0] == 7)):
66                     list_1[k][3] = pri_max_1+1
67         if list_1[j][0] == 6:
68             for k in range (len(list_1)):
69                 if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 7):
70                     list_1[k][3] = pri_max_1+1
71
72     class agent_1:
73         def __init__(self, char, seat, bag, pri):
74             self.char = char
75             self.seat = seat
76             self.bag = bag
77             self.pri = pri
78
79     passenger_1 = []
80     for i in range(len(list_1)):
81         passenger_1.append(agent_1(list_1[i][0], list_1[i][1], list_1[i][2], list_1[
i][3]))
82
83
84     #Case2
85     list_2 = copy.deepcopy(list_0)
86     pri_max_2 = 3
87
88     for i in range (len(list_2)):
89         if (list_2[i][0] == 1) | (list_2[i][0] == 7):
90             list_2[i].append(3)
91         elif (list_2[i][0] == 2) | (list_2[i][0] == 6):
92             list_2[i].append(2)
93         elif (list_2[i][0] == 3) | (list_2[i][0] == 5):
94             list_2[i].append(1)
95
96     list_late_2 = random.sample(list_2, round(N*RL))
97
98     for i in range (len(list_late_2)):
99         for j in range (len(list_2)):
100             if list_late_2[i] == list_2[j]:
101                 list_2[j][3] = pri_max_2+1
102                 if list_2[j][0] == 3:
103                     for k in range (len(list_2)):
104                         if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 1)
| (list_2[k][0] == 2)):
105                             list_2[k][3] = pri_max_2+1
106                 if list_2[j][0] == 2:
107                     for k in range (len(list_2)):
108                         if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 1):
109                             list_2[k][3] = pri_max_2+1
110                 if list_2[j][0] == 5:

```

```

111         for k in range (len(list_2)):
112             if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 6)
| (list_2[k][0] == 7)):
113                 list_2[k][3] = pri_max_2+1
114             if list_2[j][0] == 6:
115                 for k in range (len(list_2)):
116                     if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 7):
117                         list_2[k][3] = pri_max_2+1
118
119 class agent_2:
120     def __init__(self, char, seat, bag, pri):
121         self.char = char
122         self.seat = seat
123         self.bag = bag
124         self.pri = pri
125
126 passenger_2 = []
127 for i in range(len(list_2)):
128     passenger_2.append(agent_2(list_2[i][0], list_2[i][1], list_2[i][2], list_2[
i][3]))
129
130 #Case3
131 list_3= copy.deepcopy(list_0)
132 pri_max_3 = 5
133
134 for i in range (len(list_3)):
135     if (list_3[i][0] == 3) & (1 <= list_3[i][1] <= 19):
136         list_3[i].append(1)
137     elif (list_3[i][0] == 5) & (2 <= list_3[i][1] <= 19):
138         list_3[i].append(1)
139     elif (list_3[i][0] == 2) & (1 <= list_3[i][1] <= 7):
140         list_3[i].append(2)
141     elif (list_3[i][0] == 6) & (2 <= list_3[i][1] <= 7):
142         list_3[i].append(2)
143     elif (list_3[i][0] == 3) & (20 <= list_3[i][1] <= 33):
144         list_3[i].append(2)
145     elif (list_3[i][0] == 5) & (20 <= list_3[i][1] <= 33):
146         list_3[i].append(2)
147     elif (list_3[i][0] == 1) & (1 <= list_3[i][1] <= 7):
148         list_3[i].append(3)
149     elif (list_3[i][0] == 2) & (8 <= list_3[i][1] <= 19):
150         list_3[i].append(3)
151     elif (list_3[i][0] == 6) & (8 <= list_3[i][1] <= 19):
152         list_3[i].append(3)
153     elif (list_3[i][0] == 7) & (2 <= list_3[i][1] <= 7):
154         list_3[i].append(3)
155     elif (list_3[i][0] == 1) & (8 <= list_3[i][1] <= 12):
156         list_3[i].append(4)
157     elif (list_3[i][0] == 2) & (20 <= list_3[i][1] <= 33):
158         list_3[i].append(4)
159     elif (list_3[i][0] == 6) & (20 <= list_3[i][1] <= 33):
160         list_3[i].append(4)
161     elif (list_3[i][0] == 7) & (8 <= list_3[i][1] <= 12):
162         list_3[i].append(4)
163     elif (list_3[i][0] == 1) & (13 <= list_3[i][1] <= 33):
164         list_3[i].append(5)
165     elif (list_3[i][0] == 7) & (13 <= list_3[i][1] <= 33):
166         list_3[i].append(5)
167
168

```

```

169 list_late_3 = random.sample(list_3,round(N*RL))
170
171 for i in range (len(list_late_3)):
172     for j in range (len(list_3)):
173         if list_late_3[i] == list_3[j]:
174             list_3[j][3] = pri_max_3+1
175             if list_3[j][0] == 3:
176                 for k in range (len(list_3)):
177                     if (list_3[k][1] == list_3[j][1]) & ((list_3[k][0] == 1)
| (list_3[k][0] == 2)):
178                         list_3[k][3] = pri_max_3+1
179                 if list_3[j][0] == 2:
180                     for k in range (len(list_3)):
181                         if (list_3[k][1] == list_3[j][1]) & (list_3[k][0] == 1):
182                             list_3[k][3] = pri_max_3+1
183                 if list_3[j][0] == 5:
184                     for k in range (len(list_3)):
185                         if (list_3[k][1] == list_3[j][1]) & ((list_3[k][0] == 6)
| (list_3[k][0] == 7)):
186                             list_3[k][3] = pri_max_3+1
187                 if list_3[j][0] == 6:
188                     for k in range (len(list_3)):
189                         if (list_3[k][1] == list_3[j][1]) & (list_3[k][0] == 7):
190                             list_3[k][3] = pri_max_3+1
191
192 class agent_3:
193     def __init__(self,char,seat,bag,pri):
194         self.char = char
195         self.seat = seat
196         self.bag = bag
197         self.pri = pri
198
199 passenger_3 = []
200 for i in range(len(list_3)):
201     passenger_3.append(agent_3(list_3[i][0],list_3[i][1],list_3[i][2],list_3[
i][3]))
202
203 class person:
204     def __init__(self,char,seat,bag,pri):
205         self.char = char
206         self.seat = seat
207         self.bag = bag
208         self.pri = pri
209
210 list_pass = [[person(0,0,0,0) for i in range(0,34)] for j in range(0,8)]
211
212 class grid:
213     def __init__(self,type,value,pass_char,pass_seat,pass_pri):
214         self.type = type
215         # 0 -> block
216         # 1 -> queue
217         # 2 -> aisle
218         # 3 -> seat
219         self.value = value
220         # 0 -> available
221         # 1 -> passenger
222         self.pass_char = pass_char
223         self.pass_seat = pass_seat
224         self.pass_pri = pass_pri
225

```

```
226 plane = [[grid(0,0,0,0,0) for i in range(0,36)] for i in range(0,203)]
227
228 for i in range(1,8):
229     for j in range(1,34):
230         plane[i][j].type = 3
231
232 for i in range(5,8):
233     plane[i][1].type = 0
234
235 for i in range(0,36):
236     plane[4][i].type = 2
237
238 for i in range(5,203):
239     plane[i][0].type = 1
240
241 def C1(passenger_1):
242     for i in range(len(passenger_1)):
243         list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
244         list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
245         list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
246         list_pass[passenger_1[i].char][passenger_1[i].seat].pri = passenger_1
[i].pri
247         plane[passenger_1[i].char][passenger_1[i].seat].value = 1
248         plane[passenger_1[i].char][passenger_1[i].seat].pass_char =
passenger_1[i].char
249         plane[passenger_1[i].char][passenger_1[i].seat].pass_seat =
passenger_1[i].seat
250         plane[passenger_1[i].char][passenger_1[i].seat].pass_pri =
passenger_1[i].pri
251
252 def C2(passenger_2):
253     for i in range(len(passenger_2)):
254         list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
255         list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
passenger_2[i].seat
256         list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
[i].bag
257         list_pass[passenger_2[i].char][passenger_2[i].seat].pri = passenger_2
[i].pri
258         plane[passenger_2[i].char][passenger_2[i].seat].value = 1
259         plane[passenger_2[i].char][passenger_2[i].seat].pass_char =
passenger_2[i].char
260         plane[passenger_2[i].char][passenger_2[i].seat].pass_seat =
passenger_2[i].seat
261         plane[passenger_2[i].char][passenger_2[i].seat].pass_pri =
passenger_2[i].pri
262
263 def C3(passenger_3):
264     for i in range(len(passenger_3)):
265         list_pass[passenger_3[i].char][passenger_3[i].seat].char =
passenger_3[i].char
266         list_pass[passenger_3[i].char][passenger_3[i].seat].seat =
passenger_3[i].seat
267         list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3
[i].bag
268         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

```



```
325     plane[i][j].pass_pri = p1p
326     plane[i][j].value = 1
327     plane[i-1][j].pass_char = 0
328     plane[i-1][j].pass_seat = 0
329     plane[i-1][j].pass_pri = 0
330     plane[i-1][j].value = 0
331
332     i = 5
333     for j in range(2,34):
334         if(plane[i][j].value==0 and plane[i+1][j].value==1):
335             p1r = plane[i+1][j].pass_char
336             p1c = plane[i+1][j].pass_seat
337             p1p = plane[i+1][j].pass_pri
338             plane[i][j].pass_char = p1r
339             plane[i][j].pass_seat = p1c
340             plane[i][j].pass_pri = p1p
341             plane[i][j].value = 1
342             plane[i+1][j].pass_char = 0
343             plane[i+1][j].pass_seat = 0
344             plane[i+1][j].pass_pri = 0
345             plane[i+1][j].value = 0
346
347     i = 2
348     for j in range(1,34):
349         if(plane[i][j].value==0 and plane[i-1][j].value==1):
350             p1r = plane[i-1][j].pass_char
351             p1c = plane[i-1][j].pass_seat
352             p1p = plane[i-1][j].pass_pri
353             plane[i][j].pass_char = p1r
354             plane[i][j].pass_seat = p1c
355             plane[i][j].pass_pri = p1p
356             plane[i][j].value = 1
357             plane[i-1][j].pass_char = 0
358             plane[i-1][j].pass_seat = 0
359             plane[i-1][j].pass_pri = 0
360             plane[i-1][j].value = 0
361
362     i = 6
363     for j in range(2,34):
364         if(plane[i][j].value==0 and plane[i+1][j].value==1):
365             p1r = plane[i+1][j].pass_char
366             p1c = plane[i+1][j].pass_seat
367             p1p = plane[i+1][j].pass_pri
368             plane[i][j].pass_char = p1r
369             plane[i][j].pass_seat = p1c
370             plane[i][j].pass_pri = p1p
371             plane[i][j].value = 1
372             plane[i+1][j].pass_char = 0
373             plane[i+1][j].pass_seat = 0
374             plane[i+1][j].pass_pri = 0
375             plane[i+1][j].value = 0
376
377     time+=1
378
379     """for i in range(0,203):
380         for j in range(0,36):
381             #print(plane[i][j].value)
382             a[i][j] = plane[i][j].value
383     plt.figure('time'+str(time))
384     im = plt.imshow(a[0:8])
```

```

385 ax = plt.gca()
386 ax.set_xticks(np.arange(-.5, 36, 1), minor=True)
387 ax.set_yticks(np.arange(-.5, 8, 1), minor=True)
388 ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
389 plt.savefig('time'+str(time)+'_png')"""
390
391 while(1):
392     x = min_pri(plane)
393     #check
394     check = check_pass(plane)
395     if(check==0):
396         break
397     time+=1
398
399     j = 0
400     for i in reversed(range(4,202)):
401         if(plane[i][j].value==1 and plane[i+1][j].value==0):
402             p1r = plane[i][j].pass_char
403             p1c = plane[i][j].pass_seat
404             p1p = plane[i][j].pass_pri
405             plane[i+1][j].pass_char = p1r
406             plane[i+1][j].pass_seat = p1c
407             plane[i+1][j].pass_pri = p1p
408             plane[i+1][j].value = 1
409             plane[i][j].pass_char = 0
410             plane[i][j].pass_seat = 0
411             plane[i][j].pass_pri = 0
412             plane[i][j].value = 0
413
414         i = 4
415         for j in range(0,34):
416             if(plane[i][j].value==1):
417                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag>0)
418 :
419                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].bag
420 ==1
421                     continue
422                 if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char][
423 plane[i][j+1].pass_seat].bag==0):
424                     p1r = plane[i][j+1].pass_char
425                     p1c = plane[i][j+1].pass_seat
426                     p1p = plane[i][j+1].pass_pri
427                     plane[i][j].pass_char = p1r
428                     plane[i][j].pass_seat = p1c
429                     plane[i][j].pass_pri = p1p
430                     plane[i][j].value = 1
431                     plane[i][j+1].pass_char = 0
432                     plane[i][j+1].pass_seat = 0
433                     plane[i][j+1].pass_pri = 0
434                     plane[i][j+1].value = 0
435                     continue
436                 if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char][
437 plane[i][j+1].pass_seat].bag!=0):
438                     continue
439                 if(j==0):
440                     continue
441                 if(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1 and plane[i
442 +1][j].pass_pri==x and plane[i+1][j].value == 1):
443                     ch = random.choice([i-1,i+1])
444                     p1r = plane[ch][j].pass_char

```

```
440     p1c = plane[ch][j].pass_seat
441     p1p = plane[ch][j].pass_pri
442     plane[i][j].pass_char = p1r
443     plane[i][j].pass_seat = p1c
444     plane[i][j].pass_pri = p1p
445     plane[i][j].value = 1
446     plane[ch][j].pass_char = 0
447     plane[ch][j].pass_seat = 0
448     plane[ch][j].pass_pri = 0
449     plane[ch][j].value = 0
450     elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
451         p1r = plane[i-1][j].pass_char
452         p1c = plane[i-1][j].pass_seat
453         p1p = plane[i-1][j].pass_pri
454         plane[i][j].pass_char = p1r
455         plane[i][j].pass_seat = p1c
456         plane[i][j].pass_pri = p1p
457         plane[i][j].value = 1
458         plane[i-1][j].pass_char = 0
459         plane[i-1][j].pass_seat = 0
460         plane[i-1][j].pass_pri = 0
461         plane[i-1][j].value = 0
462     elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
463         p1r = plane[i+1][j].pass_char
464         p1c = plane[i+1][j].pass_seat
465         p1p = plane[i+1][j].pass_pri
466         plane[i][j].pass_char = p1r
467         plane[i][j].pass_seat = p1c
468         plane[i][j].pass_pri = p1p
469         plane[i][j].value = 1
470         plane[i+1][j].pass_char = 0
471         plane[i+1][j].pass_seat = 0
472         plane[i+1][j].pass_pri = 0
473         plane[i+1][j].value = 0
474     else:
475         continue
476
477 i = 3
478 for j in range(1,34):
479     if(plane[i][j].value==0 and plane[i-1][j].value==1):
480         p1r = plane[i-1][j].pass_char
481         p1c = plane[i-1][j].pass_seat
482         p1p = plane[i-1][j].pass_pri
483         plane[i][j].pass_char = p1r
484         plane[i][j].pass_seat = p1c
485         plane[i][j].pass_pri = p1p
486         plane[i][j].value = 1
487         plane[i-1][j].pass_char = 0
488         plane[i-1][j].pass_seat = 0
489         plane[i-1][j].pass_pri = 0
490         plane[i-1][j].value = 0
491
492 i = 5
493 for j in range(2,34):
494     if(plane[i][j].value==0 and plane[i+1][j].value==1):
495         p1r = plane[i+1][j].pass_char
496         p1c = plane[i+1][j].pass_seat
497         p1p = plane[i+1][j].pass_pri
498         plane[i][j].pass_char = p1r
499         plane[i][j].pass_seat = p1c
```

```

500         plane[i][j].pass_pri = p1p
501         plane[i][j].value = 1
502         plane[i+1][j].pass_char = 0
503         plane[i+1][j].pass_seat = 0
504         plane[i+1][j].pass_pri = 0
505         plane[i+1][j].value = 0
506
507     i = 2
508     for j in range(1,34):
509         if(plane[i][j].value==0 and plane[i-1][j].value==1):
510             p1r = plane[i-1][j].pass_char
511             p1c = plane[i-1][j].pass_seat
512             p1p = plane[i-1][j].pass_pri
513             plane[i][j].pass_char = p1r
514             plane[i][j].pass_seat = p1c
515             plane[i][j].pass_pri = p1p
516             plane[i][j].value = 1
517             plane[i-1][j].pass_char = 0
518             plane[i-1][j].pass_seat = 0
519             plane[i-1][j].pass_pri = 0
520             plane[i-1][j].value = 0
521
522     i = 6
523     for j in range(2,34):
524         if(plane[i][j].value==0 and plane[i+1][j].value==1):
525             p1r = plane[i+1][j].pass_char
526             p1c = plane[i+1][j].pass_seat
527             p1p = plane[i+1][j].pass_pri
528             plane[i][j].pass_char = p1r
529             plane[i][j].pass_seat = p1c
530             plane[i][j].pass_pri = p1p
531             plane[i][j].value = 1
532             plane[i+1][j].pass_char = 0
533             plane[i+1][j].pass_seat = 0
534             plane[i+1][j].pass_pri = 0
535             plane[i+1][j].value = 0
536
537     """j = 0
538     i = 4
539     if(plane[i][j].value==1 and plane[i+1][j].value==0):
540         p1r = plane[i][j].pass_char
541         p1c = plane[i][j].pass_seat
542         p1p = plane[i][j].pass_pri
543         plane[i+1][j].pass_char = p1r
544         plane[i+1][j].pass_seat = p1c
545         plane[i+1][j].pass_pri = p1p
546         plane[i+1][j].value = 1
547         plane[i][j].pass_char = 0
548         plane[i][j].pass_seat = 0
549         plane[i][j].pass_pri = 0
550         plane[i][j].value = 0"""
551
552     #print("time",time)
553     #print("check",check)
554     #print(plane[4][0].value,plane[4][0].pass_pri)
555     #print(plane[5][0].value,plane[5][0].pass_pri)
556     """i = 4
557     for j in reversed(range(0,36)):
558         if(plane[i][j].value==1):
559             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 Weibull
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]
31 luggage_arr = np.array(luggage)
32 mean = luggage_arr.mean()
33 std = stdev(luggage_arr)
34 k = (std/mean)**(-1.086)
35 z = 1 + 1/k
36 def f(x):
37     return math.exp(-x)*(x**(z-1))
38 gamma,err = quad(f, 0, math.inf)
39 c = mean/gamma
40 for i in range (len(list_0)):
41     weibull = (c*(np.random.weibull(k, 1))).tolist()
42     time = round((weibull[0]/1.42))
43     list_0[i].append(time)
44
45 #Case1
46 list_1 = copy.deepcopy(list_0)
47 pri_max_1 = 1
48
49 for i in range (len(list_1)):
50     list_1[i].append(1)
51
52 list_late_1 = random.sample(list_1,round(N*RL))
53
54 for i in range (len(list_late_1)):
55     for j in range (len(list_1)):
56         if list_late_1[i] == list_1[j]:
57             list_1[j][3] = pri_max_1+1
58             if list_1[j][0] == 3:
59                 for k in range (len(list_1)):
60                     if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 1)
| (list_1[k][0] == 2)):
61                         list_1[k][3] = pri_max_1+1
62             if list_1[j][0] == 2:
63                 for k in range (len(list_1)):
64                     if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 1):
65                         list_1[k][3] = pri_max_1+1
66             if list_1[j][0] == 5:
67                 for k in range (len(list_1)):
68                     if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 6)
| (list_1[k][0] == 7)):
69                         list_1[k][3] = pri_max_1+1
70             if list_1[j][0] == 6:
71                 for k in range (len(list_1)):
72                     if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 7):
73                         list_1[k][3] = pri_max_1+1
74             if list_1[j][0] == 10:
75                 for k in range (len(list_1)):
76                     if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 9)
| (list_1[k][0] == 8)):
77                         list_1[k][3] = pri_max_1+1
78             if list_1[j][0] == 9:
79                 for k in range (len(list_1)):
80                     if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 8):
81                         list_1[k][3] = pri_max_1+1
82             if list_1[j][0] == 12:
83                 for k in range (len(list_1)):
84                     if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 13)

```

```

85 | (list_1[k][0] == 14)):
86     list_1[k][3] = pri_max_1+1
87     if list_1[j][0] == 13:
88         for k in range (len(list_1)):
89             if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 14):
90                 list_1[k][3] = pri_max_1+1
91     if list_1[j][0] == 17:
92         for k in range (len(list_1)):
93             if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 16)
94 | (list_1[k][0] == 15)):
95     list_1[k][3] = pri_max_1+1
96     if list_1[j][0] == 16:
97         for k in range (len(list_1)):
98             if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 15):
99                 list_1[k][3] = pri_max_1+1
100     if list_1[j][0] == 19:
101         for k in range (len(list_1)):
102             if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 20)
103 | (list_1[k][0] == 21)):
104     list_1[k][3] = pri_max_1+1
105     if list_1[j][0] == 20:
106         for k in range (len(list_1)):
107             if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 21):
108                 list_1[k][3] = pri_max_1+1
109     if list_1[j][0] == 24:
110         for k in range (len(list_1)):
111             if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 23)
112 | (list_1[k][0] == 22)):
113     list_1[k][3] = pri_max_1+1
114     if list_1[j][0] == 23:
115         for k in range (len(list_1)):
116             if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 22):
117                 list_1[k][3] = pri_max_1+1
118     if list_1[j][0] == 26:
119         for k in range (len(list_1)):
120             if (list_1[k][1] == list_1[j][1]) & ((list_1[k][0] == 27)
121 | (list_1[k][0] == 28)):
122     list_1[k][3] = pri_max_1+1
123     if list_1[j][0] == 27:
124         for k in range (len(list_1)):
125             if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 28):
126                 list_1[k][3] = pri_max_1+1
127
128 class agent_1:
129     def __init__(self, char, seat, bag, pri):
130         self.char = char
131         self.seat = seat
132         self.bag = bag
133         self.pri = pri
134
135 passenger_1 = []
136 for i in range(len(list_1)):
137     passenger_1.append(agent_1(list_1[i][0], list_1[i][1], list_1[i][2], list_1[
138 i][3]))
139
140 #Case2
141 list_2 = copy.deepcopy(list_0)
142 pri_max_2 = 3
143
144 for i in range (len(list_2)):

```

```

139     if (list_0[i][0] == 3) or (list_0[i][0] == 5) or (list_0[i][0] == 10) or
140     (list_0[i][0] == 12) or (list_0[i][0] == 17) or (list_0[i][0] == 19) or (
141     list_0[i][0] == 24) or (list_0[i][0] == 26):
142         list_2[i].append(1)
143     elif (list_0[i][0] == 2) or (list_0[i][0] == 6) or (list_0[i][0] == 9) or
144     (list_0[i][0] == 13) or (list_0[i][0] == 16) or (list_0[i][0] == 20) or (
145     list_0[i][0] == 23) or (list_0[i][0] == 27):
146         list_2[i].append(2)
147     elif (list_0[i][0] == 1) or (list_0[i][0] == 7) or (list_0[i][0] == 8) or
148     (list_0[i][0] == 14) or (list_0[i][0] == 15) or (list_0[i][0] == 21) or (
149     list_0[i][0] == 22) or (list_0[i][0] == 28):
150         list_2[i].append(3)
151
152 list_late_2 = random.sample(list_2,round(N*RL))
153
154 for i in range (len(list_late_2)):
155     for j in range (len(list_2)):
156         if list_late_2[i] == list_2[j]:
157             list_2[j][3] = pri_max_2+1
158             if list_2[j][0] == 3:
159                 for k in range (len(list_2)):
160                     if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 1)
161 | (list_2[k][0] == 2)):
162                         list_2[k][3] = pri_max_2+1
163             if list_2[j][0] == 2:
164                 for k in range (len(list_2)):
165                     if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 1):
166                         list_2[k][3] = pri_max_2+1
167             if list_2[j][0] == 5:
168                 for k in range (len(list_2)):
169                     if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 6)
170 | (list_2[k][0] == 7)):
171                         list_2[k][3] = pri_max_2+1
172             if list_2[j][0] == 6:
173                 for k in range (len(list_2)):
174                     if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 7):
175                         list_2[k][3] = pri_max_2+1
176             if list_2[j][0] == 10:
177                 for k in range (len(list_2)):
178                     if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 9)
179 | (list_2[k][0] == 8)):
180                         list_2[k][3] = pri_max_2+1
181             if list_2[j][0] == 9:
182                 for k in range (len(list_2)):
183                     if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 8):
184                         list_2[k][3] = pri_max_2+1
185             if list_2[j][0] == 12:
186                 for k in range (len(list_2)):
187                     if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 13)
188 | (list_2[k][0] == 14)):
189                         list_2[k][3] = pri_max_2+1
190             if list_2[j][0] == 13:
191                 for k in range (len(list_2)):
192                     if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 14):
193                         list_2[k][3] = pri_max_2+1
194             if list_2[j][0] == 17:
195                 for k in range (len(list_2)):
196                     if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 16)
197 | (list_2[k][0] == 15)):
198                         list_2[k][3] = pri_max_2+1

```



```

188         if list_2[j][0] == 16:
189             for k in range (len(list_2)):
190                 if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 15):
191                     list_2[k][3] = pri_max_2+1
192         if list_2[j][0] == 19:
193             for k in range (len(list_2)):
194                 if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 20)
| (list_2[k][0] == 21)):
195                     list_2[k][3] = pri_max_2+1
196         if list_2[j][0] == 20:
197             for k in range (len(list_2)):
198                 if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 21):
199                     list_2[k][3] = pri_max_2+1
200         if list_2[j][0] == 24:
201             for k in range (len(list_2)):
202                 if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 23)
| (list_2[k][0] == 22)):
203                     list_2[k][3] = pri_max_2+1
204         if list_2[j][0] == 23:
205             for k in range (len(list_2)):
206                 if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 22):
207                     list_2[k][3] = pri_max_2+1
208         if list_2[j][0] == 26:
209             for k in range (len(list_2)):
210                 if (list_2[k][1] == list_2[j][1]) & ((list_2[k][0] == 27)
| (list_2[k][0] == 28)):
211                     list_2[k][3] = pri_max_2+1
212         if list_2[j][0] == 27:
213             for k in range (len(list_2)):
214                 if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 28):
215                     list_2[k][3] = pri_max_2+1
216
217     class agent_2:
218         def __init__(self, char, seat, bag, pri):
219             self.char = char
220             self.seat = seat
221             self.bag = bag
222             self.pri = pri
223
224     passenger_2 = []
225     for i in range(len(list_2)):
226         passenger_2.append(agent_2(list_2[i][0], list_2[i][1], list_2[i][2], list_2[
i][3]))
227
228
229     class person:
230         def __init__(self, char, seat, bag, pri):
231             self.char = char
232             self.seat = seat
233             self.bag = bag
234             self.pri = pri
235
236     list_pass = [[person(0,0,0,0) for i in range(0,15)] for j in range(0,29)]
237
238     class grid:
239         def __init__(self, type, value, pass_char, pass_seat, pass_pri):
240             self.type = type
241             # 0 -> block
242             # 1 -> queue
243             # 2 -> aisle

```

```
244     # 3 -> seat
245     self.value = value
246     # 0 -> available
247     # 1 -> passenger
248     self.pass_char = pass_char
249     self.pass_seat = pass_seat
250     self.pass_pri = pass_pri
251
252 plane = [[grid(0,0,0,0,0) for i in range(0,17)] for j in range(0,347)]
253
254 for i in range(1,4):
255     for j in range(4,15):
256         plane[i][j].type = 3
257
258 for i in range(5,11):
259     for j in range(1,15):
260         plane[i][j].type = 3
261
262 for i in range(12,18):
263     for j in range(1,15):
264         plane[i][j].type = 3
265
266 for i in range(19,25):
267     for j in range(1,15):
268         plane[i][j].type = 3
269
270 for i in range(26,29):
271     for j in range(4,15):
272         plane[i][j].type = 3
273
274 for i in range(4,347):
275     plane[i][0].type = 1
276
277 q = [4,11,18,25]
278 for i in q:
279     for j in range(0,17):
280         plane[i][j].type = 2
281
282 def C1(passenger_1):
283     for i in range(len(passenger_1)):
284         list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
285         list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
286         list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
287         list_pass[passenger_1[i].char][passenger_1[i].seat].pri = passenger_1
[i].pri
288         plane[passenger_1[i].char][passenger_1[i].seat].value = 1
289         plane[passenger_1[i].char][passenger_1[i].seat].pass_char =
passenger_1[i].char
290         plane[passenger_1[i].char][passenger_1[i].seat].pass_seat =
passenger_1[i].seat
291         plane[passenger_1[i].char][passenger_1[i].seat].pass_pri =
passenger_1[i].pri
292
293 def C2(passenger_2):
294     for i in range(len(passenger_2)):
295         list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
```

```
296         list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
passenger_2[i].seat
297         list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
[i].bag
298         list_pass[passenger_2[i].char][passenger_2[i].seat].pri = passenger_2
[i].pri
299         plane[passenger_2[i].char][passenger_2[i].seat].value = 1
300         plane[passenger_2[i].char][passenger_2[i].seat].pass_char =
passenger_2[i].char
301         plane[passenger_2[i].char][passenger_2[i].seat].pass_seat =
passenger_2[i].seat
302         plane[passenger_2[i].char][passenger_2[i].seat].pass_pri =
passenger_2[i].pri
303
304         if(case==1):
305             C1(passenger_1)
306         if(case==2):
307             C2(passenger_2)
308         time = 0
309
310         a = [[0 for i in range(0,17)] for i in range(0,347)]
311
312         for i in range(0,347):
313             for j in range(0,17):
314                 #print(plane[i][j].value)
315                 a[i][j] = plane[i][j].type
316
317         time = 0
318
319         """plt.figure('time'+str(time))
320         im = plt.imshow(a[0:29])
321         ax = plt.gca()
322         ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
323         ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
324         ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
325         plt.savefig("figure_2_d.png")
326         plt.show()"""
327
328         def check_pass(plane):
329             check = 0
330             for w in range(0,29):
331                 for z in range(0,17):
332                     if(plane[w][z].type>0 and plane[w][z].value == 1):
333                         check+=1
334             return check
335
336         def min_pri(plane):
337             x = 10
338             for w in range(0,29):
339                 for z in range(0,17):
340                     if(plane[w][z].type>0):
341                         if(plane[w][z].value==1):
342                             if(plane[w][z].pass_pri<x):
343                                 x = plane[w][z].pass_pri
344             return x
345
346         left_1 = [3,10,17,24]
347         for i in left_1:
348             for j in range(1,15):
349                 if(plane[i][j].value==0 and plane[i-1][j].value==1):
```

```
350     p1r = plane[i-1][j].pass_char
351     p1c = plane[i-1][j].pass_seat
352     p1p = plane[i-1][j].pass_pri
353     plane[i][j].pass_char = p1r
354     plane[i][j].pass_seat = p1c
355     plane[i][j].pass_pri = p1p
356     plane[i][j].value = 1
357     plane[i-1][j].pass_char = 0
358     plane[i-1][j].pass_seat = 0
359     plane[i-1][j].pass_pri = 0
360     plane[i-1][j].value = 0
361
362 right_1 = [5,12,19,26]
363 for i in right_1:
364     for j in range(1,15):
365         if(plane[i][j].value==0 and plane[i+1][j].value==1):
366             p1r = plane[i+1][j].pass_char
367             p1c = plane[i+1][j].pass_seat
368             p1p = plane[i+1][j].pass_pri
369             plane[i][j].pass_char = p1r
370             plane[i][j].pass_seat = p1c
371             plane[i][j].pass_pri = p1p
372             plane[i][j].value = 1
373             plane[i+1][j].pass_char = 0
374             plane[i+1][j].pass_seat = 0
375             plane[i+1][j].pass_pri = 0
376             plane[i+1][j].value = 0
377
378 left_2 = [2,9,16,23]
379 for i in left_2:
380     for j in range(1,15):
381         if(plane[i][j].value==0 and plane[i-1][j].value==1):
382             p1r = plane[i-1][j].pass_char
383             p1c = plane[i-1][j].pass_seat
384             p1p = plane[i-1][j].pass_pri
385             plane[i][j].pass_char = p1r
386             plane[i][j].pass_seat = p1c
387             plane[i][j].pass_pri = p1p
388             plane[i][j].value = 1
389             plane[i-1][j].pass_char = 0
390             plane[i-1][j].pass_seat = 0
391             plane[i-1][j].pass_pri = 0
392             plane[i-1][j].value = 0
393
394 right_2 = [6,13,20,27]
395 for i in right_2:
396     for j in range(1,15):
397         if(plane[i][j].value==0 and plane[i+1][j].value==1):
398             p1r = plane[i+1][j].pass_char
399             p1c = plane[i+1][j].pass_seat
400             p1p = plane[i+1][j].pass_pri
401             plane[i][j].pass_char = p1r
402             plane[i][j].pass_seat = p1c
403             plane[i][j].pass_pri = p1p
404             plane[i][j].value = 1
405             plane[i+1][j].pass_char = 0
406             plane[i+1][j].pass_seat = 0
407             plane[i+1][j].pass_pri = 0
408             plane[i+1][j].value = 0
409
```

```

410     time+=1
411
412     for i in range(0,347):
413         for j in range(0,17):
414             #print(plane[i][j].value)
415             a[i][j] = plane[i][j].value
416 plt.figure('time'+str(time))
417 im = plt.imshow(a[0:29])
418 ax = plt.gca()
419 ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
420 ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
421 ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
422 plt.savefig('time'+str(time)+'.png')
423
424 while(1):
425     x = min_pri(plane)
426     #check
427     check = check_pass(plane)
428     if(check==0):
429         break
430     time+=1
431
432     j = 0
433     for i in reversed(range(4,346)):
434         if(plane[i][j].value==1 and plane[i+1][j].value==0):
435             p1r = plane[i][j].pass_char
436             p1c = plane[i][j].pass_seat
437             p1p = plane[i][j].pass_pri
438             plane[i+1][j].pass_char = p1r
439             plane[i+1][j].pass_seat = p1c
440             plane[i+1][j].pass_pri = p1p
441             plane[i+1][j].value = 1
442             plane[i][j].pass_char = 0
443             plane[i][j].pass_seat = 0
444             plane[i][j].pass_pri = 0
445             plane[i][j].value = 0
446
447     aisle = [4,11,18,25]
448     for i in aisle:
449         for j in range(0,15):
450             if(plane[i][j].value==1):
451                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
452                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
453                     continue
454                 if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
][plane[i][j+1].pass_seat].bag==0):
455                     p1r = plane[i][j+1].pass_char
456                     p1c = plane[i][j+1].pass_seat
457                     p1p = plane[i][j+1].pass_pri
458                     plane[i][j].pass_char = p1r
459                     plane[i][j].pass_seat = p1c
460                     plane[i][j].pass_pri = p1p
461                     plane[i][j].value = 1
462                     plane[i][j+1].pass_char = 0
463                     plane[i][j+1].pass_seat = 0
464                     plane[i][j+1].pass_pri = 0
465                     plane[i][j+1].value = 0
466                     continue

```

```

467         if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
468         ][plane[i][j+1].pass_seat].bag!=0):
469             continue
470         if(j==0):
471             continue
472         if(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1 and
473         plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
474             ch = random.choice([i-1,i+1])
475             p1r = plane[ch][j].pass_char
476             p1c = plane[ch][j].pass_seat
477             p1p = plane[ch][j].pass_pri
478             plane[i][j].pass_char = p1r
479             plane[i][j].pass_seat = p1c
480             plane[i][j].pass_pri = p1p
481             plane[i][j].value = 1
482             plane[ch][j].pass_char = 0
483             plane[ch][j].pass_seat = 0
484             plane[ch][j].pass_pri = 0
485             plane[ch][j].value = 0
486         elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
487             p1r = plane[i-1][j].pass_char
488             p1c = plane[i-1][j].pass_seat
489             p1p = plane[i-1][j].pass_pri
490             plane[i][j].pass_char = p1r
491             plane[i][j].pass_seat = p1c
492             plane[i][j].pass_pri = p1p
493             plane[i][j].value = 1
494             plane[i-1][j].pass_char = 0
495             plane[i-1][j].pass_seat = 0
496             plane[i-1][j].pass_pri = 0
497             plane[i-1][j].value = 0
498         elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
499             p1r = plane[i+1][j].pass_char
500             p1c = plane[i+1][j].pass_seat
501             p1p = plane[i+1][j].pass_pri
502             plane[i][j].pass_char = p1r
503             plane[i][j].pass_seat = p1c
504             plane[i][j].pass_pri = p1p
505             plane[i][j].value = 1
506             plane[i+1][j].pass_char = 0
507             plane[i+1][j].pass_seat = 0
508             plane[i+1][j].pass_pri = 0
509             plane[i+1][j].value = 0
510         else:
511             continue
512
513     left_1 = [3,10,17,24]
514     for i in left_1:
515         for j in range(1,15):
516             if(plane[i][j].value==0 and plane[i-1][j].value==1):
517                 p1r = plane[i-1][j].pass_char
518                 p1c = plane[i-1][j].pass_seat
519                 p1p = plane[i-1][j].pass_pri
520                 plane[i][j].pass_char = p1r
521                 plane[i][j].pass_seat = p1c
522                 plane[i][j].pass_pri = p1p
523                 plane[i][j].value = 1
524                 plane[i-1][j].pass_char = 0
525                 plane[i-1][j].pass_seat = 0
526                 plane[i-1][j].pass_pri = 0

```

```
525         plane[i-1][j].value = 0
526
527     right_1 = [5,12,19,26]
528     for i in right_1:
529         for j in range(1,15):
530             if(plane[i][j].value==0 and plane[i+1][j].value==1):
531                 p1r = plane[i+1][j].pass_char
532                 p1c = plane[i+1][j].pass_seat
533                 p1p = plane[i+1][j].pass_pri
534                 plane[i][j].pass_char = p1r
535                 plane[i][j].pass_seat = p1c
536                 plane[i][j].pass_pri = p1p
537                 plane[i][j].value = 1
538                 plane[i+1][j].pass_char = 0
539                 plane[i+1][j].pass_seat = 0
540                 plane[i+1][j].pass_pri = 0
541                 plane[i+1][j].value = 0
542
543     left_2 = [2,9,16,23]
544     for i in left_2:
545         for j in range(1,15):
546             if(plane[i][j].value==0 and plane[i-1][j].value==1):
547                 p1r = plane[i-1][j].pass_char
548                 p1c = plane[i-1][j].pass_seat
549                 p1p = plane[i-1][j].pass_pri
550                 plane[i][j].pass_char = p1r
551                 plane[i][j].pass_seat = p1c
552                 plane[i][j].pass_pri = p1p
553                 plane[i][j].value = 1
554                 plane[i-1][j].pass_char = 0
555                 plane[i-1][j].pass_seat = 0
556                 plane[i-1][j].pass_pri = 0
557                 plane[i-1][j].value = 0
558
559     right_2 = [6,13,20,27]
560     for i in right_2:
561         for j in range(1,15):
562             if(plane[i][j].value==0 and plane[i+1][j].value==1):
563                 p1r = plane[i+1][j].pass_char
564                 p1c = plane[i+1][j].pass_seat
565                 p1p = plane[i+1][j].pass_pri
566                 plane[i][j].pass_char = p1r
567                 plane[i][j].pass_seat = p1c
568                 plane[i][j].pass_pri = p1p
569                 plane[i][j].value = 1
570                 plane[i+1][j].pass_char = 0
571                 plane[i+1][j].pass_seat = 0
572                 plane[i+1][j].pass_pri = 0
573                 plane[i+1][j].value = 0
574
575     """print("time",time)
576     print("check",check)
577
578     for i in range(0,347):
579         for j in range(0,17):
580             #print(plane[i][j].value)
581             a[i][j] = plane[i][j].value
582     plt.figure('time'+str(time))
583     im = plt.imshow(a[0:29])
584     ax = plt.gca()
```

```

585     ax.set_xticks(np.arange(-.5, 17, 1), minor=True)
586     ax.set_yticks(np.arange(-.5, 29, 1), minor=True)
587     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
588     plt.savefig('time'+str(time)+'.png')"""
589     return time
590
591 print(run2(1,0,0,318))
592 print(run2(2,0,0,318))

```

B.6 Codes of Disembarking Process, Airplane III

```

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 run3(case,RL,RJ,N):
12     # Create position of people in "Narrow Body" Passenger Aircraft
13     arr = np.arange(1,243)
14     list_0 = arr.tolist()
15     for i in range (28):
16         list_0[i] = [math.floor(list_0[i]/14)+1, (list_0[i]%14)]
17     for i in range (28,67):
18         list_0[i] = [math.floor((list_0[i]-28)/13)+4, ((list_0[i]-28)%13)]
19     for i in range (67,95):
20         list_0[i] = [math.floor((list_0[i]-67)/14)+8, ((list_0[i]-67)%14)]
21     for i in range (95,137):
22         list_0[i] = [math.floor((list_0[i]-95)/21)+1, ((list_0[i]-95)%21)+19]
23     for i in range (137,200):
24         list_0[i] = [math.floor((list_0[i]-137)/21)+4, ((list_0[i]-137)%21)+19]
25     for i in range (200,242):
26         list_0[i] = [math.floor((list_0[i]-200)/21)+8, ((list_0[i]-200)%21)+19]
27     for i in range (242):
28         if (list_0[i][1] == 0) | (list_0[i][1] == 19):
29             list_0[i] = [list_0[i-1][0],list_0[i-1][1]+1]
30
31
32     # Random luggage stow time of each people by experimental data and Weibull
33     # distribution
34     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,
35     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,
36     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,
37     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,
38     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,
39     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,
40     7.5]
41     luggage_arr = np.array(luggage)
42     mean = luggage_arr.mean()
43     std = stdev(luggage_arr)
44     k = (std/mean)**(-1.086)
45     z = 1 + 1/k
46     def f(x):
47         return math.exp(-x)*(x**(z-1))
48     gamma,err = quad(f, 0, math.inf)
49     c = mean/gamma

```



```
43 for i in range (len(list_0)):
44     weibull = (c*(np.random.weibull(k, 1))).tolist()
45     time = round((weibull[0]/1.42))
46     list_0[i].append(time)
47
48 #Case1
49 list_1 = copy.deepcopy(list_0)
50 pri_max_1 = 1
51
52 for i in range (len(list_1)):
53     list_1[i].append(1)
54
55 list_late_1 = random.sample(list_1,round(N*RL))
56
57 for i in range (len(list_late_1)):
58     for j in range (len(list_1)):
59         if list_late_1[i] == list_1[j]:
60             list_1[j][3] = pri_max_1+1
61             if (list_1[j][0] == 4) or (list_1[j][0] == 6):
62                 for k in range (len(list_1)):
63                     if (list_1[k][1] == list_1[j][1]) & (list_1[k][0] == 5):
64                         list_1[k][3] = pri_max_1+1
65
66 class agent_1:
67     def __init__(self, char, seat, bag, pri):
68         self.char = char
69         self.seat = seat
70         self.bag = bag
71         self.pri = pri
72
73 passenger_1 = []
74 for i in range(len(list_1)):
75     passenger_1.append(agent_1(list_1[i][0], list_1[i][1], list_1[i][2], list_1[
76 i][3]))
77
78 #Case2
79 list_2 = copy.deepcopy(list_0)
80 pri_max_2 = 4
81
82 for i in range (len(list_2)):
83     if (list_2[i][0] == 1) or (list_2[i][0] == 5):
84         list_2[i].append(4)
85     elif (list_2[i][0] == 2) or (list_2[i][0] == 4):
86         list_2[i].append(3)
87     elif (list_2[i][0] == 9):
88         list_2[i].append(2)
89     elif (list_2[i][0] == 6) or (list_2[i][0] == 8):
90         list_2[i].append(1)
91
92 list_late_2 = random.sample(list_2,round(N*RL))
93
94 for i in range (len(list_late_2)):
95     for j in range (len(list_2)):
96         if list_late_2[i] == list_2[j]:
97             list_2[j][3] = pri_max_2+1
98             if (list_2[j][0] == 4) or (list_2[j][0] == 6):
99                 for k in range (len(list_2)):
100                     if (list_2[k][1] == list_2[j][1]) & (list_2[k][0] == 5):
101                         list_2[k][3] = pri_max_2+1
```

```
102     class agent_2:
103         def __init__(self, char, seat, bag, pri):
104             self.char = char
105             self.seat = seat
106             self.bag = bag
107             self.pri = pri
108
109     passenger_2 = []
110     for i in range(len(list_2)):
111         passenger_2.append(agent_2(list_2[i][0], list_2[i][1], list_2[i][2], list_2[
112 i][3]))
113
114     #Case3
115     list_3 = copy.deepcopy(list_0)
116     pri_max_3 = 4
117
118     for i in range (len(list_3)):
119         if (list_3[i][0] == 1) | (list_3[i][0] == 5):
120             list_3[i].append(4)
121         if (list_3[i][0] == 9) & (1 <= list_3[i][1] <= 14):
122             list_3[i].append(4)
123
124         if (list_3[i][0] == 2) | (list_3[i][0] == 4):
125             list_3[i].append(3)
126         if (list_3[i][0] == 6) & (1 <= list_3[i][0] <= 13):
127             list_3[i].append(3)
128
129         if (list_3[i][0] == 8) & (1 <= list_3[i][0] <= 14):
130             list_3[i].append(2)
131         if (list_3[i][0] == 9) & (20 <= list_3[i][1] <= 40):
132             list_3[i].append(2)
133
134         if (list_3[i][0] == 6) & (20 <= list_3[i][0] <= 40):
135             list_3[i].append(1)
136         if (list_3[i][0] == 8) & (20 <= list_3[i][0] <= 40):
137             list_3[i].append(1)
138
139     list_late_3 = random.sample(list_3, round(N*RL))
140
141     for i in range (len(list_late_3)):
142         for j in range (len(list_3)):
143             if list_late_3[i] == list_3[j]:
144                 list_3[j][3] = pri_max_3+1
145                 if (list_3[j][0] == 4) or (list_3[j][0] == 6):
146                     for k in range (len(list_3)):
147                         if (list_3[k][1] == list_3[j][1]) & (list_3[k][0] == 5):
148                             list_3[k][3] = pri_max_3+1
149
150     class agent_3:
151         def __init__(self, char, seat, bag, pri):
152             self.char = char
153             self.seat = seat
154             self.bag = bag
155             self.pri = pri
156
157     passenger_3 = []
158     for i in range(len(list_3)):
159         passenger_3.append(agent_3(list_3[i][0], list_3[i][1], list_3[i][2], list_3[
160 i][3]))
```

```

160 #Case4
161 list_4 = copy.deepcopy(list_0)
162 pri_max_4 = 4
163
164 for i in range (len(list_4)):
165     if (list_4[i][0] == 1) | (list_4[i][0] == 5):
166         list_4[i].append(4)
167     if (list_4[i][0] == 9) & (1 <= list_4[i][1] <= 14):
168         list_4[i].append(2)
169     if (list_4[i][0] == 2) | (list_4[i][0] == 4):
170         list_4[i].append(3)
171     if (list_4[i][0] == 6) & (1 <= list_4[i][0] <= 13):
172         list_4[i].append(1)
173     if (list_4[i][0] == 8) & (1 <= list_4[i][0] <= 14):
174         list_4[i].append(1)
175     if (list_4[i][0] == 9) & (20 <= list_4[i][1] <= 40):
176         list_4[i].append(4)
177     if (list_4[i][0] == 6) & (20 <= list_4[i][0] <= 40):
178         list_4[i].append(3)
179     if (list_4[i][0] == 8) & (20 <= list_4[i][0] <= 40):
180         list_4[i].append(3)
181
182 list_late_4 = random.sample(list_4,round(N*RL))
183
184 for i in range (len(list_late_4)):
185     for j in range (len(list_4)):
186         if list_late_4[i] == list_4[j]:
187             list_4[j][3] = pri_max_4+1
188             if (list_4[j][0] == 4) or (list_4[j][0] == 6):
189                 for k in range (len(list_4)):
190                     if (list_4[k][1] == list_4[j][1]) & (list_4[k][0] == 5):
191                         list_4[k][3] = pri_max_4+1
192
193 class agent_4:
194     def __init__(self, char, seat, bag, pri):
195         self.char = char
196         self.seat = seat
197         self.bag = bag
198         self.pri = pri
199
200 passenger_4 = []
201 for i in range(len(list_4)):
202     passenger_4.append(agent_4(list_4[i][0], list_4[i][1], list_4[i][2], list_4[
i][3]))
203
204 class person:
205     def __init__(self, char, seat, bag, pri):
206         self.char = char
207         self.seat = seat
208         self.bag = bag
209         self.pri = pri
210
211 list_pass = [[person(0,0,0,0) for i in range(0,41)] for j in range(0,10)]
212
213 class grid:
214     def __init__(self, type, value, pass_char, pass_seat, pass_pri):
215         self.type = type
216         # -1 -> block
217         # 0 -> cabin
218         # 1 -> queue

```

```
219     # 2 -> aisle
220     # 3 -> seat
221     self.value = value
222     # 0 -> available
223     # 1 -> passenger
224     self.pass_char = pass_char
225     self.pass_seat = pass_seat
226     self.pass_pri = pass_pri
227
228 plane = [[grid(-1,0,0,0,0) for i in range(0,42)] for j in range(0,252)]
229
230 for i in range(1,3):
231     for j in range(1,15):
232         plane[i][j].type = 3
233
234 for i in range(4,7):
235     for j in range(1,14):
236         plane[i][j].type = 3
237
238 for i in range(8,10):
239     for j in range(1,15):
240         plane[i][j].type = 3
241
242 for i in range(1,3):
243     for j in range(20,41):
244         plane[i][j].type = 3
245
246 for i in range(4,7):
247     for j in range(20,41):
248         plane[i][j].type = 3
249
250 for i in range(8,10):
251     for j in range(20,41):
252         plane[i][j].type = 3
253
254 for i in range(3,252):
255     plane[i][0].type = 1
256     plane[i][41].type = 1
257
258 q = [3,7]
259 for i in q:
260     for j in range(0,17):
261         plane[i][j].type = 2
262     for j in range(18,42):
263         plane[i][j].type = 2
264
265 for i in range(1,10):
266     plane[i][17].type = 0
267
268 def C1(passenger_1):
269     for i in range(len(passenger_1)):
270         list_pass[passenger_1[i].char][passenger_1[i].seat].char =
passenger_1[i].char
271         list_pass[passenger_1[i].char][passenger_1[i].seat].seat =
passenger_1[i].seat
272         list_pass[passenger_1[i].char][passenger_1[i].seat].bag = passenger_1
[i].bag
273         list_pass[passenger_1[i].char][passenger_1[i].seat].pri = passenger_1
[i].pri
274         plane[passenger_1[i].char][passenger_1[i].seat].value = 1
```

```
275     plane[passenger_1[i].char][passenger_1[i].seat].pass_char =
passenger_1[i].char
276     plane[passenger_1[i].char][passenger_1[i].seat].pass_seat =
passenger_1[i].seat
277     plane[passenger_1[i].char][passenger_1[i].seat].pass_pri =
passenger_1[i].pri
278     def C2(passenger_2):
279         for i in range(len(passenger_2)):
280             list_pass[passenger_2[i].char][passenger_2[i].seat].char =
passenger_2[i].char
281             list_pass[passenger_2[i].char][passenger_2[i].seat].seat =
passenger_2[i].seat
282             list_pass[passenger_2[i].char][passenger_2[i].seat].bag = passenger_2
[i].bag
283             list_pass[passenger_2[i].char][passenger_2[i].seat].pri = passenger_2
[i].pri
284             plane[passenger_2[i].char][passenger_2[i].seat].value = 1
285             plane[passenger_2[i].char][passenger_2[i].seat].pass_char =
passenger_2[i].char
286             plane[passenger_2[i].char][passenger_2[i].seat].pass_seat =
passenger_2[i].seat
287             plane[passenger_2[i].char][passenger_2[i].seat].pass_pri =
passenger_2[i].pri
288     def C3(passenger_3):
289         for i in range(len(passenger_3)):
290             list_pass[passenger_3[i].char][passenger_3[i].seat].char =
passenger_3[i].char
291             list_pass[passenger_3[i].char][passenger_3[i].seat].seat =
passenger_3[i].seat
292             list_pass[passenger_3[i].char][passenger_3[i].seat].bag = passenger_3
[i].bag
293             list_pass[passenger_3[i].char][passenger_3[i].seat].pri = passenger_3
[i].pri
294             plane[passenger_3[i].char][passenger_3[i].seat].value = 1
295             plane[passenger_3[i].char][passenger_3[i].seat].pass_char =
passenger_3[i].char
296             plane[passenger_3[i].char][passenger_3[i].seat].pass_seat =
passenger_3[i].seat
297             plane[passenger_3[i].char][passenger_3[i].seat].pass_pri =
passenger_3[i].pri
298     def C4(passenger_4):
299         for i in range(len(passenger_4)):
300             list_pass[passenger_4[i].char][passenger_4[i].seat].char =
passenger_4[i].char
301             list_pass[passenger_4[i].char][passenger_4[i].seat].seat =
passenger_4[i].seat
302             list_pass[passenger_4[i].char][passenger_4[i].seat].bag = passenger_4
[i].bag
303             list_pass[passenger_4[i].char][passenger_4[i].seat].pri = passenger_4
[i].pri
304             plane[passenger_4[i].char][passenger_4[i].seat].value = 1
305             plane[passenger_4[i].char][passenger_4[i].seat].pass_char =
passenger_4[i].char
306             plane[passenger_4[i].char][passenger_4[i].seat].pass_seat =
passenger_4[i].seat
307             plane[passenger_4[i].char][passenger_4[i].seat].pass_pri =
passenger_4[i].pri
308
309     if(case==1):
310         C1(passenger_1)
```

```
311     if(case==2):
312         C1(passenger_2)
313     if(case==3):
314         C3(passenger_3)
315     if(case==4):
316         C4(passenger_4)
317
318     time = 0
319
320     a = [[0 for i in range(0,42)] for i in range(0,252)]
321
322     for i in range(0,252):
323         for j in range(0,42):
324             #print(plane[i][j].value)
325             a[i][j] = plane[i][j].value
326
327     time = 0
328
329     plt.figure('time'+str(time))
330     im = plt.imshow(a[0:10])
331     ax = plt.gca()
332     ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
333     ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
334     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
335     plt.savefig("figure_3_d.png")
336     #plt.show()
337
338     def check_pass(plane):
339         check = 0
340         for w in range(0,10):
341             for z in range(0,42):
342                 if(plane[w][z].type>0 and plane[w][z].value == 1):
343                     check+=1
344         return check
345
346     def min_pri(plane):
347         x = 10
348         for w in range(0,10):
349             for z in range(0,42):
350                 if(plane[w][z].type>0):
351                     if(plane[w][z].value==1):
352                         if(plane[w][z].pass_pri<x):
353                             x = plane[w][z].pass_pri
354         return x
355
356     left_1 = [2,6]
357     for i in left_1:
358         for j in range(1,41):
359             if(plane[i][j].value==0 and plane[i-1][j].value==1):
360                 p1r = plane[i-1][j].pass_char
361                 p1c = plane[i-1][j].pass_seat
362                 p1p = plane[i-1][j].pass_pri
363                 plane[i][j].pass_char = p1r
364                 plane[i][j].pass_seat = p1c
365                 plane[i][j].pass_pri = p1p
366                 plane[i][j].value = 1
367                 plane[i-1][j].pass_char = 0
368                 plane[i-1][j].pass_seat = 0
369                 plane[i-1][j].pass_pri = 0
370                 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

```

```

431     plane[i+1][j].pass_char = p1r
432     plane[i+1][j].pass_seat = p1c
433     plane[i+1][j].pass_pri = p1p
434     plane[i+1][j].value = 1
435     plane[i][j].pass_char = 0
436     plane[i][j].pass_seat = 0
437     plane[i][j].pass_pri = 0
438     plane[i][j].value = 0
439
440     aisle = [3,7]
441     #front
442     for i in aisle:
443         for j in range(0,15):
444             if(plane[i][j].value==1):
445                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
446                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag-=1
447                     continue
448                 if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
][plane[i][j+1].pass_seat].bag==0):
449                     p1r = plane[i][j+1].pass_char
450                     p1c = plane[i][j+1].pass_seat
451                     p1p = plane[i][j+1].pass_pri
452                     plane[i][j].pass_char = p1r
453                     plane[i][j].pass_seat = p1c
454                     plane[i][j].pass_pri = p1p
455                     plane[i][j].value = 1
456                     plane[i][j+1].pass_char = 0
457                     plane[i][j+1].pass_seat = 0
458                     plane[i][j+1].pass_pri = 0
459                     plane[i][j+1].value = 0
460                     continue
461                 if(plane[i][j+1].value == 1 and list_pass[plane[i][j+1].pass_char
][plane[i][j+1].pass_seat].bag!=0):
462                     continue
463                 if(j==0):
464                     continue
465                 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):
466                     ch = random.choice([i-1,i+1])
467                     p1r = plane[ch][j].pass_char
468                     p1c = plane[ch][j].pass_seat
469                     p1p = plane[ch][j].pass_pri
470                     plane[i][j].pass_char = p1r
471                     plane[i][j].pass_seat = p1c
472                     plane[i][j].pass_pri = p1p
473                     plane[i][j].value = 1
474                     plane[ch][j].pass_char = 0
475                     plane[ch][j].pass_seat = 0
476                     plane[ch][j].pass_pri = 0
477                     plane[ch][j].value = 0
478                 elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
479                     p1r = plane[i-1][j].pass_char
480                     p1c = plane[i-1][j].pass_seat
481                     p1p = plane[i-1][j].pass_pri
482                     plane[i][j].pass_char = p1r
483                     plane[i][j].pass_seat = p1c
484                     plane[i][j].pass_pri = p1p
485                     plane[i][j].value = 1

```



```

486         plane[i-1][j].pass_char = 0
487         plane[i-1][j].pass_seat = 0
488         plane[i-1][j].pass_pri = 0
489         plane[i-1][j].value = 0
490         elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
491             p1r = plane[i+1][j].pass_char
492             p1c = plane[i+1][j].pass_seat
493             p1p = plane[i+1][j].pass_pri
494             plane[i][j].pass_char = p1r
495             plane[i][j].pass_seat = p1c
496             plane[i][j].pass_pri = p1p
497             plane[i][j].value = 1
498             plane[i+1][j].pass_char = 0
499             plane[i+1][j].pass_seat = 0
500             plane[i+1][j].pass_pri = 0
501             plane[i+1][j].value = 0
502         else:
503             continue
504     #back
505     for i in aisle:
506         for j in reversed(range(20,42)):
507             if(plane[i][j].value==1):
508                 if(list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag>0):
509                     list_pass[plane[i][j].pass_char][plane[i][j].pass_seat].
bag -=1
510                 continue
511             if(plane[i][j-1].value == 1 and list_pass[plane[i][j-1].pass_char
][plane[i][j-1].pass_seat].bag==0):
512                 p1r = plane[i][j-1].pass_char
513                 p1c = plane[i][j-1].pass_seat
514                 p1p = plane[i][j-1].pass_pri
515                 plane[i][j].pass_char = p1r
516                 plane[i][j].pass_seat = p1c
517                 plane[i][j].pass_pri = p1p
518                 plane[i][j].value = 1
519                 plane[i][j-1].pass_char = 0
520                 plane[i][j-1].pass_seat = 0
521                 plane[i][j-1].pass_pri = 0
522                 plane[i][j-1].value = 0
523                 continue
524             if(plane[i][j-1].value == 1 and list_pass[plane[i][j-1].pass_char
][plane[i][j-1].pass_seat].bag!=0):
525                 continue
526             if(j==41):
527                 continue
528             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):
529                 ch = random.choice([i-1,i+1])
530                 p1r = plane[ch][j].pass_char
531                 p1c = plane[ch][j].pass_seat
532                 p1p = plane[ch][j].pass_pri
533                 plane[i][j].pass_char = p1r
534                 plane[i][j].pass_seat = p1c
535                 plane[i][j].pass_pri = p1p
536                 plane[i][j].value = 1
537                 plane[ch][j].pass_char = 0
538                 plane[ch][j].pass_seat = 0
539                 plane[ch][j].pass_pri = 0
540                 plane[ch][j].value = 0

```

```
541         elif(plane[i-1][j].pass_pri==x and plane[i-1][j].value == 1):
542             p1r = plane[i-1][j].pass_char
543             p1c = plane[i-1][j].pass_seat
544             p1p = plane[i-1][j].pass_pri
545             plane[i][j].pass_char = p1r
546             plane[i][j].pass_seat = p1c
547             plane[i][j].pass_pri = p1p
548             plane[i][j].value = 1
549             plane[i-1][j].pass_char = 0
550             plane[i-1][j].pass_seat = 0
551             plane[i-1][j].pass_pri = 0
552             plane[i-1][j].value = 0
553         elif(plane[i+1][j].pass_pri==x and plane[i+1][j].value == 1):
554             p1r = plane[i+1][j].pass_char
555             p1c = plane[i+1][j].pass_seat
556             p1p = plane[i+1][j].pass_pri
557             plane[i][j].pass_char = p1r
558             plane[i][j].pass_seat = p1c
559             plane[i][j].pass_pri = p1p
560             plane[i][j].value = 1
561             plane[i+1][j].pass_char = 0
562             plane[i+1][j].pass_seat = 0
563             plane[i+1][j].pass_pri = 0
564             plane[i+1][j].value = 0
565         else:
566             continue
567     left_1 = [2,6]
568     for i in left_1:
569         for j in range(1,41):
570             if(plane[i][j].value==0 and plane[i-1][j].value==1):
571                 p1r = plane[i-1][j].pass_char
572                 p1c = plane[i-1][j].pass_seat
573                 p1p = plane[i-1][j].pass_pri
574                 plane[i][j].pass_char = p1r
575                 plane[i][j].pass_seat = p1c
576                 plane[i][j].pass_pri = p1p
577                 plane[i][j].value = 1
578                 plane[i-1][j].pass_char = 0
579                 plane[i-1][j].pass_seat = 0
580                 plane[i-1][j].pass_pri = 0
581                 plane[i-1][j].value = 0
582
583     right_1 = [4,8]
584     for i in right_1:
585         for j in range(1,41):
586             if(plane[i][j].value==0 and plane[i+1][j].value==1):
587                 p1r = plane[i+1][j].pass_char
588                 p1c = plane[i+1][j].pass_seat
589                 p1p = plane[i+1][j].pass_pri
590                 plane[i][j].pass_char = p1r
591                 plane[i][j].pass_seat = p1c
592                 plane[i][j].pass_pri = p1p
593                 plane[i][j].value = 1
594                 plane[i+1][j].pass_char = 0
595                 plane[i+1][j].pass_seat = 0
596                 plane[i+1][j].pass_pri = 0
597                 plane[i+1][j].value = 0
598
599     """print("time",time)
600     print("check",check)
```

```
601     print(x)
602     for i in range(0,250):
603         for j in range(0,42):
604             #print(plane[i][j].value)
605             a[i][j] = plane[i][j].value
606     plt.figure('time'+str(time))
607     im = plt.imshow(a[0:10])
608     ax = plt.gca()
609     ax.set_xticks(np.arange(-.5, 42, 1), minor=True)
610     ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
611     ax.grid(which='minor', color='w', linestyle='-', linewidth=2)
612     plt.savefig('time'+str(time)+'.png')"""
613     return time
614
615 print(run3(1,0,0,242))
616 print(run3(2,0,0,242))
617 print(run3(3,0,0,242))
618 print(run3(4,0,0,242))
```