Summary

Nowadays, millions of people always like to fly from place to place, either for business or for holiday. Airlines have adopted a variety of boarding methods to reduce the gate turnaround time for airplanes, which includes the boarding and disembarking time. Significant reductions in gate delays would improve on the quality of life for long-suffering air travelers, and yield significant economic benefits from more efficient use of aircraft and airport infrastructure.[1]

For task one, we construct a unified model to calculate the total time of delay, including boarding and disembarking time. In particular, we consider different kinds of boarding methods with the influence of different number of bags passengers carrying and the percentage of people disobey the instruction. In the boarding part, the baggage packing time and the time to sit down is more important than other factors. For safety, we need to make sure everybody is well seated until the plane departs. Thus, we focus only on the last person to sit down and trace his road and calculate the total time he blocked by others. We call that tracing model. For the disembarking part, normally people get out the plane well organized by seat, which is easy to count. While for real reason, there are always some people jump the queue and make the delay of other people. Here we construct a jump-queue model.

For task two, with this tracing model in the narrow body aircraft, we compare three normal boarding methods, concluding that boarding by seat is better than boarding by section as well as random boarding. When they carry more bags, the boarding time will increase significantly and boarding by seat is still the best. While with the increase of people disobeying the instruction, random boarding is better. We also construct two new boarding methods, which is back-to-front polices [4] and Steffen method [8]. It's proved in the article that Steffen method is always better.

For task three, we modify our model for different shape of airplanes and calculate their boarding and disembarking time. The Steffen method is still the best.

For task four, our model can still be used to calculate the delay time when the number of passengers is limited to some percentage.

For task five, we have written a letter to the manager of airplane with convincing reasons to adopt Steffen method in most airlines. Our results are both examined with tracing model and discrete event simulation [2,3,4].

Table of Content

1. Introduction

1.1 Background

In air transport, efficiency is time and time is money. Studies have shown that every 1 minute's reduction in flight turnaround time can save about $ 30 . The turnaround time of an aircraft includes the time for disembarking passengers on board, the boarding time for new passengers, and the time for airport staff to check the status of the aircraft and transfer passengers' checked luggage. The time-consuming of the latter two operations is basically unchanged, while the time-consuming of the first two operations is largely determined by the passenger's behavior. For airlines, the reduction in turnaround time can increase the number of flights and increase the number of aircrafts in the parking lot ; The boarding process is easier to perceive and the boarding experience is better.

In addition to the normal walking time of passengers, the boarding and disembarking process of the plane also includes the time for passengers to place their luggage and move away from their seats to let others get to their seats. Passengers in the back are blocked when passengers in the front leave luggage or give up their seats, resulting in increased boarding time. Therefore, by boarding in a reasonable order, passengers can be blocked for less time, thereby minimizing the boarding time of the aircraft.

In the following, we will use a mathematical model to figure out the shortest boarding and disembarking strategies to provide airlines with a feasible boarding plan.

1.2 Question restatement

In order to improve the boarding efficiency and save the airline's economic loss and time loss, we will establish a mathematical model to calculate the boarding and disembarking time required for various boarding strategies.

To test whether our model is indeed usable, we will perform an analysis of the accuracy, generality and sensitivity of the model.

Our model will practically calculate the boarding time of six kinds of boarding strategies which are "random boarding", "segment boarding", "seat boarding", "inverted pyramid boarding" and "steffen boarding strategy" . After that, we will find the best strategy based on the data.

In order to verify the generality of the model, we will use the model to calculate the boarding time required for some boarding strategies on different types of aircraft, such as flying-wing aircraft with relatively wide and short cabins and more special dual-aisle dual-entry aircraft .

We will also perform a sensitivity analysis on our model by calculating the stability of our model in some extreme cases. For example, sometimes the attendance rate is limited to 30 % , 50 % or 70 % , we will verify whether the calculation result of our model is still accurate.

2. core model

2.1 Basic "Tracking Model"

2.1.1 The role of the model

To calculate the optimal boarding strategy, we build a mathematical model, hereinafter referred to as the "tracking model". Due to the large number of passengers and their random behavior, it is not realistic to calculate the time for each seat situation one by one, so we select the last passenger who boarded the plane for tracking. We will call him A hereinafter.

2.1.2 Analysis of the model

For the convenience of calculation, we divide the aisle in the cabin into 33 grids, denoted as G_1, G_2, G_3, G_4 ……G_{32}, G_{33}. Each grid corresponds to a row of seats (there are 33 rows of seats in total for a narrow-body passenger aircraft), and each compartment can only accommodate one passenger. In the same way, the aisle outside the cabin can be divided into 198 grids (the first passenger's grid is recorded as G_0, and the corner mark of the next passenger's grid is minus by one in turn), namely G_{-197}, G_{-196}, G_{-195}, G_{-194} ……G_1, G_0 (narrow-body

passenger aircrafts can accommodate 198 passengers when fully loaded). The boarding time of an airplane refers to the time period from when the first passenger enters the cabin to the last passenger seated. Therefore, taking tracker $\text{PERSON}_{\text{LAST}}$ as an example, this period of time can be divided into the time he normally walks and the time he is blocked by the previous passenger who stops because of stowing luggage.

2.1.3 Analysis of Tracking Model

When calculating the time that $\text{PERSON}_{\text{LAST}}$ a was delayed by the passenger in front of him due to putting his luggage and move away from his seat to let others get to their seats. We need to know which passenger is delaying the effective time (that is, the passenger who was delayed the longest, we only need to calculate his time, and the other A passenger was seated before the end). If the time for each passenger to put their bags is added, double-counting problems occur because there may be multiple passengers putting bags at the same time or the time periods partially overlap.

If two passengers A and B before $\text{PERSON}_{\text{LAST}}$ a put their luggage at the same time (A is in front of B), passenger A will not directly affect $\text{PERSON}_{\text{LAST}}$ a because only the passenger B in the back really blocks $\text{PERSON}_{\text{LAST}}$ a, that is, B is finished Bag, after taking a seat, A has already taken a seat. So just count the time of the passenger in the aisle who blocks the B closest to $\text{PERSON}_{\text{LAST}}$ a. It is assumed here that each person puts the bag for the same time and carries the same number of backpacks.

Assuming that 5 seconds later, another passenger stops between $\text{PERSON}_{\text{LAST}}$ a and B with passenger C, then passenger B who was just behind is not the last. So for this case, we only count the time spent in the aisle when C is the "passenger closest to $\text{PERSON}_{\text{LAST}}$ a ".

If the queue is no longer blocked after the passenger closest to $\text{PERSON}_{\text{LAST}}$ a is seated, stop the timer until someone blocking the queue appears again.

2.1.4 The formula derived from the model

From 2.1.2 , we derive a set of methods for calculating the boarding time T:

1. ***A(p)= p.row+ p.number***

(This is a function that calculates the distance between each passenger and his location, p is a passenger)

2. ***S(x)=min{a(x)}***

(This is a function that finds the first passenger with luggage, x is the set of some passengers)

3. ***i(x)= s (x-s(x)).row -s(x).row***

(At this time, ***s (x-s(x)).row-s(x-s(x))***). ***nowrow =1*** , that is, passenger 2 and passenger 3 in Figure 1 start to put luggage at the same time)

或 ***i (x)=a(s(x-s(x)))-a(s(x))+ OccTime(S(x))***

(At this time, ***s (x-s(x)).row-s(x-s(x)*** . ***nowrow ≠1***, that is, passenger 2 and passenger 4 in Figure 2 start to put luggage at different times)

4. ***m(x) = OccTime(S(x)) - i (x)***

(at this time ***OccTime(S(x))*** $\geq$***i (x)***).

or ***m(x) = OccTime(S(x))***

(At this time , ***OccTime(S(x))*** $<$ ***i (x)*** , this function is used to calculate the time when a is blocked when B n puts luggage).

Then $\boldsymbol{TIME_{TOTAL}} = \sum_{i=1}^{len} \boldsymbol{M}\ \ (x - \{\sum_{j=1}^{i} \boldsymbol{person(No.\ j)}\ \})$

For the obstruction of the rear passengers caused by the passengers moving away from their seats to let others get to their seats when boarding, we can follow the idea of the tracking model. Since the seats of the two passengers who gave up their seats must be in the same row, and the seat of the last occupant must be closer to the window than the first occupant, we can record the two passengers as I,J. Let the number of seat rows of I be Q_I and the number of columns to be P_I . Similarly, the number of seat rows and columns of j are respectively Q_J and P_J . (When A(j) <A(i)).

When $Q_I = Q_J$, two passengers may need to give up their seats. For the convenience of calculation, we denote the columns A, B, C, D, E, and F of the narrow-body aircraft as 3, 2, 1, -1, -2, -3 respectively (the aisle is 0). Then the relationship between P_I and P_J has five situations:

1. $\boldsymbol{P_I < P_J}$, both P i and P j are positive, no need to give up your seat at this time, ***T rang (i)=0***.

2. $P_I > P_J$, both P i and P j are negative, no need to give up your seat at this time, ***T rang (i)=0***.
3. P_I , P_J one is positive and one is negative, no need to give up your seat at this time, ***T rang (i)=0.***
4. $P_I > P_J$, P i and P j are both positive, at this time T rang (i)=(P i -P j)T rang
5. $P_I < P_J$, P_i and P_j are both negative, at this time ***T rang (i)=(P j -P i)T rang .***

Then ***OccTime(S(x))*** = ***pre_bag_time total + T rang (i)***.

$$T = OccTime(S(x)) + \frac{197+\text{number of rows for the last passenger}}{v}.$$

2.2 Basic "queue-cutting model"

In general, the order in which passengers get off the plane is fixed. Since the passengers by the aisle must get off the plane before the passengers by the window, and the aisles are only one person wide, we can assume that the disembarkation method is column C-column D-column B-column E-column A-column F (for narrow-body airliners from From left to right are columns A, B, C, D, E, F), that is, passengers in column C put their luggage together first, and after they get off the plane in turn, passengers in column D put their luggage together, and so on. Figure 3 is a schematic diagram of the dismounting sequence.

	Column A	Column B	Column C	aisle	Column D	Column E	Column F
line 1	5	3	1		2	4	6
line 2	5	3	1		2	4	6
...							
line 31_	5	3	1		2	4	6
line 32_	5	3	1		2	4	6
line 33	5	3	1		2	4	6

_							

image 3

But there are not a few passengers who refuse to board the plane according to the above method. Not all the passengers know that the above methods of disembarking can save the total disembarkation time of all passengers, and many passengers will be inserted into the queue of passengers who disembark before him because they are in a hurry. In addition, the space in the cabin is small, and it is very difficult for the flight attendants to maintain order. Therefore, the delay time for passengers to cut the queue is a considerable value, which needs to be calculated in detail and included in the total time of disembarking.

There are two situations in which passengers can cut into the queue : at the middle of the queue and at the end of the queue. That is:

JQ (k)=JQT (k) (At this time,

$$T_k < \frac{len - ky}{speed}$$

, that is, the k^{th} passenger who cuts the queue is inserted into the queue)

Or JQ (k) = 0 (at this time $T_k > \frac{len-ky}{speed}$, that is, the k^{th} passenger who cuts the queue is inserted at the end of the queue)

If a passenger inserts at the end of the queue, he starts to put luggage at the same time as the passenger in the next queue, that is to say, he replaces the passenger in the same row as him in the next queue, which is equivalent to the two exchanged queues, so Do not delay any time (if the replaced passenger is re-inserted into the original queue, the time he has been delayed due to queue insertion will be recalculated), namely:

JQT(k)=0

If a passenger inserts into the queue, it is similar to the "tracking model", only the passenger who inserts at the end of the queue will block the passengers who normally disembark when they take their luggage. therefore:

$JQT_1(k)$= pre _bag_time (at this time $T_{k+1} - T_k >$pre_bag_time , that is, the k^{th} queue-cut passenger does not appear as the (k+ 1) -th queue-cut passenger during the process of taking his luggage)

Or $JQT_1(k)$=pre_bag_time+$T_k - T_{k+1}$ (at this time $T_{k+1} - T_k <$pre_bag_time , that

is, during the process of the passenger of the k row seat taking the baggage, the (k +1) th queue -cut of travelers appear)

So $T_{deplane}$ = pre_bag_time * $\frac{PersonNum}{RowNum}$ + $\frac{PersonNum}{V}$ + $\sum_{k=1}^{m} JQ(k)$

3 Model calculations

3.1 Comparison of boarding strategies

According to the above model, we use the computer to simulate 1,000 passenger seat situations for the four boarding strategies of "random boarding", "segment boarding", "boarding by seat" and "inverted pyramid boarding", (the "steffen boarding strategy" has only one seat situation) and calculates the time required for boarding for each situation, and the average of the time required for each boarding strategy. In the same way, we also simulated 1,000 cases of passengers cutting in the queue when getting off the plane, and calculated the average value. The results are as follows.

	minimum time	maximum time	Average time	Disembarkation time	Minimum total time on and off the machine	Maximum total time of getting on and off the machine	Average total time on and off the plane	error
random boarding	1 139	1 473	1 290.79	9 19.5	2 058.5	2 392.5	2 210.29	±20
Staged boarding	1 176	1 466	1 311.314		2 095.5	2 385.5	2 230.814	
Boarding by Seat	1 035	1 223	1 126.7		1 954.5	2 142.5	2 046.2	
Inverted Pyramid Boarding	9 76	1 164	1 066.511		1 895.5	2 083.5	1 986.011	
Steffen	8 32	8 32	8 32		1 751.5	1 751.5	1 751.5	

boarding								

According to the above table, we can conclude that the steffen boarding strategy takes the shortest time.

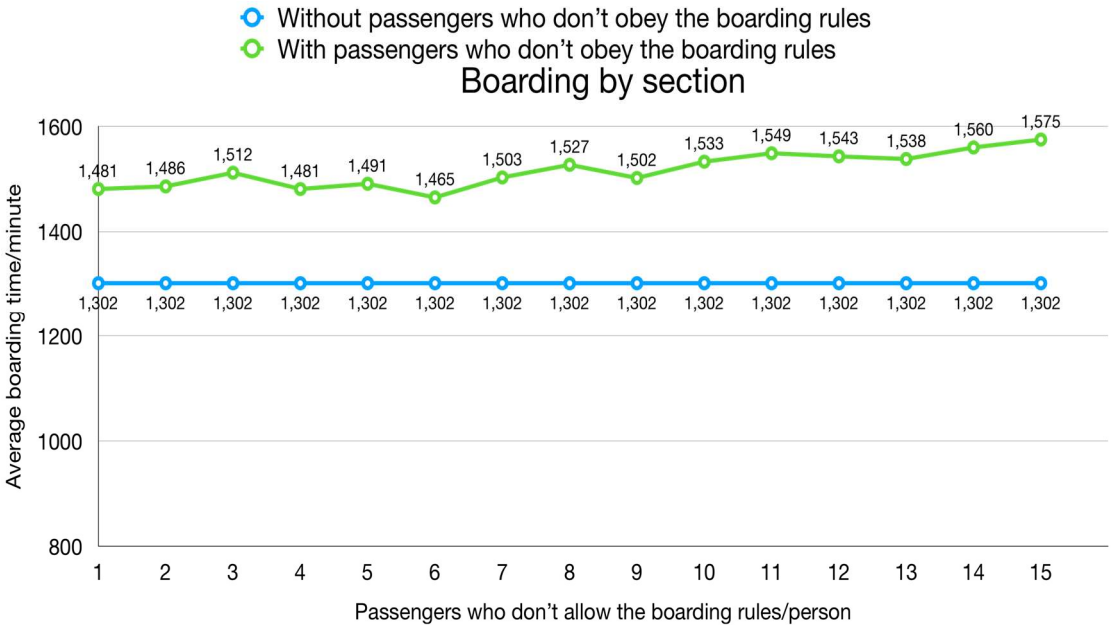
The Steffen boarding strategy is a staggered, inside-out boarding strategy. As shown in the picture below, the numbers on the seats represent the passengers' boarding sequence.

1	2	3	Aisle	6	5	4
19	20	21		24	23	22
7	8	9		12	11	10
25	26	27		30	29	28
13	14	15		18	17	16
31	32	33		36	35	34

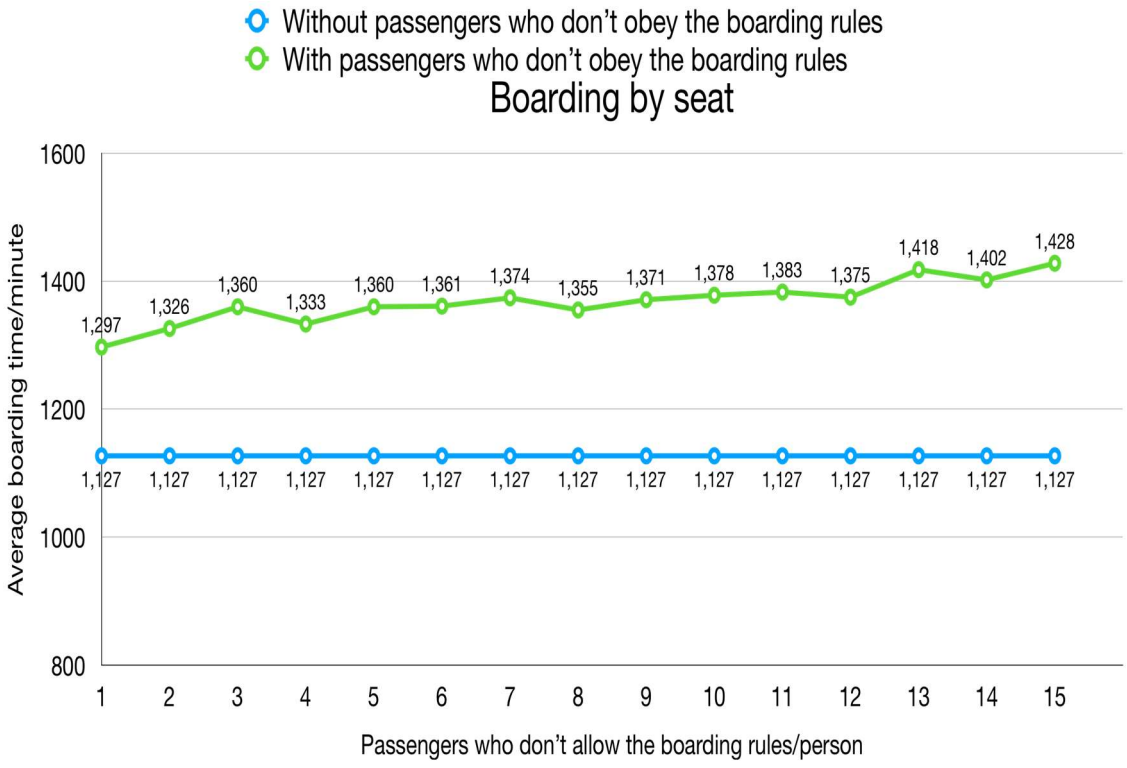
3.2 Sensitivity analysis

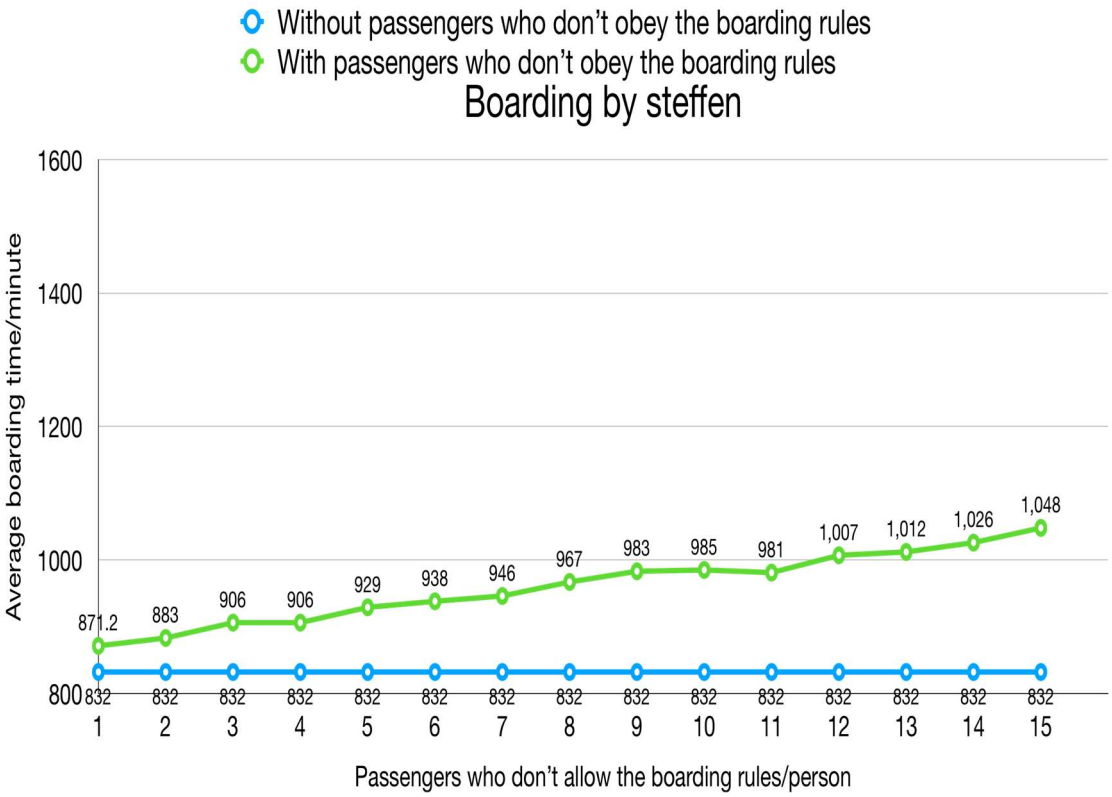
3.2.1 Consider passengers who do not follow the boarding method

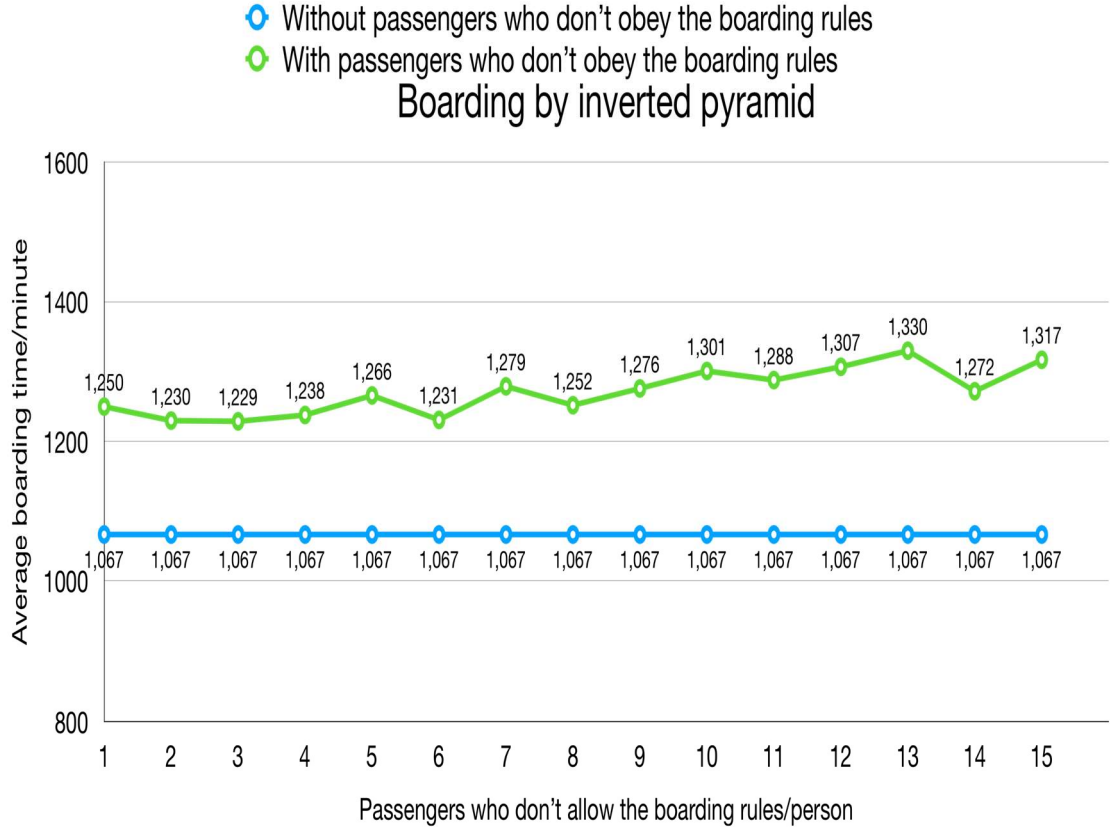
In the actual boarding process, not all passengers will board according to



regulations, so we need to consider the impact of passengers who do not follow the boarding method on the total boarding time. We simulated 1,000 times for each of the 4 boarding strategies other than "random boarding" with 1-20 passengers who did not follow the boarding method, and took the average value. The results are as follows.

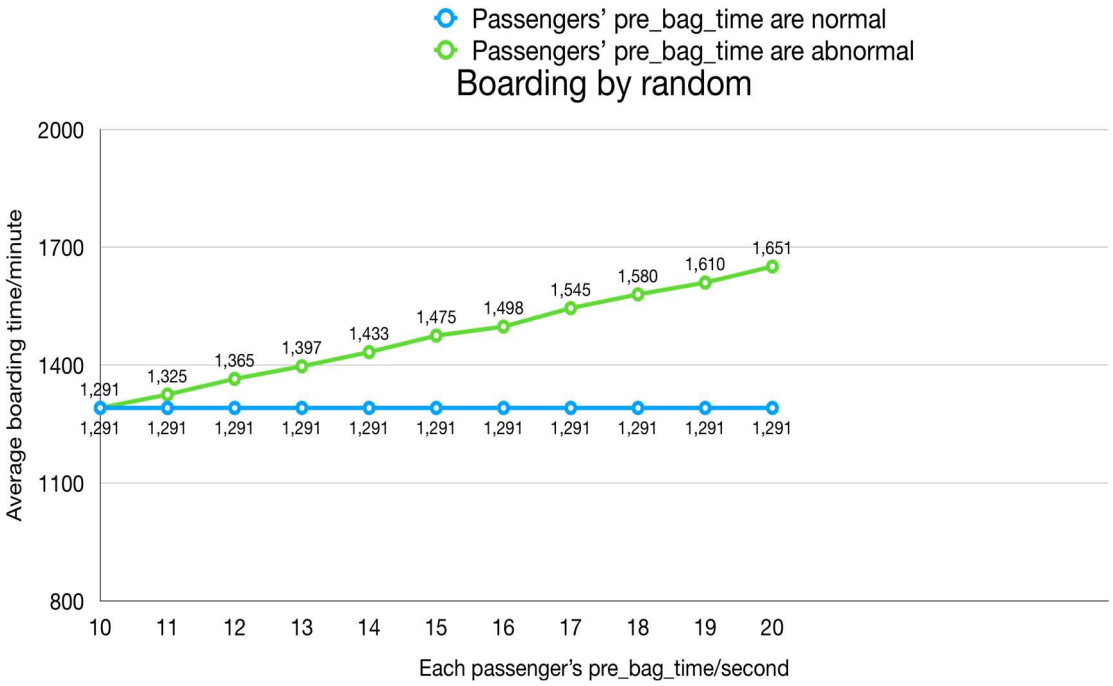


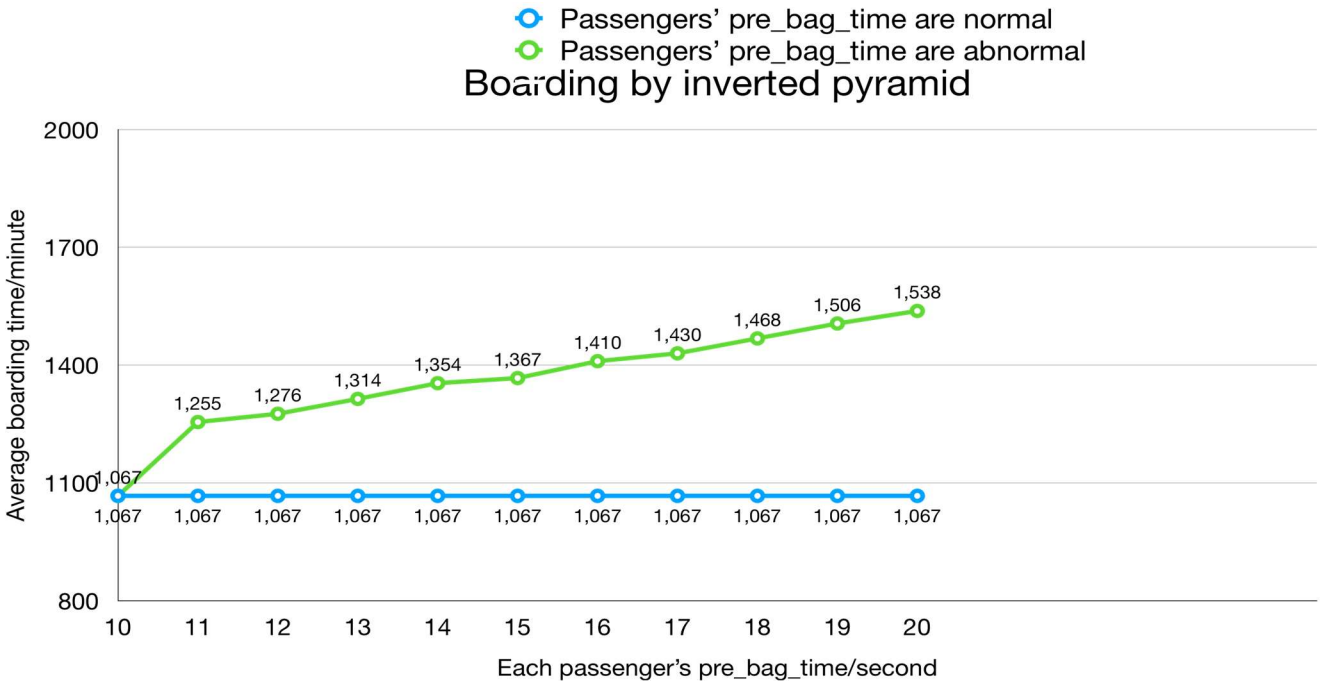

Without passengers who don't obey the boarding rules
With passengers who don't obey the boarding rules
Boarding by steffen
Average boarding time/minute
1600
1400
1200
1000
800
871.2
883
906
906
929
938
946
967
983
985
981
1,007
1,012
1,026
1,048
832
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Passengers who don't allow the boarding rules/person

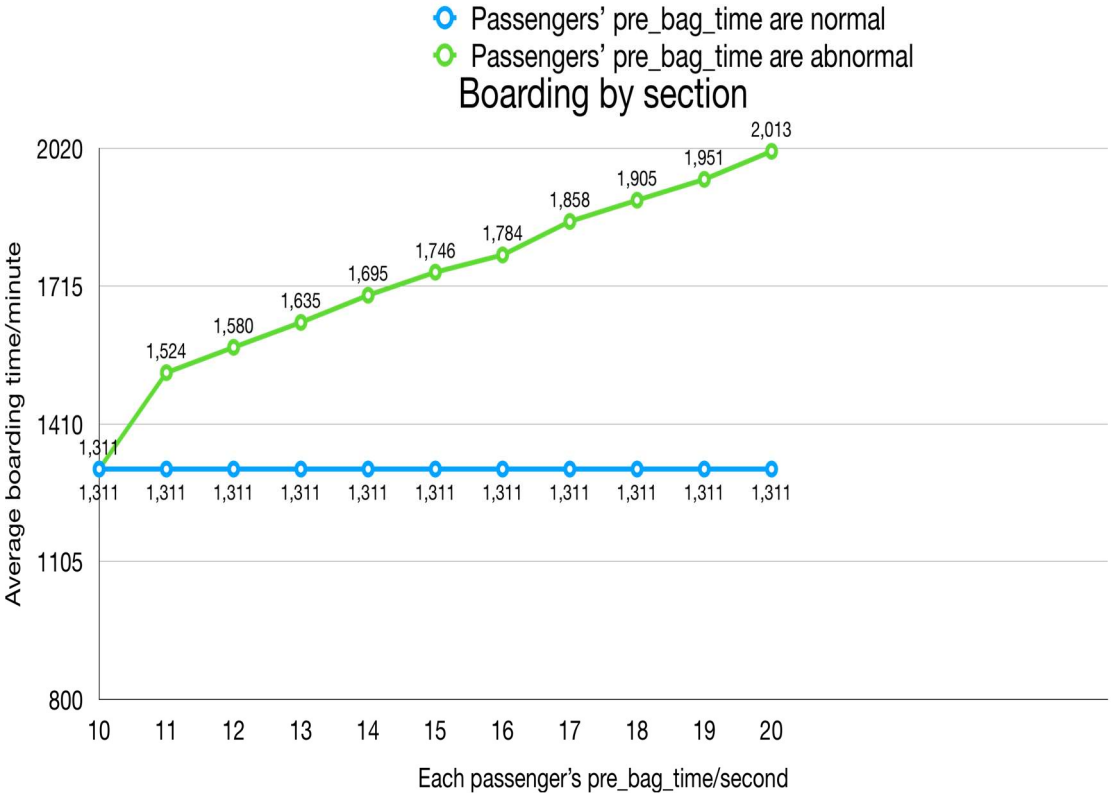

Without passengers who don't obey the boarding rules
With passengers who don't obey the boarding rules
Boarding by inverted pyramid
Average boarding time/minute
1600
1400
1200
1000
800
1,250
1,230
1,229
1,238
1,266
1,231
1,279
1,252
1,276
1,301
1,288
1,307
1,330
1,272
1,317
1,067
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Passengers who don't allow the boarding rules/person

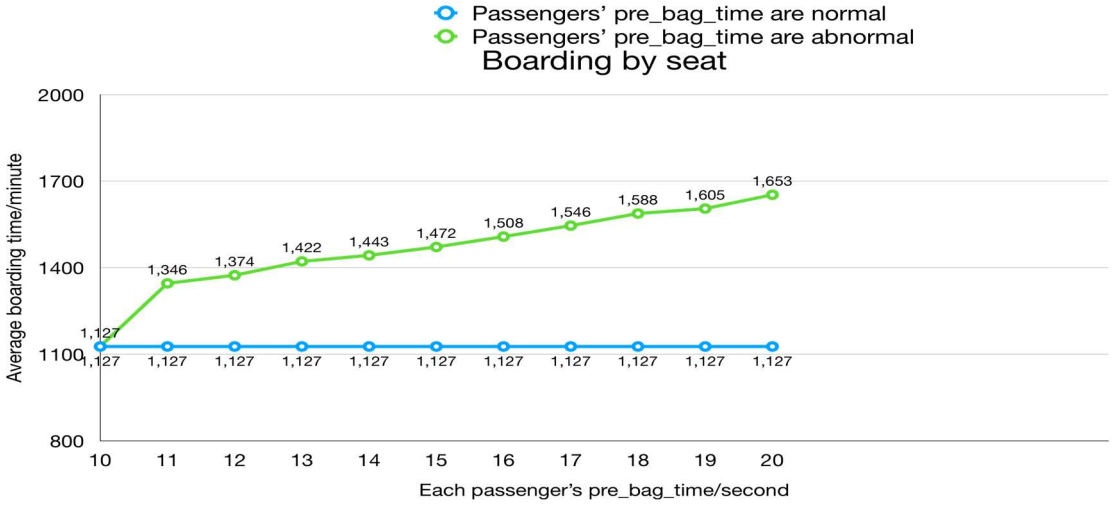
3.2.2 Considering the situation that passengers have more luggage

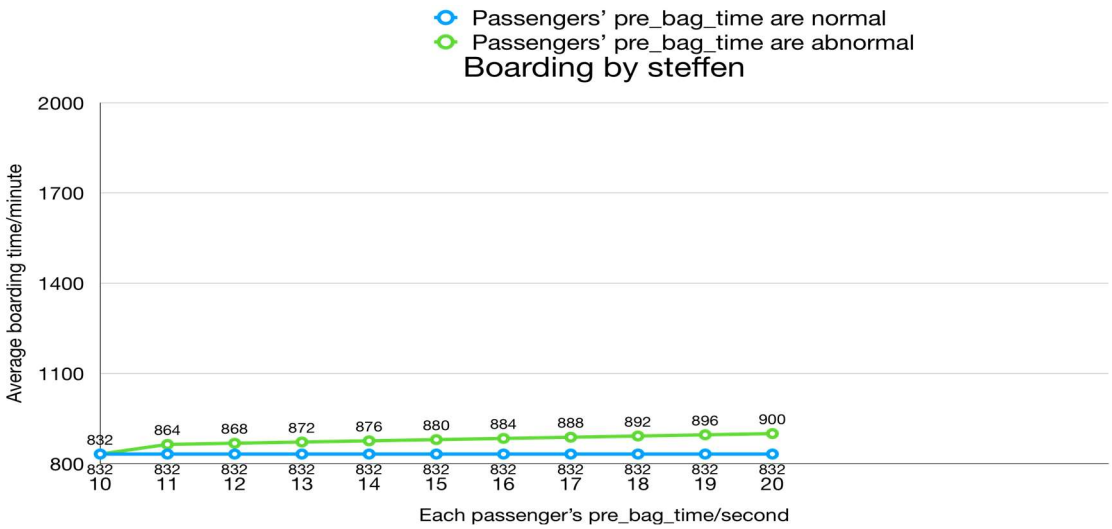
As above, we apply the model to the case of more passengers and luggage, and the results are as follows.






Passengers' pre_bag_time are normal
Passengers' pre_bag_time are abnormal
Boarding by section
Average boarding time/minute
2020
1715
1410
1105
800
1,311
1,524
1,580
1,635
1,695
1,746
1,784
1,858
1,905
1,951
2,013
10
11
12
13
14
15
16
17
18
19
20
Each passenger's pre_bag_time/second


Passengers' pre_bag_time are normal
Passengers' pre_bag_time are abnormal
Boarding by seat
Average boarding time/minute
2000
1700
1400
1100
800
1,127
1,346
1,374
1,422
1,443
1,472
1,508
1,546
1,588
1,605
1,653
10
11
12
13
14
15
16
17
18
19
20
Each passenger's pre_bag_time/second



3.2.3 The best boarding plan

According to the computer calculations in **3.1** , it is not difficult to conclude that among the five boarding strategies we discussed, the Steffen boarding strategy is the least time-consuming and least disturbed one. Therefore, we guess that the Steffen boarding strategy may be the best strategy for boarding.

To test this conjecture, we compared the Steffen boarding strategy with other strategies.

1. Boarding Time Taken by Steffen Boarding Strategy: Time Taken by Steffen Boarding Strategy = $\frac{198+Person_{last.row}}{V}$+ 6 ×OccTime

Explanation: Here, the boarding time of the last passenger is tracked and calculated using the method of the core model of **2.1**. The original formula is equivalent to the estimated time for the last passenger to reach the seat add the delayed time for luggage.

2. Other boarding strategies Boarding time:

To demonstrate that the Steffen boarding strategy is the optimal strategy, we apply scaling. Compare the time required for the Steffen boarding strategy with the minimum time required for other boarding strategies.

Other boarding strategies Minimum boarding time = $\frac{198+Person_{last.row}}{V}$+6 ×OccTime

Explanation: The last passenger's seat time is divided into the estimated time to

get to the seat + the delay time for luggage. Because other boarding strategies are divided into at least 1 98 ÷3 3=6 batches to enter the cabin, there will be at least six OccTimes for the delay time.

So other boarding strategies have minimum boarding time ≥ $\frac{198+Person_{last.row}}{V}$+6 ×OccTime

3. To sum up, other boarding strategies required boarding time ≥Steffen boarding strategies required boarding strategies.

3.So the Steffen boarding strategy is the best of all boarding strategies.

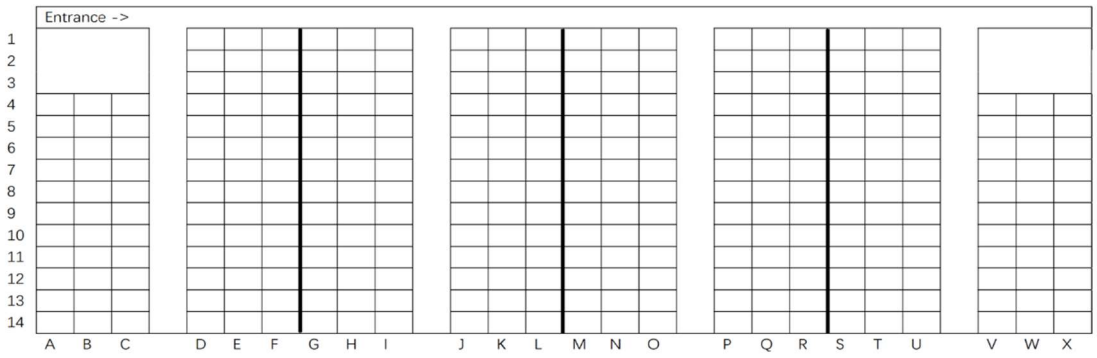
The best deplane strategy

Passengers' disembarkation time can be divided into two parts: normal walking time and the time passengers behind are blocked by passengers in front of them who are picking up

luggage. The total amount of time for all passengers to carry luggage is unchangeable, so the more time between passengers to carry luggage overlaps, the shorter time to get off the plane. Therefore, the best disembarkation strategy (for narrow-body aircrafts) is that passengers in each row by the aisle take luggage and disembark at the same time, and then passengers on the other side of the aisle take their luggage and disembark at the same time, and so on.

4. Adjust the boarding plan for different kinds of planes

4.1 Flying wing aircraft



The picture is a simplified version of the seat map of the flying wing airliner. In order to improve efficiency and facilitate calculation, we can divide it into 4 areas, as shown by the thick line in the figure. We call from left to right Zone I (A to F), Zone II (G to L), Zone III (M to R), and Zone IV (S to X).

Because the entrance is in Zone I , and the flow of the entrance is not limited, we can ignore the boarding time of Zone I , because it can allow passengers to complete boarding in a very short time. And the four districts Ⅰ , Ⅱ , Ⅲ and Ⅳ are connected by narrow corridors, so only one person can pass at a time. This limits the number of people who can go to II , III , and IV per unit time .

In 2.1 , we calculated that the Steffen boarding strategy is the most time-saving, so our detailed boarding strategies will use Steffen, which will not be explained in this section. The steffen boarding strategy we use will also be improved due to the special type of Wings.

Our boarding strategy is as follows:

The rows of seats D, E, F, G, H, and I form a seat block with 14 rows and 6 columns, plus the aisle on the right side , there are 14 rows and 7 columns in total. J, K, L, M, N, O, and P, Q, R, S, T, and U, including the right aisle, are also in 14 rows and 7 columns.

Because it's 7-column, we can have a group of 7 passengers stand on top of the three seating blocks and simultaneously walk into the aisle to their right to board the plane according to the Steffen strategy. This action takes 7 blocks of time.

When the last passenger above Zone II entered the right aisle, the second group of passengers following behind had already reached the top of the aisle, only 14 blocks away from the target position. (To minimize the time between two groups of passengers, the second group must be immediately behind the first group.)

The time for the second group of passengers to walk 14 blocks is enough for the first

group of passengers to put their bags and sit down, so after the second group reaches the "initial position" of the first group, the first group has already sat down and the aisle has been cleared.

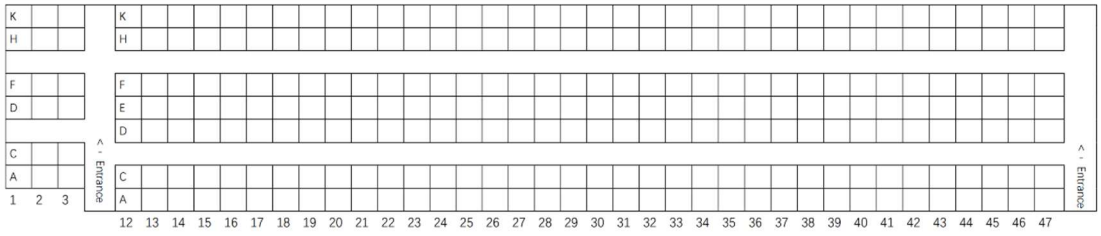
At this time, a group of boarding is completed, which takes a total of 21 grids of time.

II and III have 84 seats respectively, and each such boarding can seat 7 passengers in each area, that is, 12 similar boardings are required to be fully seated. And Zone IV has only 75 seats and only 11 similar boardings.

In this way, it should take a maximum of 252 blocks of time. At a rate of 4 seconds/grid, it takes roughly 1000 seconds, or about 17 minutes.

In this way, boarding can make the time for luggage storage more abundant, the operating space when luggage is more spacious, and the aisle is less crowded, which improves the passenger's comfort while increasing the speed.

4.2 Twin-aisle twin-entrance airplane



The picture is a simplified seat diagram of a "twin-aisle twin-entrance " aircraft.

This plane has first class, but first class has priority for boarding, so you can temporarily ignore first class and only focus on the economy class part.

To minimize the boarding time, we need to use the fastest boarding method combined. In Section 2.1 , we found out that the Steffen boarding method is the fastest and most stable boarding method, so we want to use the Steffen boarding method as much as possible.

Economy class has 7 rows (pictured), but Steffen only works in the case of an even row (symmetrical along the aisle),so we can divide the whole boarding process into two sets of Steffen ("AC- DE " and " FH ") and a separate K column.

The two groups of Steffen can make the time as short as possible. In order to make the K column fill up as soon as possible, we can adjust the boarding order by putting K in front of the two groups of Steffen, and let the two boarding gates fill K together first. The

reasons are as follows.

(1) Put K in front of the two groups of Steffen, so that the boarding of column K will not delay the boarding of "AC-DE". Because the boarding time of the "AC-DE" group is significantly longer than that of the "FH" group, it can avoid continuing to widen the gap between the two groups, thus wasting resources.

(2) It is very necessary to fill the K column with two boarding gates, which can reduce the time by about half compared to one boarding gate.

(3) Because there are 4 rows of passengers to board the plane from one aisle, and the other 3 rows to board the plane from the other, so to make the total distance for the passengers to be as short as possible, let the 4 rows of passengers leave the boarding gates from the closer channel, let the remaining 3 rows go to the farther channel. Here, those 4 rows of passengers are obviously the Steffen boarding group with 4 rows - "AC-DE" or "K H-FE ", and "AC-DE" uses the passageway closer to the entrance, so there is no "KH- FE", "DC" and A method.

5. Conclusion

In recent years, more and more people travel by air. As the number of passengers boarding at the airport increases, the boarding gate is congested and disorderly, which results in increased operating costs and reduced passenger satisfaction. If a suitable boarding strategy can be found and the boarding methods of passengers can be reasonably planned, the boarding pressure can be relieved to a certain extent. During the boarding process of passengers from the cabin door to the cabin seat, many uncertain interference factors will greatly prolong the boarding time. Especially in some busy airports, it is particularly important to reduce the waiting time of the aircraft and raise the boarding efficiency.

To help airlines find boarding strategies more efficiently and accurately, we built an original "tracking model" to calculate the time required for each boarding strategy. It can precisely calculate the boarding time required for each

boarding strategy by tracking the last passenger seated. Since the boarding time starts when the first passenger enters the cabin and ends when the last passenger takes a seat, this time can be divided into the walking time and blocking time of the last passenger, so by tracking the last passenger seated The method of calculating the total boarding time is accurate and feasible. This approach also greatly reduces the computing resources required to calculate boarding time, which can fundamentally help airlines reduce boarding time.

We also performed practical calculations on several boarding strategies using the "tracking model" to verify the accuracy of the model. It turns out that all the six boarding strategies, "random boarding", "segment boarding", "boarding by seat", "inverted pyramid boarding" and "steffen boarding strategy" Can accurately calculate the boarding time.

The process of getting off the plane is equally important. The increased efficiency of disembarkation also helps reduce flight turnaround times, boosting airline revenue, and saving passengers' time. For the process of getting off the plane, we followed some of the ideas of the "tracking model" and established a "queue-cutting model", adding the time delayed due to passengers' queue-cutting to the time for passengers to get off the plane in an orderly manner to obtain the total alighting time . .

To verify the generality of the model, we also applied the model to several other models. In the calculation of the boarding and disembarking time of "flying-wing aircraft" and "dual-entry dual-aisle aircraft", our model can still calculate the results accurately.

In response to some of the more extreme factors, such as the occupancy of seats that are restricted by the epidemic, we performed a sensitivity analysis on our model to test the stability of the model. We calculating the boarding time when the attendance is 30% , 50 % , and 70 % of the occupancy rate , our model still performed normally .

Various tests show that our model can help airlines calculate the boarding time required for various boarding strategies with strong accuracy, generality and stability. It can actually save boarding time and improve boarding efficiency.

Expectations

Firstly, our model can adapt very well in some special situations, but when there are some extreme situations, our model cannot contain very well. For example, when there are too many passengers who don't obey the boarding rules, steffen boarding strategy may lose its efficacy and boarding by random may become the fastest boarding strategy.

Secondly, when calculating the boarding and deplaning time, our model can't ensure its stability.

Thirdly, when calculating the boarding time of flying wing aircrafts, because of their complexity, we temporarily cannot calculate the time which is delayed because that passengers who seat first let the others go to their seats.

If we have the chance to take part in IMMC again in the future, we will strengthen the accuracy and universality of our model. For some complicated situations, we will find some easier solutions and make our answers more specific and comprehensive.

Letter to the manager

Dear airline executives,

Nowadays, more and more people prefer to travel by air. Therefore, the usage rate of aircrafts and departure gates are becoming higher and higher. To improve the efficiency of air transport, accelerating passengers' boarding and deplaning is very necessary.

For boarding, there are a few strategies, such as "random boarding", "segment boarding", "seat boarding", "inverted pyramid boarding" and "Steffen boarding strategy". We calculated the average boarding time of each strategy by computer simulation, and found that Steffen boarding strategy is the fastest one, that is, letting the slowest passengers board first can save most time. However, it is very hard to implement the strategy because many passengers will not obey the rule. But we still recommend you to try this boarding strategy. It is much faster than the other boarding strategies.

For deplaning, the fastest strategy is obvious: let the passengers beside the aisle deplane first, and the passengers beside the window deplane last. The only problem of deplaning is that some passengers jump the queue, so we hope you can find some ways to keep the cabin in order.

Our boarding and deplaning strategies are compatible in all the aircrafts, including narrow-body aircrafts, flying wing aircrafts and twin-aisle aircrafts. In some special situations, for example, when the occupancy rate is limited and when passengers have more luggage, our strategies still work.

Last but not least, we strongly recommend you to try our strategies. Time is money, and using our strategies can help you save much more time, and money as well.

Best wishes,

The IMMC team

Addendums

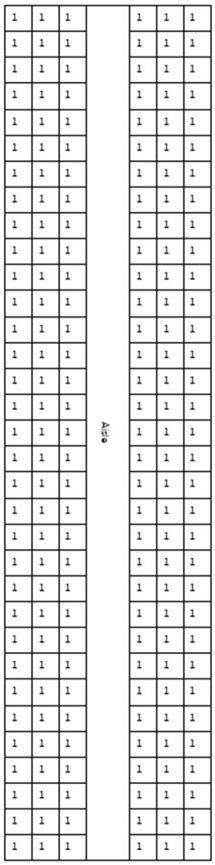
6.Variables

symbol	Using place	definition	remark
x	all	The group of people	
A（X)	2.1.4, 2.1.5	Give the group of people an order	
S（X)	2.1.4		
K（X)	2.1.4, 2.1.6	The last one to put the baggages	
I（X)	2.1.4	The time that the first person in set x starts to unload the luggage until the second person starts to unload the luggage	
M（X)	2.1.4, 2.1.6	The time when the first person in set x is traced	
$Person_{last}$	2.1.1, 2.1.2, 2.1.3, 2.1.6	The passenger who gets to seat latest.	
$PRE\_BAG\_TIME_{total}$	2.1.5	Total estimated time for all persons to place or collect their luggage	
OCC_TIME()	2.1.4, 2.1.5	The amount of time passengers occupy the aisle.	
Q_k	2.1.5	Number of rows of seats in k	
P_K	2.1.5	Number of seat columns of k	
T_0	2.1.4, 2.1.5	Total boarding time	
T_{total}	2.1.6	$Person_{last}$'s total amount of time wasted	
T_{rang}(I)	2.1.5	The time to let others get in	
T_k	2.2	K the amount of time the person cuts in line	
JQ（k)	2.2	The time k wasted in jumping	

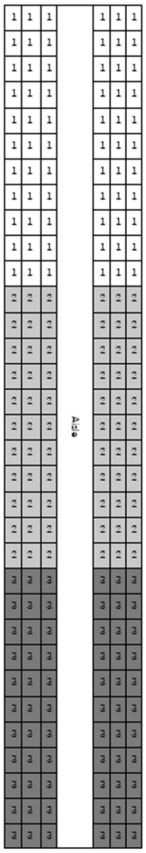
		a queue	
JQT（k)	2.2	The time it took k to jump a queue	
JQT	2.2	The time it took k to cut from the back of the line	
len	2.2	Length of aircraft	
$\mathbf{T}_{\mathbf{deplane}}$	2.2	The total time to disemarking	
V	all	The speed of speed	The unit is grid/second
grid	all	One grid is equal to space between the front and back stools	

Addendum 1 :

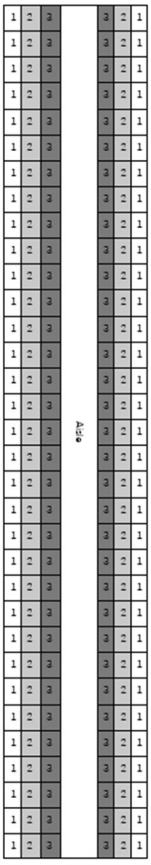
Five boarding modes mentioned in the paper.



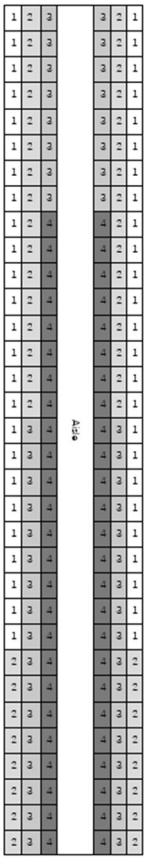
Random



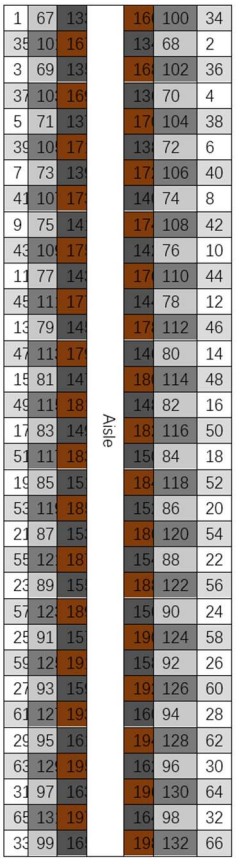
Boarding by Section



Boarding by Seat



Boarding by InvitedPyramid



Boarding by Steffen

Code :

Main.py

```
import random
import board
import deplane

# mode:
M freedom = 1
M section = 2
M seat = 3
M steffen = 4
M inv = 5


def cmp(l1):
    return l1[1]


def p(li):
    rt = []
    for x in li:
        rt.append(x[0])
    return rt


class Board:
    def   init  (self, pre bag time, pre speed, give seat time,
row=33, col=6):
        board.pre bag time = pre bag time
        board.pre speed = pre speed
        board.give seat time = give seat time
        self.lst = [x + str(i) for i in range(1, row + 1) for x in
                    [chr(number) for number in range(ord('A'),
ord('A') + col)]]

    def freedom(self, sumof=198):
        lst = self.lst.copy()
        for   in range(sumof):
            i = random.randint(0, sumof - 1)
            j = random.randint(0, sumof - 1)
            lst[i], lst[j] = lst[j], lst[i]

        return lst

    def section(self):
        lst = self.lst.copy()
        for   in range(133, 197):
            i = random.randint(133, 197)
            j = random.randint(133, 197)
            lst[i], lst[j] = lst[j], lst[i]
        for   in range(67, 133):
            i = random.randint(67, 132)
            j = random.randint(67, 132)
            lst[i], lst[j] = lst[j], lst[i]
        for   in range(0, 66):
            i = random.randint(0, 65)
            j = random.randint(0, 65)
            lst[i], lst[j] = lst[j], lst[i]

        return lst

    def seat(self):
        lst = self.lst.copy()
        for   in range(0, 197, 6):
            i = random.randrange(0, 197, 6)
            j = random.randrange(0, 197, 6)
```

```
            lst[i], lst[j] = lst[j], lst[i]
        for   in range(5, 197, 6):
            i = random.randrange(5, 197, 6)
            j = random.randrange(5, 197, 6)
            lst[i], lst[j] = lst[j], lst[i]

        for   in range(1, 197, 6):
            i = random.randrange(1, 197, 6)
            j = random.randrange(1, 197, 6)
            lst[i], lst[j] = lst[j], lst[i]
        for   in range(4, 197, 6):
            i = random.randrange(4, 197, 6)
            j = random.randrange(4, 197, 6)
            lst[i], lst[j] = lst[j], lst[i]

        for   in range(2, 197, 6):
            i = random.randrange(2, 197, 6)
            j = random.randrange(2, 197, 6)
            lst[i], lst[j] = lst[j], lst[i]
        for   in range(3, 197, 6):
            i = random.randrange(3, 197, 6)
            j = random.randrange(3, 197, 6)
            lst[i], lst[j] = lst[j], lst[i]

        li = []
        for i in range(0, 197, 6):
            li.append(lst[i])
        for i in range(5, 197, 6):
            li.append(lst[i])
        for i in range(1, 197, 6):
            li.append(lst[i])
        for i in range(4, 197, 6):
            li.append(lst[i])
        for i in range(2, 197, 6):
            li.append(lst[i])
        for i in range(3, 197, 6):
            li.append(lst[i])

        return li

    def steffen(self):
        lst = ["A33", "F32", "A31", "F30", "A29", "F28", "A27",
"F26", "A25",           "F24", "A23", "F22", "A21", "F20", "A19",
"F18", "A17", "F16",  "A15", "F14", "A13", "F12", "A11", "F10", "A9",
"F8", "A7", "F6",   "A5", "F4", "A3", "F2", "A1", "F33", "A32",
"F31", "A30", "F29",  "A28", "F27", "A26", "F25", "A24", "F23",
"A22", "F21", "A20",  "F19", "A18", "F17", "A16", "F15", "A14",
"F13", "A12", "F11", "A10", "F9", "A8", "F7", "A6", "F5", "A4", "F3",
"A2", "F1", "B33", "E32", "B31", "E30", "B29", "E28", "B27", "E26",
"B25", "E24", "B23", "E22", "B21", "E20", "B19", "E18", "B17", "E16",
"B15", "E14", "B13", "E12", "B11", "E10", "B9", "E8", "B7", "E6",
"B5", "E4", "B3", "E2", "B1", "E33", "B32", "E31", "B30", "E29",
"B28", "E27", "B26", "E25", "B24", "E23", "B22", "E21", "B20", "E19",
"B18", "E17", "B16", "E15", "B14", "E13", "B12", "E11", "B10", "E9",
"B8", "E7", "B6", "E5", "B4", "E3", "B2", "E1", "C33", "D32", "C31",
"D30", "C29", "D28", "C27", "D26", "C25", "D24", "C23", "D22", "C21",
"D20", "C19", "D18", "C17", "D16", "C15", "D14", "C13", "D12", "C11",
"D10", "C9", "D8", "C7", "D6", "C5", "D4", "C3", "D2", "C1", "D33",
"C32", "D31", "C30", "D29", "C28", "D27", "C26", "D25", "C24", "D23",
"C22", "D21", "C20", "D19", "C18", "D17", "C16", "D15", "C14", "D13",
"C12", "D11", "C10", "D9", "C8", "D7", "C6", "D5", "C4", "D3", "C2",
"D1"]
        return lst

    def inv(self):
        lst = self.lst
        ls = []

        # A9-A33
```

```
for   in range(55, 194, 6):
    i = random.randrange(55, 194, 6)
    j = random.randrange(55, 194, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(55, 194, 6):
    ls.append(lst[i])

# F9-F33
for   in range(59, 198, 6):
    i = random.randrange(60, 198, 6)
    j = random.randrange(60, 198, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(60, 198, 6):
    ls.append(lst[i])

# A1-A8
for   in range(0, 49, 6):
    i = random.randrange(0, 49, 6)
    j = random.randrange(0, 49, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(1, 49, 6):
    ls.append(lst[i])

# F1-F8
for   in range(6, 55, 6):
    i = random.randrange(6, 55, 6)
    j = random.randrange(6, 55, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(6, 55, 6):
    ls.append(lst[i])

# B18-B33
for   in range(103, 193, 6):
    i = random.randrange(103, 193, 6)
    j = random.randrange(103, 193, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(103, 193, 6):
    ls.append(lst[i])

# E18-E33
for   in range(106, 196, 6):
    i = random.randrange(106, 196, 6)
    j = random.randrange(106, 196, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(106, 196, 6):
    ls.append(lst[i])

# B1-B17
for   in range(1, 97, 6):
    i = random.randrange(1, 97, 6)
    j = random.randrange(1, 97, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(1, 97, 6):
    ls.append(lst[i])

# E1-E17
for   in range(3, 100, 6):
    i = random.randrange(3, 100, 6)
    j = random.randrange(3, 100, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(3, 100, 6):
    ls.append(lst[i])

# C26-C33
for   in range(152, 194, 6):
    i = random.randrange(152, 194, 6)
    j = random.randrange(152, 194, 6)
    lst[i], lst[j] = lst[j], lst[i]
for i in range(152, 194, 6):
```

```
            ls.append(lst[i])

        # D26-D33
        for   in range(153, 195, 6):
            i = random.randrange(153, 195, 6)
            j = random.randrange(153, 195, 6)
            lst[i], lst[j] = lst[j], lst[i]
        for i in range(153, 195, 6):
            ls.append(lst[i])

        # C1-C25
        for   in range(2, 146, 6):
            i = random.randrange(2, 146, 6)
            j = random.randrange(2, 146, 6)
            lst[i], lst[j] = lst[j], lst[i]
        for i in range(2, 146, 6):
            ls.append(lst[i])

        # D1-D25
        for   in range(3, 147, 6):
            i = random.randrange(3, 147, 6)
            j = random.randrange(3, 147, 6)
            lst[i], lst[j] = lst[j], lst[i]
        for i in range(3, 147, 6):
            ls.append(lst[i])

        return ls

    def NA(self, lst, num, sumof=198):
        for   in range(num):
            i = random.randint(0, sumof - 1)
            j = random.randint(0, sumof - 1)
            lst.insert(i, lst[j])
        return lst

    def main(self, mode, na=0, sumof=None):
        if mode == M freedom:
            if sumof is not None:
                ls = self.NA(self.freedom(sumof=sumof), na)
            else:
                ls = self.NA(self.freedom(), na)
        elif mode == M section:
            ls = self.NA(self.section(), na)
        elif mode == M seat:
            ls = self.NA(self.seat(), na)
        elif mode == M steffen:
            ls = self.NA(self.steffen(), na)
        elif mode == M inv:
            ls = self.NA(self.inv(), na)
        return board.main(lst=ls), ls


class Deplane:
    def   init  (self, pre bag time, V, plane length, plane row):
        deplane.pre bag time = pre bag time
        deplane.pre speed = V  # cell/s
        deplane.plane length = plane length
        deplane.plane row = plane row
        self.lst = [x + str(i) for i in range(1, 34) for x in ['A',
'B', 'C', 'D', 'E', 'F']]

    def main(self, person num, Tk2 Max):
        li = []

        for   in range(person num):
            i = random.randint(0, 197)
            tk = random.randint(1, Tk2 Max)
            if [self.lst[i]] in p(li):
                person_num += 1
```

```
                continue
            li.append([self.lst[i], tk])
        li.sort(key=cmp)
        for i in range(len(li)):
            if i + 1 < len(li):
                li[i].append(li[i + 1][1])
        ti = deplane.main(li)
        return ti, li


"""
boarding:
"""
# ====<random>====
result_bo_fre = []
for i in range(1000):
    bd = Board(10, 4, 7)
    t = bd.main(M_freedom)
    print(i, t)
    result_bo_fre.append(t[0])
print(result_bo_fre)
print(sum(result_bo_fre) / 1000)

# ====<Boarding by Section>====
result_bo_sec = []
for i in range(1, 11):
    bd = Board(10 + i, 4, 7)
    rst = 0
    for j in range(100):
        t = bd.main(M_section)
        rst += t[0]
    print(i, rst / 100)
    result_bo_sec.append(rst / 100)
print(result_bo_sec)
print(sum(result_bo_sec) / 1000)

# ====<Boarding by Seat>====
result_bo_sea = []
for i in range(1, 11):
    bd = Board(10 + i, 4, 7)
    rs = 0
    for j in range(100):
        t = bd.main(M_seat)
        rs += t[0]
    print(i, rs / 100)
    result_bo_sea.append(rs / 100)
print(result_bo_sea)
print(sum(result_bo_sea) / 1000)

# ====<Boarding by Steffen>====
result_bo_steffen = []
for i in range(1, 11):
    bd = Board(10 + i, 4, 7)
    rt = 0
    rt += bd.main(M_steffen)[0]
    print(rt)
    result_bo_steffen.append(rt)
print(result_bo_steffen)

# ====<Boarding by InvertedPyramid>====
result_bo_inv = []
for i in range(1, 11):
    bd = Board(10+i, 4, 7)
    st = 0
    for j in range(100):
        try:
            t = bd.main(M_inv)
        except:
            try:
```

```
                t = bd.main(M inv)
            except:
                try:
                    t = bd.main(M inv)
                except:
                    t = bd.main(M inv)
        st += t[0]
    print(i, st/100)
    result bo inv.append(st/100)
print(result bo inv)
print(sum(result bo inv) / 1000)

"""
deplane
"""
result de = []
num = 1000
dp = Deplane(10, 4, 33, 6)
for i in range(num):
    try:
        t = dp.main(30, 100)
        print(i, t)
        result de.append(t[0])
    except:
        pass
print(result de)
print(sum(result de) / len(result de))


"""
Boarding (Q4)
"""

# <Random>
bd = Board(10, 4, 7, col=4)
print(bd.lst)
res = []
for i in range(1000):
    t = bd.main(M freedom, sumof=198*2//3)
    res.append(t[0])
    print(i+1, t)
print(res)
```

board.py

```
def rm(LST, item):
    LI = LST.copy()
    LI.remove(item)
    return LI


def get(number):
    return number[0], int(number[1:])


def OccTime(Set, p, letter='C'):
    for x in Set:
        P = get(p)
        X = get(x)
        if P[1] == X[1] \
                and ((ord(P[0]) < ord(letter) and ord(X[0]) <
ord(letter))
                     or (ord(P[0]) >= ord(letter) and ord(X[0]) >=
ord(letter))):
            # In the same row.
            if ord(P[0]) >= ord(letter) and ord(X[0]) >= ord(letter):
# P,X is all in DEF
                if ord(P[0]) > ord(X[0]):  # need to give_seat.
                    return (ord(P[0]) - ord(X[0])) * give_seat_time +
pre_bag_time
            if ord(P[0]) < ord(letter) and ord(X[0]) < ord(letter):
# P,X is all in ABC
                if ord(P[0]) < ord(X[0]):  # need to give_seat
                    return (ord(X[0]) - ord(P[0])) * give_seat_time +
pre_bag_time

    return pre_bag_time


def nowrow(number, Set):
    ing = S(Set)
    return get(ing)[1] - Set.index(number) + Set.index(ing)


def A(y, Set):
    return get(y)[1] + Set.index(y) + 1
```

```
def S(Set):
    MINV = 100000
    MIN = ''
    for x in Set:
        if A(x, Set) < MINV:
            MIN = x
            MINV = A(x, Set)
    return MIN


def I(Set):
    if get(S(rm(Set, S(Set))))[1] - nowrow(S(rm(Set, (S(Set)))), Set)
== 1:
        return get(S(rm(Set, S(Set))))[1] - get(S(Set))[1]
    else:
        return A(S(rm(Set, (S(Set)))), Set) - A(S(Set), Set) +
OccTime(Set, S(Set))


def M(Set):
    if OccTime(Set, S(Set)) - I(Set) >= 0:
        return OccTime(Set, S(Set)) - I(Set)
    else:
        return OccTime(Set, S(Set))


"""
MAIN:
"""


def main(lst):
    length = len(lst)
    SUM = 0
    for _ in lst:  # mEi=1
        SUM += M(lst)
        for x in lst[:lst.index(S(lst)) + 1]:
            lst.remove(x)

    normal = length * pre_speed + OccTime(lst, S(lst))
    return SUM + normal
```

deplane.py

```
def get(number):
    return number[0], int(number[1:])


def JQT(k, pos, Tk, Tk2):
    if Tk2 - Tk < pre_bag_time:
        return pre_bag_time - (Tk2 - Tk)
    else:
        return pre_bag_time


def JQ(k, pos, Tk, ky, Tk2=10000):
    if Tk < (plane_length - ky) / pre_speed:
        return JQT(k, pos, Tk, Tk2)
    elif Tk > (plane_length - ky) / pre_speed:
        return 0


def main(lst):  # [pos, Tk, Tk2]
    SUM = 0
    for i in range(len(lst)):
        k = i + 1
        pos = lst[i][0]
        Tk = lst[i][1]
        if len(lst[i]) == 2:
            Tk2 = Tk + 100
        else:
            Tk2 = lst[i][2]
        ky = get(pos)[1]  # 'F13', got 13
        SUM += JQ(k, pos, Tk, ky, Tk2)
    return SUM + (plane_row * pre_bag_time + plane_row * plane_length
* pre_speed) + plane_row * pre_bag_time


# print(main(li))
```

FlyWings.py

```
def get(number):
    return number[0], int(number[1:])


class FlyWing:
    def __init__(self, pbt):
        self.lst = []
        self.Time = 0
        self.pl = ""
        self.pn = 0
        self.pre_bag_time = pbt

    def add(self, item):
        self.lst = [item] + self.lst

    def Twalk(self, val):
        xy = get(val)
        if xy[0] in "ABCDEF":
            x = 0
        elif xy[0] in "GHIJKL":
            x = 1
        elif xy[0] in "MNOPQR":
            x = 2
        elif xy[0] in "STUVWX":
            x = 3
        return xy[1] + x * 6 + 1

    def check(self):
        for i in range(len(self.lst)):
            if self.Twalk(self.lst[i]) - i - 1 <= 0:
                if self.pl == self.lst[i]:
                    self.pn += 1
                else:
                    self.pl = self.lst[i]
                    self.pn = 0
                if self.pn >= self.pre_bag_time:
                    self.lst = self.lst[:i]
                    return False
                self.Time += pre_speed
                self.lst = self.lst[:i + 1]
                return True
```

```
            return False


def main(lst, pbt=10):
    fw = FlyWing(pbt)
    for i in range(len(lst)):
        fw.add(lst[i])
        fw.check()
    for _ in range(10):
        fw.check()
    return fw.Time + len(lst) + fw.Twalk(lst[-1])


std = [x + str(i) for i in range(4, 15) for x in "ABC"] + [x + str(i)
for i in range(1, 15) for x in "DEFGHIJKLMNOPQRSTU"] + [x + str(i)
for i in range(4, 15) for x in "VWX"]
pre_speed = 4
print(main(std))
```

Data.py

```
print("Random Boarding:")
lst = [1409, 1207, 1336, 1281, 1280, 1307, 1256, 1286, 1301, 1200,
1257, 1232, 1296, 1269, 1279, 1353, 1336, 1226, 1279, 1301, 1337,
1340, 1273, 1198, 1305, 1189, 1252, 1276, 1334, 1329, 1220, 1298,
1365, 1258, 1280, 1330, 1310, 1355, 1415, 1259, 1231, 1322, 1272,
1306, 1296, 1228, 1326, 1251, 1276, 1349, 1362, 1314, 1217, 1237,
1301, 1196, 1329, 1336, 1348, 1230, 1239, 1233, 1243, 1252, 1290,
1240, 1328, 1240, 1353, 1281, 1283, 1315, 1333, 1334, 1243, 1344,
1344, 1256, 1259, 1287, 1265, 1280, 1408, 1306, 1301, 1283, 1229,
1211, 1322, 1389, 1256, 1347, 1239, 1276, 1315, 1239, 1357, 1319,
1235, 1320, 1204, 1332, 1259, 1341, 1211, 1218, 1324, 1295, 1278,
1277, 1239, 1255, 1321, 1340, 1364, 1323, 1267, 1223, 1246, 1208,
1337, 1286, 1255, 1265, 1271, 1333, 1237, 1294, 1282, 1333, 1340,
1254, 1255, 1244, 1356, 1331, 1303, 1266, 1364, 1278, 1169, 1241,
1278, 1292, 1356, 1325, 1350, 1233, 1254, 1231, 1269, 1328, 1337,
1255, 1251, 1287, 1283, 1315, 1300, 1278, 1380, 1282, 1458, 1255,
1294, 1253, 1298, 1315, 1332, 1268, 1301, 1201, 1234, 1224, 1306,
1294, 1253, 1226, 1271, 1307, 1307, 1289, 1275, 1257, 1344, 1266,
1300, 1265, 1322, 1225, 1342, 1314, 1265, 1283, 1269, 1286, 1249,
1256, 1351, 1357, 1303, 1214, 1238, 1289, 1306, 1281, 1243, 1283,
1210, 1347, 1188, 1303, 1302, 1298, 1273, 1281, 1380, 1260, 1205,
1259, 1346, 1265, 1343, 1244, 1410, 1281, 1310, 1240, 1200, 1276,
1398, 1263, 1305, 1303, 1334, 1342, 1222, 1341, 1279, 1366, 1338,
1261, 1422, 1248, 1308, 1153, 1272, 1238, 1225, 1242, 1259, 1308,
1285, 1351, 1322, 1220, 1337, 1294, 1178, 1253, 1296, 1296, 1343,
1229, 1300, 1296, 1278, 1303, 1336, 1358, 1336, 1270, 1282, 1322,
1311, 1281, 1256, 1267, 1292, 1307, 1312, 1269, 1311, 1293, 1223,
1222, 1235, 1263, 1268, 1303, 1280, 1267, 1285, 1325, 1193, 1315,
1285, 1224, 1229, 1211, 1319, 1227, 1362, 1336, 1206, 1327, 1301,
1218, 1281, 1401, 1216, 1349, 1241, 1348, 1298, 1332, 1273, 1259,
1383, 1253, 1139, 1245, 1290, 1301, 1312, 1329, 1316, 1325, 1256,
1300, 1343, 1243, 1324, 1285, 1286, 1259, 1315, 1283, 1288, 1303,
1277, 1320, 1335, 1309, 1283, 1264, 1193, 1271, 1316, 1219, 1298,
1273, 1260, 1323, 1264, 1325, 1288, 1274, 1186, 1281, 1269, 1301,
1334, 1250, 1262, 1252, 1316, 1396, 1252, 1287, 1279, 1288, 1313,
1335, 1297, 1316, 1257, 1316, 1244, 1349, 1192, 1325, 1341, 1315,
1290, 1303, 1282, 1239, 1295, 1290, 1305, 1245, 1309, 1225, 1340,
1214, 1224, 1269, 1296, 1292, 1291, 1256, 1303, 1283, 1230, 1335,
1295, 1342, 1236, 1297, 1282, 1287, 1278, 1273, 1282, 1314, 1241,
1330, 1335, 1327, 1294, 1416, 1299, 1291, 1311, 1384, 1335, 1253,
1295, 1277, 1338, 1270, 1265, 1339, 1294, 1225, 1236, 1234, 1297,
```

1331, 1260, 1329, 1290, 1384, 1311, 1231, 1321, 1337, 1402, 1270,
1305, 1332, 1277, 1367, 1342, 1335, 1375, 1272, 1396, 1405, 1237,
1311, 1262, 1277, 1366, 1470, 1358, 1289, 1298, 1188, 1294, 1211,
1239, 1250, 1307, 1307, 1260, 1321, 1257, 1272, 1259, 1197, 1236,
1347, 1322, 1241, 1323, 1323, 1357, 1311, 1299, 1367, 1201, 1297,
1339, 1276, 1307, 1334, 1318, 1276, 1283, 1311, 1320, 1316, 1328,
1304, 1344, 1227, 1199, 1330, 1272, 1282, 1233, 1264, 1299, 1347,
1277, 1334, 1295, 1303, 1290, 1258, 1305, 1273, 1200, 1256, 1235,
1316, 1325, 1267, 1288, 1312, 1250, 1291, 1319, 1280, 1328, 1272,
1305, 1239, 1238, 1347, 1239, 1341, 1261, 1278, 1378, 1287, 1289,
1303, 1301, 1289, 1255, 1349, 1324, 1307, 1246, 1360, 1264, 1259,
1256, 1327, 1310, 1299, 1256, 1303, 1295, 1215, 1296, 1267, 1270,
1352, 1299, 1324, 1283, 1367, 1295, 1316, 1288, 1224, 1205, 1303,
1281, 1285, 1267, 1303, 1274, 1301, 1239, 1309, 1362, 1230, 1338,
1313, 1230, 1344, 1365, 1240, 1324, 1232, 1250, 1262, 1293, 1205,
1262, 1244, 1248, 1242, 1239, 1286, 1278, 1274, 1210, 1293, 1337,
1260, 1246, 1258, 1330, 1279, 1234, 1320, 1355, 1201, 1263, 1238,
1289, 1285, 1296, 1203, 1309, 1309, 1365, 1315, 1258, 1301, 1369,
1329, 1363, 1326, 1255, 1342, 1318, 1287, 1263, 1323, 1221, 1226,
1319, 1365, 1315, 1325, 1299, 1314, 1287, 1362, 1270, 1241, 1284,
1254, 1357, 1287, 1200, 1340, 1285, 1272, 1282, 1289, 1312, 1283,
1260, 1334, 1294, 1312, 1319, 1315, 1209, 1208, 1246, 1251, 1252,
1347, 1281, 1326, 1397, 1297, 1263, 1253, 1371, 1327, 1282, 1303,
1354, 1238, 1225, 1296, 1343, 1290, 1394, 1380, 1269, 1353, 1442,
1473, 1321, 1253, 1250, 1292, 1366, 1317, 1297, 1312, 1278, 1372,
1336, 1272, 1348, 1327, 1225, 1239, 1262, 1195, 1271, 1273, 1255,
1267, 1331, 1256, 1164, 1294, 1298, 1365, 1329, 1275, 1362, 1393,
1270, 1265, 1292, 1220, 1394, 1254, 1292, 1268, 1279, 1256, 1312,
1315, 1344, 1378, 1226, 1291, 1321, 1231, 1268, 1273, 1247, 1279,
1314, 1209, 1293, 1413, 1244, 1209, 1251, 1330, 1243, 1288, 1379,
1302, 1240, 1395, 1333, 1351, 1276, 1302, 1200, 1182, 1303, 1320,
1274, 1410, 1306, 1223, 1300, 1323, 1272, 1203, 1378, 1247, 1265,
1308, 1318, 1291, 1300, 1329, 1259, 1286, 1279, 1227, 1298, 1298,
1205, 1244, 1316, 1216, 1323, 1311, 1233, 1365, 1285, 1301, 1322,
1418, 1283, 1304, 1287, 1277, 1251, 1320, 1309, 1337, 1224, 1215,
1334, 1227, 1331, 1340, 1321, 1225, 1224, 1344, 1366, 1256, 1304,
1204, 1299, 1402, 1338, 1288, 1306, 1313, 1243, 1305, 1281, 1297,
1268, 1255, 1316, 1324, 1202, 1313, 1293, 1342, 1349, 1301, 1316,
1229, 1305, 1334, 1315, 1283, 1322, 1314, 1272, 1288, 1360, 1218,
1255, 1246, 1219, 1402, 1263, 1338, 1316, 1308, 1235, 1258, 1368,
1341, 1309, 1296, 1213, 1262, 1259, 1280, 1348, 1312, 1337, 1245,
1315, 1231, 1330, 1257, 1316, 1356, 1248, 1244, 1224, 1242, 1319,
1244, 1311, 1307, 1321, 1289, 1277, 1263, 1295, 1309, 1323, 1341,
1296, 1300, 1290, 1329, 1271, 1248, 1359, 1364, 1320, 1300, 1339,

```
1309, 1270, 1256, 1215, 1326, 1275, 1293, 1334, 1352, 1353, 1234,
1280, 1336, 1233, 1352, 1321, 1312, 1260, 1248, 1202, 1231, 1276,
1283, 1330, 1264, 1211, 1300, 1276, 1325, 1295, 1264, 1287, 1307,
1294, 1305, 1289, 1272, 1266, 1318, 1260, 1330, 1288, 1313, 1280,
1371, 1323, 1376, 1320, 1330, 1346, 1275, 1308, 1255, 1317, 1215,
1269, 1245, 1377, 1250, 1232, 1315, 1321, 1315, 1273, 1364, 1352,
1331, 1296, 1296, 1311, 1355, 1233, 1306, 1267, 1213, 1339, 1290]
print("max :", max(lst))
print("min :", min(lst))
print("avg :", sum(lst)/len(lst))

print()

print("Deplan(Random):")
li = [912, 921, 924, 918, 922, 912, 918, 922, 912, 918, 913, 912,
938, 912, 912, 955, 918, 939, 914, 912, 929, 915, 912, 912, 932, 920,
919, 936, 928, 912, 912, 914, 913, 922, 912, 922, 918, 934, 922, 923,
922, 912, 936, 920, 918, 918, 919, 912, 938, 912, 912, 912, 922, 921,
912, 914, 927, 912, 919, 912, 922, 921, 912, 912, 922, 930, 922, 926,
936, 912, 912, 914, 930, 912, 912, 912, 922, 917, 912, 912, 922, 917,
927, 938, 912, 912, 918, 919, 914, 919, 929, 912, 921, 912, 921, 912,
924, 925, 916, 912, 920, 912, 921, 917, 912, 931, 914, 917, 921, 929,
937, 912, 917, 912, 912, 913, 912, 917, 918, 947, 935, 912, 915, 936,
917, 926, 930, 938, 938, 919, 912, 922, 918, 912, 912, 938, 929, 929,
928, 915, 932, 919, 928, 912, 920, 912, 920, 920, 929, 912, 917, 920,
920, 922, 929, 927, 912, 918, 916, 918, 923, 930, 926, 939, 919, 919,
928, 921, 919, 920, 929, 912, 912, 929, 912, 921, 912, 938, 915, 930,
919, 912, 921, 918, 912, 912, 916, 912, 928, 912, 921, 912, 912, 925,
912, 912, 919, 917, 912, 935, 912, 921, 926, 917, 912, 921, 912, 913,
912, 916, 912, 928, 912, 912, 931, 914, 921, 920, 926, 920, 919, 929,
912, 928, 912, 912, 912, 917, 920, 912, 924, 912, 952, 925, 915, 929,
920, 912, 912, 922, 925, 912, 916, 928, 916, 922, 912, 912, 912, 921,
918, 919, 928, 912, 939, 949, 921, 917, 937, 912, 929, 914, 922, 919,
912, 921, 912, 920, 935, 913, 912, 919, 939, 933, 912, 912, 912, 916,
912, 912, 912, 915, 939, 912, 912, 922, 922, 912, 929, 929, 919, 927,
915, 929, 912, 912, 927, 913, 926, 918, 912, 912, 912, 918, 919, 912,
912, 920, 912, 912, 912, 916, 912, 939, 912, 912, 921, 912, 925, 917,
919, 912, 920, 918, 919, 912, 912, 928, 912, 942, 925, 920, 917, 912,
922, 921, 912, 922, 939, 920, 912, 914, 912, 922, 917, 912, 927, 912,
916, 917, 927, 912, 912, 920, 920, 912, 921, 915, 923, 912, 912, 914,
912, 914, 922, 916, 923, 918, 938, 912, 912, 919, 915, 925, 928, 912,
912, 943, 936, 912, 916, 915, 912, 940, 912, 920, 926, 945, 912, 912,
912, 930, 912, 912, 922, 918, 918, 922, 918, 912, 916, 921, 922, 912,
921, 912, 918, 912, 912, 938, 920, 921, 912, 924, 927, 912, 920, 918,
```

```
921, 917, 922, 912, 920, 928, 929, 916, 912, 917, 912, 912, 920, 916,
921, 912, 941, 928, 912, 922, 924, 912, 912, 912, 912, 926, 922, 920,
912, 921, 925, 922, 928, 921, 932, 918, 912, 918, 914, 920, 912, 929,
912, 912, 923, 920, 927, 920, 925, 912, 928, 912, 918, 920, 934, 943,
939, 912, 938, 916, 916, 912, 919, 912, 922, 919, 919, 914, 930, 930,
912, 920, 912, 922, 916, 930, 912, 942, 912, 939, 920, 912, 945, 929,
912, 912, 944, 918, 912, 929, 915, 918, 912, 918, 912, 912, 920, 926,
912, 928, 921, 916, 912, 921, 912, 914, 921, 926, 918, 915, 912, 912,
912, 912, 929, 912, 912, 912, 926, 932, 912, 949, 928, 922, 947, 912,
932, 912, 912, 919, 922, 920, 918, 922, 912, 917, 927, 912, 912, 922,
921, 920, 918, 929, 912, 922, 919, 912, 920, 916, 921, 928, 912, 912,
922, 931, 917, 925, 919, 912, 915, 912, 912, 912, 912, 912, 914, 918,
912, 915, 912, 912, 928, 919, 917, 912, 916, 929, 922, 912, 926, 912,
917, 933, 912, 922, 921, 917, 950, 937, 919, 930, 912, 920, 941, 912,
919, 929, 919, 912, 912, 932, 920, 912, 920, 912, 921, 929, 930, 912,
929, 912, 912, 912, 920, 929, 912, 918, 941, 919, 912, 912, 912, 928,
912, 914, 921, 932, 916, 919, 912, 912, 916, 912, 929, 918, 912, 930,
917, 912, 912, 922, 922, 912, 937, 912, 912, 920, 912, 955, 921, 928,
920, 912, 936, 947, 922, 919, 916, 912, 912, 916, 912, 918, 912, 913,
918, 939, 915, 922, 934, 921, 931, 912, 929, 912, 920, 912, 920, 912,
912, 912, 915, 912, 912, 921, 912, 912, 917, 912, 912, 922, 912, 921,
937, 922, 918, 929, 922, 912, 912, 918, 918, 921, 922, 918, 915, 921,
923, 912, 925, 925, 925, 925, 912, 912, 912, 922, 913, 912, 915, 912,
962, 916, 919, 927, 927, 930, 921, 926, 918, 912, 930, 930, 912, 922,
912, 920, 912, 912, 912, 920, 914, 929, 912, 912, 929, 928, 916, 922,
933, 927, 912, 922, 935, 912, 919, 912, 912, 915, 912, 918, 917, 930,
925, 929, 919, 912, 912, 918, 919, 926, 912, 919, 917, 912, 939, 936,
912, 912, 912, 924, 931, 932, 919, 941, 930, 924, 912, 912, 916, 931,
920, 921, 919, 925, 912, 912, 941, 922, 912, 929, 919, 914, 930, 920,
912, 912, 927, 912, 940, 921, 912, 922, 912, 912, 922, 928, 912, 926,
924, 924, 922, 912, 916, 920, 920, 920, 912, 912, 936, 912, 920, 912,
912, 916, 916, 922, 924, 920, 912, 913, 922, 912, 920, 937, 920, 947,
912, 912, 912, 935, 930, 927, 912, 917, 912, 918, 912, 912, 919, 916,
918, 912, 916, 915, 912, 916, 926, 922, 915, 929, 930, 912, 912, 920,
912, 943, 912, 934, 912, 929, 927, 917, 912, 912, 915, 914, 920, 930,
918, 924, 940, 918, 912, 912, 912, 912, 912, 912, 912, 912, 912, 938,
917, 922, 912, 922, 912, 922, 920, 912, 912, 913, 912, 912, 912, 912,
929]
print("max :", max(li))
print("min :", min(li))
print("avg :", sum(li)/len(li))

print()
```

```
print("Boarding by Section:")
lis = [1343, 1265, 1313, 1287, 1319, 1334, 1376, 1356, 1327, 1276,
1327, 1311, 1222, 1213, 1303, 1329, 1371, 1278, 1394, 1328, 1308,
1377, 1267, 1332, 1334, 1320, 1373, 1273, 1289, 1212, 1329, 1277,
1345, 1270, 1345, 1329, 1311, 1339, 1254, 1263, 1255, 1460, 1341,
1231, 1389, 1351, 1324, 1300, 1303, 1259, 1307, 1236, 1301, 1382,
1265, 1466, 1363, 1227, 1290, 1316, 1267, 1256, 1286, 1383, 1337,
1304, 1295, 1353, 1255, 1311, 1387, 1279, 1245, 1322, 1411, 1242,
1297, 1392, 1293, 1332, 1211, 1359, 1304, 1310, 1337, 1318, 1304,
1373, 1319, 1277, 1327, 1318, 1312, 1300, 1315, 1356, 1281, 1358,
1316, 1342, 1271, 1284, 1317, 1237, 1266, 1330, 1333, 1327, 1247,
1348, 1309, 1359, 1260, 1346, 1284, 1321, 1307, 1310, 1288, 1329,
1355, 1320, 1240, 1258, 1307, 1384, 1274, 1313, 1303, 1352, 1293,
1379, 1341, 1350, 1262, 1303, 1351, 1299, 1296, 1317, 1357, 1268,
1392, 1279, 1227, 1305, 1365, 1239, 1333, 1310, 1285, 1285, 1358,
1274, 1281, 1329, 1249, 1307, 1379, 1414, 1265, 1318, 1343, 1337,
1338, 1301, 1327, 1245, 1353, 1312, 1312, 1307, 1322, 1364, 1407,
1407, 1280, 1254, 1330, 1262, 1383, 1312, 1329, 1326, 1275, 1347,
1276, 1330, 1176, 1280, 1268, 1292, 1310, 1291, 1343, 1287, 1322,
1323, 1353, 1285, 1304, 1280, 1244, 1278, 1364, 1206, 1352, 1268,
1263, 1260, 1320, 1314, 1288, 1286, 1337, 1391, 1319, 1323, 1277,
1358, 1418, 1282, 1277, 1294, 1392, 1368, 1345, 1267, 1391, 1300,
1327, 1275, 1287, 1289, 1324, 1299, 1342, 1294, 1265, 1405, 1342,
1295, 1299, 1341, 1332, 1261, 1322, 1307, 1323, 1295, 1266, 1334,
1292, 1303, 1303, 1262, 1287, 1362, 1326, 1372, 1266, 1366, 1363,
1380, 1319, 1327, 1319, 1325, 1313, 1234, 1322, 1268, 1204, 1300,
1283, 1255, 1268, 1328, 1263, 1328, 1378, 1328, 1304, 1384, 1323,
1298, 1371, 1373, 1206, 1329, 1352, 1368, 1308, 1277, 1274, 1261,
1343, 1354, 1275, 1337, 1291, 1281, 1311, 1381, 1336, 1231, 1378,
1299, 1357, 1365, 1361, 1253, 1314, 1274, 1302, 1367, 1241, 1295,
1324, 1308, 1277, 1276, 1353, 1263, 1287, 1328, 1327, 1316, 1273,
1346, 1330, 1375, 1303, 1291, 1311, 1281, 1297, 1278, 1417, 1263,
1279, 1279, 1308, 1268, 1300, 1269, 1265, 1285, 1318, 1266, 1338,
1335, 1279, 1333, 1327, 1360, 1278, 1299, 1336, 1299, 1324, 1299,
1334, 1326, 1325, 1319, 1399, 1288, 1274, 1300, 1309, 1277, 1356,
1274, 1333, 1345, 1350, 1304, 1309, 1368, 1324, 1303, 1405, 1342,
1297, 1339, 1301, 1291, 1320, 1285, 1310, 1245, 1382, 1285, 1231,
1322, 1359, 1346, 1268, 1368, 1340, 1337, 1248, 1327, 1316, 1260,
1286, 1273, 1318, 1325, 1269, 1297, 1337, 1313, 1303, 1281, 1292,
1286, 1316, 1196, 1307, 1371, 1380, 1269, 1187, 1271, 1308, 1250,
1286, 1272, 1325, 1327, 1368, 1271, 1381, 1304, 1289, 1287, 1308,
1287, 1345, 1333, 1291, 1308, 1301, 1268, 1313, 1326, 1285, 1325,
1344, 1270, 1221, 1397, 1252, 1295, 1218, 1342, 1296, 1279, 1232,
1255, 1325, 1303, 1381, 1257, 1369, 1376, 1238, 1295, 1326, 1352,
```

1229, 1362, 1319, 1345, 1335, 1296, 1232, 1328, 1358, 1365, 1306,
1363, 1261, 1355, 1348, 1265, 1337, 1223, 1310, 1328, 1360, 1310,
1413, 1305, 1313, 1331, 1314, 1308, 1323, 1328, 1319, 1374, 1287,
1310, 1268, 1244, 1279, 1229, 1344, 1386, 1227, 1280, 1329, 1299,
1281, 1350, 1271, 1365, 1262, 1307, 1309, 1309, 1277, 1286, 1376,
1355, 1368, 1379, 1309, 1327, 1279, 1353, 1275, 1374, 1369, 1316,
1321, 1311, 1371, 1339, 1255, 1307, 1312, 1313, 1261, 1357, 1299,
1336, 1307, 1304, 1269, 1343, 1280, 1231, 1351, 1329, 1409, 1284,
1348, 1385, 1228, 1261, 1266, 1355, 1305, 1333, 1290, 1257, 1325,
1349, 1297, 1332, 1258, 1320, 1330, 1376, 1368, 1287, 1308, 1238,
1371, 1326, 1303, 1363, 1309, 1322, 1251, 1336, 1350, 1341, 1417,
1328, 1260, 1304, 1306, 1288, 1365, 1377, 1276, 1322, 1365, 1333,
1284, 1346, 1335, 1313, 1219, 1331, 1324, 1368, 1358, 1256, 1270,
1281, 1337, 1316, 1283, 1329, 1286, 1262, 1362, 1351, 1266, 1245,
1395, 1258, 1322, 1322, 1251, 1337, 1426, 1296, 1256, 1275, 1346,
1375, 1319, 1352, 1380, 1272, 1263, 1295, 1253, 1354, 1308, 1406,
1384, 1285, 1292, 1333, 1289, 1248, 1312, 1311, 1291, 1278, 1353,
1345, 1300, 1372, 1284, 1245, 1417, 1400, 1312, 1274, 1316, 1352,
1304, 1325, 1327, 1336, 1292, 1385, 1336, 1366, 1223, 1322, 1281,
1322, 1261, 1336, 1323, 1254, 1304, 1286, 1349, 1405, 1279, 1346,
1337, 1341, 1262, 1290, 1359, 1195, 1362, 1424, 1287, 1322, 1299,
1355, 1259, 1336, 1284, 1340, 1335, 1292, 1280, 1307, 1277, 1318,
1257, 1306, 1229, 1222, 1335, 1307, 1374, 1265, 1302, 1351, 1226,
1316, 1282, 1299, 1311, 1295, 1312, 1361, 1223, 1246, 1340, 1280,
1296, 1321, 1318, 1320, 1321, 1348, 1316, 1391, 1360, 1341, 1244,
1346, 1357, 1233, 1383, 1265, 1355, 1325, 1333, 1363, 1325, 1330,
1374, 1265, 1309, 1311, 1296, 1312, 1312, 1259, 1235, 1348, 1327,
1230, 1281, 1297, 1348, 1334, 1362, 1315, 1313, 1297, 1301, 1206,
1320, 1369, 1349, 1349, 1343, 1252, 1315, 1317, 1292, 1309, 1305,
1250, 1217, 1239, 1298, 1337, 1272, 1356, 1368, 1218, 1311, 1357,
1310, 1261, 1331, 1301, 1300, 1317, 1286, 1297, 1351, 1278, 1336,
1318, 1367, 1282, 1372, 1290, 1326, 1229, 1263, 1348, 1299, 1252,
1386, 1305, 1347, 1280, 1208, 1241, 1279, 1310, 1297, 1305, 1273,
1348, 1361, 1293, 1336, 1270, 1244, 1363, 1355, 1275, 1215, 1310,
1279, 1309, 1351, 1313, 1232, 1305, 1266, 1297, 1314, 1280, 1255,
1362, 1305, 1341, 1257, 1366, 1279, 1272, 1311, 1301, 1260, 1261,
1362, 1244, 1242, 1342, 1284, 1257, 1300, 1336, 1385, 1323, 1288,
1331, 1327, 1318, 1344, 1276, 1335, 1305, 1368, 1323, 1320, 1255,
1315, 1289, 1280, 1306, 1274, 1224, 1341, 1316, 1302, 1299, 1352,
1334, 1332, 1310, 1307, 1405, 1213, 1324, 1310, 1359, 1359, 1416,
1306, 1260, 1276, 1326, 1275, 1341, 1283, 1271, 1275, 1252, 1301,
1313, 1283, 1299, 1356, 1338, 1302, 1308, 1310, 1317, 1238, 1294,
1408, 1289, 1370, 1340, 1323, 1274, 1379, 1368, 1309, 1322, 1320,
1344, 1320, 1289, 1297, 1356, 1309, 1317, 1257, 1331, 1355, 1326,

```
1356, 1320, 1327, 1334, 1349, 1324, 1301, 1263, 1286, 1285, 1324,
1286, 1302, 1255, 1265, 1340, 1259, 1334, 1291, 1362, 1302, 1308,
1395, 1305, 1266, 1380, 1293, 1341, 1333, 1310, 1401, 1333, 1308,
1353, 1330, 1282, 1352, 1238, 1345, 1391, 1274, 1308, 1404, 1303]
print("max :", max(lis))
print("min :", min(lis))
print("avg :", sum(lis)/len(lis))

print()

print("Boarding by Seat:")
ls = [1095, 1125, 1125, 1140, 1159, 1133, 1121, 1153, 1156, 1152,
1117, 1094, 1186, 1159, 1107, 1125, 1141, 1136, 1096, 1152, 1073,
1085, 1050, 1166, 1180, 1155, 1162, 1119, 1145, 1154, 1108, 1097,
1150, 1136, 1202, 1160, 1196, 1110, 1138, 1123, 1091, 1113, 1117,
1105, 1150, 1140, 1108, 1138, 1165, 1150, 1129, 1114, 1207, 1143,
1119, 1142, 1103, 1108, 1103, 1158, 1122, 1136, 1153, 1111, 1111,
1126, 1123, 1100, 1131, 1109, 1117, 1158, 1134, 1114, 1108, 1100,
1176, 1141, 1150, 1178, 1211, 1121, 1120, 1130, 1107, 1136, 1090,
1143, 1066, 1168, 1145, 1151, 1141, 1111, 1162, 1084, 1164, 1161,
1170, 1108, 1181, 1108, 1075, 1136, 1156, 1157, 1122, 1102, 1109,
1156, 1125, 1132, 1130, 1125, 1110, 1126, 1142, 1063, 1113, 1137,
1130, 1121, 1114, 1118, 1094, 1152, 1126, 1142, 1134, 1153, 1083,
1110, 1110, 1061, 1140, 1165, 1086, 1089, 1178, 1124, 1167, 1115,
1134, 1146, 1128, 1143, 1139, 1142, 1131, 1092, 1135, 1092, 1126,
1158, 1129, 1114, 1120, 1100, 1103, 1104, 1043, 1101, 1097, 1161,
1137, 1175, 1113, 1133, 1117, 1160, 1122, 1093, 1186, 1108, 1169,
1134, 1126, 1132, 1106, 1121, 1094, 1076, 1112, 1137, 1138, 1125,
1144, 1223, 1119, 1088, 1141, 1113, 1118, 1096, 1156, 1165, 1108,
1121, 1180, 1159, 1150, 1142, 1152, 1092, 1154, 1136, 1126, 1123,
1132, 1110, 1120, 1088, 1130, 1141, 1126, 1137, 1135, 1131, 1112,
1186, 1127, 1134, 1142, 1127, 1139, 1174, 1066, 1127, 1109, 1171,
1147, 1103, 1094, 1204, 1173, 1094, 1139, 1100, 1125, 1164, 1127,
1117, 1161, 1145, 1125, 1107, 1099, 1060, 1116, 1110, 1138, 1166,
1076, 1138, 1120, 1122, 1111, 1102, 1158, 1160, 1081, 1139, 1201,
1099, 1165, 1075, 1103, 1115, 1135, 1121, 1169, 1140, 1141, 1110,
1159, 1168, 1123, 1155, 1070, 1072, 1115, 1071, 1104, 1121, 1139,
1104, 1187, 1159, 1089, 1149, 1121, 1135, 1137, 1129, 1112, 1088,
1164, 1158, 1107, 1127, 1114, 1151, 1168, 1132, 1114, 1129, 1108,
1124, 1162, 1113, 1121, 1175, 1118, 1131, 1158, 1188, 1161, 1119,
1112, 1138, 1104, 1094, 1133, 1128, 1079, 1087, 1142, 1150, 1168,
1127, 1141, 1118, 1083, 1123, 1068, 1136, 1121, 1131, 1128, 1126,
1084, 1081, 1156, 1092, 1148, 1113, 1130, 1147, 1100, 1098, 1153,
1165, 1111, 1161, 1155, 1122, 1164, 1059, 1139, 1121, 1177, 1148,
```

1089, 1097, 1094, 1138, 1136, 1080, 1159, 1169, 1101, 1116, 1121,
1150, 1082, 1103, 1113, 1128, 1149, 1126, 1169, 1099, 1204, 1120,
1098, 1109, 1131, 1133, 1100, 1129, 1125, 1100, 1114, 1112, 1120,
1124, 1082, 1066, 1103, 1080, 1142, 1146, 1142, 1179, 1092, 1149,
1117, 1115, 1142, 1132, 1117, 1105, 1126, 1164, 1102, 1091, 1098,
1136, 1112, 1104, 1121, 1119, 1180, 1111, 1094, 1155, 1125, 1115,
1135, 1068, 1161, 1056, 1149, 1062, 1154, 1114, 1129, 1146, 1168,
1134, 1137, 1172, 1129, 1175, 1176, 1149, 1148, 1092, 1113, 1115,
1067, 1136, 1162, 1087, 1116, 1117, 1105, 1055, 1097, 1098, 1084,
1117, 1148, 1110, 1139, 1125, 1097, 1173, 1135, 1089, 1091, 1158,
1092, 1087, 1153, 1121, 1117, 1098, 1109, 1107, 1097, 1146, 1136,
1091, 1184, 1104, 1138, 1149, 1150, 1103, 1090, 1123, 1108, 1182,
1158, 1134, 1134, 1137, 1149, 1131, 1118, 1134, 1136, 1159, 1118,
1089, 1108, 1120, 1171, 1153, 1085, 1121, 1166, 1113, 1095, 1111,
1184, 1134, 1135, 1131, 1133, 1120, 1058, 1151, 1150, 1100, 1149,
1132, 1111, 1124, 1095, 1169, 1136, 1137, 1145, 1089, 1105, 1129,
1091, 1068, 1107, 1203, 1136, 1108, 1185, 1126, 1114, 1131, 1169,
1085, 1084, 1146, 1107, 1086, 1098, 1206, 1132, 1129, 1140, 1064,
1149, 1168, 1151, 1141, 1075, 1094, 1164, 1148, 1108, 1137, 1121,
1127, 1100, 1153, 1152, 1104, 1146, 1129, 1149, 1120, 1107, 1115,
1071, 1128, 1180, 1073, 1049, 1118, 1084, 1133, 1106, 1124, 1165,
1136, 1091, 1086, 1154, 1074, 1163, 1124, 1161, 1076, 1111, 1137,
1111, 1118, 1132, 1119, 1151, 1100, 1197, 1139, 1066, 1136, 1129,
1150, 1130, 1144, 1185, 1115, 1118, 1133, 1120, 1099, 1096, 1135,
1122, 1120, 1170, 1113, 1153, 1139, 1114, 1170, 1085, 1113, 1134,
1140, 1157, 1061, 1132, 1111, 1134, 1079, 1157, 1117, 1111, 1179,
1168, 1102, 1110, 1094, 1131, 1117, 1185, 1099, 1112, 1138, 1135,
1181, 1120, 1132, 1099, 1059, 1147, 1109, 1122, 1076, 1076, 1112,
1145, 1123, 1105, 1161, 1192, 1149, 1103, 1146, 1051, 1167, 1127,
1136, 1145, 1102, 1093, 1126, 1112, 1128, 1120, 1139, 1151, 1105,
1152, 1090, 1145, 1137, 1125, 1116, 1098, 1155, 1116, 1160, 1179,
1167, 1060, 1063, 1098, 1058, 1139, 1146, 1131, 1107, 1154, 1137,
1214, 1129, 1110, 1094, 1136, 1095, 1112, 1102, 1170, 1162, 1138,
1186, 1147, 1133, 1086, 1137, 1140, 1177, 1141, 1143, 1142, 1127,
1119, 1090, 1137, 1113, 1122, 1168, 1104, 1150, 1114, 1158, 1148,
1111, 1099, 1057, 1138, 1144, 1075, 1116, 1115, 1138, 1157, 1133,
1202, 1147, 1159, 1119, 1110, 1109, 1118, 1128, 1088, 1164, 1122,
1142, 1168, 1119, 1091, 1171, 1125, 1187, 1097, 1097, 1080, 1113,
1125, 1129, 1137, 1102, 1144, 1165, 1115, 1128, 1117, 1162, 1146,
1092, 1092, 1159, 1144, 1121, 1135, 1139, 1118, 1128, 1110, 1150,
1095, 1128, 1145, 1131, 1075, 1115, 1089, 1145, 1162, 1092, 1152,
1103, 1129, 1135, 1153, 1130, 1103, 1152, 1115, 1119, 1138, 1093,
1119, 1135, 1079, 1112, 1097, 1121, 1103, 1147, 1102, 1122, 1134,
1061, 1132, 1123, 1121, 1136, 1139, 1093, 1152, 1169, 1144, 1120,

```
1157, 1146, 1106, 1171, 1171, 1117, 1132, 1149, 1118, 1082, 1101,
1104, 1102, 1102, 1119, 1156, 1108, 1188, 1122, 1139, 1086, 1150,
1168, 1159, 1111, 1154, 1109, 1134, 1125, 1114, 1145, 1113, 1162,
1133, 1120, 1175, 1139, 1122, 1109, 1145, 1103, 1135, 1186, 1147,
1184, 1139, 1035, 1114, 1092, 1150, 1074, 1159, 1112, 1120, 1128,
1123, 1109, 1133, 1117, 1142, 1149, 1128, 1116, 1126, 1159, 1114,
1107, 1121, 1154, 1090, 1123, 1106, 1129, 1100, 1127, 1089, 1116,
1112, 1094, 1119, 1124, 1112, 1146, 1106, 1160, 1173, 1144, 1112,
1100, 1130, 1155, 1092, 1110, 1094, 1112, 1104, 1113, 1139, 1087,
1102, 1133, 1108, 1145, 1148, 1116, 1084, 1116, 1145, 1115, 1140,
1115, 1117, 1137, 1143, 1151, 1177, 1135, 1106, 1165, 1094, 1220,
1096, 1169, 1095, 1142, 1112, 1127, 1181, 1147, 1121, 1124, 1188,
1200, 1137, 1133, 1141, 1141, 1159, 1121, 1114, 1101, 1109, 1100,
1117, 1113, 1139, 1110, 1162, 1090, 1101, 1115, 1111, 1123, 1128]
print("max :", max(ls))
print("min :", min(ls))
print("avg :", sum(ls)/len(ls))

print()

print("Boarding by Steffen:")
ls = [832]
print("max :", max(ls))
print("min :", min(ls))
print("avg :", sum(ls)/len(ls))

print()

print("Boarding by InventedPyramid:")
lt = [1142, 1112, 1039, 1079, 1038, 1091, 1104, 1079, 1140, 1117,
1071, 1037, 1042, 1061, 1050, 992, 1026, 1047, 1082, 1097, 1047,
1033, 1079, 1131, 1080, 1068, 1081, 1082, 1122, 1053, 1107, 1103,
1085, 1084, 1048, 1045, 1016, 1102, 1074, 1085, 1055, 1126, 1040,
1058, 1115, 1052, 1015, 1104, 1056, 1081, 1069, 1065, 1056, 1038,
1045, 1113, 1089, 1077, 1089, 1051, 1101, 1086, 1101, 1077, 1068,
1057, 1058, 1089, 1116, 1083, 1061, 1056, 1065, 1055, 1055, 1054,
1076, 1112, 1098, 1065, 1075, 1086, 1120, 1053, 1073, 1052, 1009,
1091, 1104, 1143, 1086, 1133, 1074, 1041, 1037, 1086, 1067, 1069,
1100, 1114, 1067, 1039, 1087, 1067, 1034, 1071, 1052, 1076, 1052,
1059, 1088, 1083, 1047, 1054, 1063, 1061, 1034, 1070, 1097, 1080,
1047, 1099, 1049, 1075, 1045, 1059, 1025, 1048, 1077, 1046, 1063,
1073, 1056, 1141, 999, 1070, 1050, 1070, 1067, 1036, 1023, 1065,
1015, 1080, 1097, 1058, 1095, 1050, 1054, 1080, 1007, 1051, 1032,
1100, 1061, 1100, 1047, 1121, 1110, 1048, 1023, 1059, 1016, 1070,
```

1030, 1134, 1028, 1111, 1089, 1059, 1030, 1068, 1081, 1051, 1103,
1080, 1021, 1075, 1083, 1139, 1079, 1062, 1072, 1117, 1074, 1117,
1053, 1067, 1064, 1015, 1054, 1077, 1073, 1039, 1058, 1068, 1044,
1032, 1112, 1037, 1076, 1087, 1098, 1089, 1071, 1036, 1081, 1060,
1036, 1062, 1114, 1113, 1094, 1091, 1081, 1113, 1065, 1085, 1062,
1047, 1091, 1054, 1100, 1050, 1055, 1102, 1059, 1109, 1049, 1105,
1074, 1047, 1109, 1078, 1073, 1072, 1093, 1065, 1059, 1092, 1075,
1112, 1061, 1067, 1033, 1080, 1125, 1071, 1050, 1069, 1087, 1055,
1081, 1064, 1045, 1059, 1070, 1164, 1060, 1106, 1109, 1017, 1092,
1061, 1041, 1027, 1062, 1039, 1124, 1048, 1045, 1054, 1066, 1027,
1091, 1093, 1112, 1063, 1095, 1027, 1063, 1031, 1102, 1048, 1070,
1048, 1083, 1085, 1046, 1046, 1075, 1109, 1064, 1046, 1041, 1097,
1034, 1107, 1036, 1095, 1067, 1047, 1134, 1031, 1106, 1067, 1076,
1082, 1049, 1025, 1040, 1092, 1094, 1059, 1100, 1071, 1085, 989,
1065, 1046, 1089, 1050, 1027, 1034, 1041, 1039, 1074, 1071, 1052,
1044, 1040, 1094, 1043, 1079, 1146, 1060, 1061, 1026, 1070, 1067,
1025, 1090, 989, 1041, 1056, 1035, 1097, 1057, 1079, 1062, 1090,
1061, 1046, 1115, 1045, 1108, 1049, 1085, 1090, 1017, 1082, 1067,
1034, 1064, 1095, 1055, 1053, 1044, 1026, 1080, 1086, 1048, 1069,
1037, 1091, 1057, 1071, 1094, 1133, 1072, 998, 1071, 1106, 1021,
1130, 1082, 1052, 1043, 1109, 1105, 1047, 1102, 1027, 1079, 1035,
1068, 1037, 1057, 1081, 1081, 1069, 1069, 1040, 1011, 1092, 1088,
1063, 1083, 1088, 1041, 1038, 1074, 1016, 1084, 1114, 1095, 1092,
993, 1106, 1032, 1075, 1053, 1058, 1070, 1074, 1108, 1075, 1098,
1064, 1064, 1057, 1093, 1055, 1059, 1123, 1074, 1063, 1072, 1034,
1020, 1082, 1028, 1074, 997, 1087, 1080, 1073, 1061, 1048, 1065,
1030, 1074, 1109, 1048, 1065, 1048, 1040, 1047, 1129, 1128, 1060,
1050, 1012, 1053, 1081, 1073, 1050, 1093, 1100, 1051, 1046, 1055,
1045, 1090, 1057, 1032, 1044, 1058, 1042, 1081, 1073, 1074, 1103,
1094, 1012, 1092, 1073, 998, 1036, 999, 1060, 1007, 1076, 1105, 1021,
1109, 1054, 1082, 1097, 1141, 1059, 1041, 1058, 1072, 1054, 1050,
1100, 1022, 1082, 1113, 1084, 1112, 1135, 1071, 1059, 1059, 1084,
1032, 1073, 1033, 1072, 1096, 1056, 1111, 1101, 1059, 1060, 1042,
1052, 1037, 1037, 1060, 1020, 1100, 1071, 1027, 1080, 1080, 1053,
1120, 1070, 1107, 1065, 1084, 1018, 998, 1047, 1057, 1050, 1055,
1129, 1074, 1082, 1052, 1029, 1007, 1067, 1068, 1026, 1074, 1072,
998, 1008, 1082, 1043, 1068, 1102, 1115, 1064, 1056, 1038, 1097,
1097, 1040, 1083, 1095, 1047, 1030, 1058, 1075, 1076, 1117, 1063,
1022, 1077, 1064, 1037, 1139, 1075, 1113, 1038, 1101, 1019, 1037,
1051, 1101, 1050, 1033, 1029, 1049, 1002, 1058, 1066, 1057, 1098,
1042, 1062, 1055, 1097, 1066, 1081, 1014, 1064, 1073, 1062, 1084,
1043, 1039, 1080, 1074, 1052, 1085, 976, 1068, 1054, 1031, 1042,
1078, 1034, 1138, 1099, 1065, 1094, 1062, 1056, 1081, 1080, 1086,
1093, 1025, 1050, 1040, 1068, 1044, 1017, 1094, 1086, 1121, 1049,

```
1059, 1062, 1033, 1063, 1052, 1058, 1056, 1110, 1058, 1134, 1065,
1099, 1132, 1027, 1097, 1043, 1068, 1078, 1096, 1078, 1058, 1085,
1058, 1063, 1045, 1020, 1106, 1059, 1057, 1085, 1043, 1085, 1044,
1064, 1053, 1129, 1055, 1110, 1064, 1084, 1128, 1068, 1096, 1068,
1072, 1078, 1038, 1050, 1028, 1065, 1085, 1099, 1042, 1063, 1040,
1054, 1077, 1038, 1068, 1013, 1044, 1060, 1093, 1078, 1047, 996,
1072, 1070, 1028, 1081, 1119, 1046, 1088, 1054, 1033, 999, 1099,
1037, 1083, 1067, 1049, 1078, 1061, 1053, 1090, 1074, 1080, 1076,
1028, 1086, 1079, 1076, 1018, 1097, 1063, 1093, 1053, 1114, 1086,
1081, 1098, 1041, 1061, 1056, 1071, 1053, 1031, 1079, 1049, 1066,
1089, 1065, 1065, 1093, 1085, 1057, 1067, 1044, 1041, 1077, 1014,
1064, 1054, 1075, 1045, 1072, 1111, 1026, 1023, 1045, 1057, 1061,
1063, 1113, 1113, 1071, 1045, 1037, 1067, 1003, 1079, 1042, 1093,
1086, 1048, 1043, 1033, 1041, 1054, 1029, 1042, 1051, 1064, 1022,
1073, 1042, 1080, 1090, 1100, 1033, 1067, 1061, 1064, 1056, 1103,
1080, 1070, 1064, 1032, 1079, 1018, 1073, 1046, 1115, 1035, 1051,
1061, 1069, 1065, 1073, 1069, 1045, 1096, 1064, 1089, 1037, 1123,
1061, 991, 1041, 1088, 1055, 1056, 1116, 1083, 1065, 1077, 1062,
1072, 1068, 1113, 1032, 1048, 1022, 1086, 1077, 1060, 1027, 1021,
1088, 1077, 1069, 1053, 1023, 1044, 1024, 1064, 1070, 1026, 1104,
1061, 1092, 1097, 1069, 1122, 1069, 1060, 1047, 1057, 1058, 1049,
1040, 1130, 1069, 1052, 1069, 1089, 1078, 1052, 1073, 1045, 1053,
1047, 1064, 1084, 1073, 1033, 1031, 1031, 1040, 1073, 1059, 1027,
1065, 1099, 1096, 1061, 1081, 1083, 1101, 1084, 1053, 1085, 1057,
1074, 1026, 1056, 1049, 1032, 1083, 1044, 1080, 1038, 1095, 1091,
1072, 1104, 1117, 1044, 1007, 1008, 1105, 1058, 1035, 1081, 1049,
1077, 1092, 1106, 1053, 1096, 1125, 1048, 1052, 1024, 1079, 1082,
1079, 1050, 1052, 1041, 1057, 1089, 1069, 1070, 1035, 1079, 1084,
1064, 1066, 1046, 1037, 1105, 1062, 1051, 1071, 1055, 1037, 1088,
1041, 1084, 1042, 1041, 1045, 1078, 1092, 1109, 1087, 1016, 1060,
1064, 1096, 1104, 1039, 1105, 1082, 1074, 1046, 1024, 1063, 1039,
1071, 1092, 1112, 1085, 1059, 1089, 1041, 1127, 1067, 1064]
print("max :", max(lt))
print("min :", min(lt))
print("avg :", sum(lt)/len(lt))

print()

print("Boarding by Random (70% only):")
lstt = [862, 824, 814, 833, 827, 818, 806, 863, 863, 832, 873, 773,
797, 825, 765, 871, 812, 827, 803, 836, 841, 843, 894, 821, 815, 843,
850, 861, 830, 759, 839, 770, 804, 827, 885, 813, 827, 876, 807, 831,
821, 808, 831, 830, 792, 777, 807, 855, 854, 812, 794, 828, 767, 865,
817, 800, 818, 799, 853, 808, 827, 760, 830, 783, 818, 790, 826, 777,
```

837, 787, 835, 797, 806, 791, 858, 901, 868, 859, 820, 851, 825, 788,
764, 742, 826, 830, 759, 830, 838, 858, 843, 826, 832, 802, 837, 817,
857, 791, 794, 823, 823, 779, 826, 841, 793, 743, 820, 829, 862, 851,
863, 862, 873, 820, 875, 844, 843, 807, 821, 798, 783, 822, 762, 773,
746, 804, 789, 774, 781, 818, 802, 811, 786, 856, 861, 779, 871, 794,
826, 836, 847, 833, 847, 803, 856, 795, 803, 805, 847, 745, 845, 781,
791, 852, 780, 787, 827, 846, 822, 822, 839, 836, 815, 753, 795, 831,
832, 804, 788, 802, 818, 884, 780, 841, 846, 805, 810, 808, 831, 816,
826, 796, 818, 781, 807, 816, 855, 852, 799, 825, 796, 805, 845, 827,
876, 855, 829, 835, 810, 820, 808, 821, 797, 823, 803, 799, 797, 782,
837, 785, 849, 827, 819, 844, 817, 881, 780, 828, 818, 833, 858, 826,
821, 848, 839, 865, 770, 760, 808, 839, 851, 829, 857, 859, 788, 823,
816, 840, 803, 869, 809, 822, 838, 827, 887, 795, 828, 780, 837, 824,
874, 799, 809, 768, 800, 779, 810, 824, 772, 813, 835, 884, 869, 787,
828, 843, 808, 780, 826, 809, 772, 768, 830, 829, 844, 813, 846, 758,
742, 766, 878, 857, 862, 800, 840, 783, 841, 829, 789, 852, 844, 837,
885, 856, 773, 781, 836, 824, 788, 870, 807, 836, 851, 807, 811, 820,
870, 791, 785, 813, 828, 783, 822, 788, 816, 783, 834, 798, 744, 825,
828, 838, 781, 865, 828, 793, 814, 812, 782, 827, 837, 802, 809, 863,
861, 793, 811, 817, 760, 791, 862, 817, 891, 818, 832, 737, 825, 806,
774, 868, 834, 819, 848, 785, 852, 834, 826, 808, 861, 806, 825, 820,
808, 812, 792, 854, 818, 824, 778, 827, 768, 863, 810, 817, 792, 869,
843, 829, 802, 774, 826, 805, 840, 854, 770, 808, 796, 796, 808, 785,
826, 794, 766, 806, 788, 818, 873, 782, 819, 875, 785, 818, 798, 822,
813, 875, 827, 870, 840, 842, 786, 810, 813, 835, 844, 834, 876, 787,
810, 809, 852, 821, 822, 808, 868, 807, 809, 839, 802, 826, 800, 801,
833, 798, 796, 836, 815, 817, 798, 827, 820, 808, 787, 768, 784, 847,
888, 864, 822, 876, 803, 806, 838, 814, 811, 801, 819, 814, 809, 811,
813, 831, 799, 875, 827, 793, 820, 850, 859, 819, 826, 813, 772, 791,
773, 814, 765, 822, 765, 827, 811, 830, 814, 826, 831, 839, 790, 781,
817, 847, 856, 800, 858, 823, 818, 843, 787, 864, 787, 785, 825, 828,
795, 813, 831, 837, 815, 783, 809, 874, 813, 833, 851, 835, 821, 829,
752, 817, 778, 740, 785, 868, 784, 751, 781, 826, 794, 810, 875, 866,
879, 875, 836, 848, 859, 804, 833, 804, 815, 827, 815, 833, 791, 845,
769, 750, 845, 785, 844, 805, 773, 815, 810, 831, 786, 837, 797, 792,
840, 851, 822, 782, 823, 811, 804, 843, 798, 831, 857, 789, 829, 736,
806, 807, 857, 798, 817, 815, 823, 801, 863, 817, 830, 796, 846, 865,
873, 893, 799, 875, 798, 782, 830, 902, 725, 804, 809, 826, 816, 840,
840, 833, 872, 821, 843, 827, 877, 810, 834, 791, 830, 838, 730, 943,
833, 771, 827, 837, 824, 762, 830, 779, 885, 799, 813, 781, 854, 800,
856, 875, 778, 749, 879, 800, 775, 848, 786, 778, 767, 853, 875, 804,
800, 862, 818, 824, 807, 821, 784, 791, 824, 755, 892, 816, 837, 818,
825, 852, 869, 853, 776, 800, 885, 848, 809, 824, 838, 790, 831, 769,
832, 817, 823, 780, 851, 803, 808, 819, 835, 812, 788, 836, 824, 791,

```
805, 848, 819, 808, 876, 779, 813, 762, 843, 779, 878, 784, 824, 845,
795, 794, 779, 863, 876, 857, 875, 786, 819, 832, 884, 820, 865, 831,
776, 825, 811, 848, 851, 783, 859, 808, 750, 757, 784, 783, 859, 824,
841, 849, 833, 820, 853, 834, 885, 860, 853, 833, 813, 824, 785, 850,
803, 817, 843, 822, 837, 810, 831, 762, 812, 850, 797, 826, 834, 814,
844, 781, 834, 821, 812, 798, 839, 864, 815, 819, 839, 859, 847, 765,
781, 793, 835, 890, 787, 781, 800, 838, 813, 797, 813, 762, 764, 856,
886, 826, 848, 804, 866, 833, 789, 822, 833, 772, 817, 820, 829, 868,
838, 771, 879, 807, 846, 856, 806, 775, 802, 804, 873, 856, 824, 786,
828, 787, 798, 805, 891, 860, 807, 805, 763, 834, 825, 818, 734, 858,
792, 834, 800, 809, 807, 876, 853, 793, 846, 791, 827, 846, 794, 833,
789, 802, 788, 812, 810, 835, 842, 806, 795, 769, 779, 798, 835, 844,
825, 809, 809, 844, 841, 834, 835, 807, 804, 803, 865, 778, 788, 804,
828, 855, 842, 795, 816, 785, 829, 822, 813, 843, 798, 774, 860, 824,
834, 854, 813, 844, 780, 818, 815, 844, 790, 870, 828, 789, 766, 764,
831, 844, 837, 813, 729, 860, 792, 780, 804, 816, 874, 794, 801, 828,
786, 773, 812, 845, 830, 864, 799, 861, 867, 792, 811, 866, 849, 853,
825, 787, 823, 833, 833, 786, 842, 855, 849, 852, 845, 782, 766, 803,
831, 838, 865, 829, 811, 869, 859, 837, 800, 869, 874, 852, 861, 836,
834, 806, 810, 856, 757, 825, 777, 734, 847, 849, 839, 797, 878, 826,
833, 769, 738, 827, 824, 799, 788, 831, 837, 799, 797, 799, 767, 836,
774, 779, 740, 832, 821, 802, 801, 827, 891, 805, 734, 782, 842, 757,
806, 813, 828, 873, 821, 853, 805, 840]
print("max :", max(lstt))
print("min :", min(lstt))
print("avg :", sum(lstt)/len(lstt))

print()

print("Boarding by Random (50% only):")
lii = [579, 639, 606, 608, 590, 638, 618, 638, 590, 595, 606, 630,
617, 590, 592, 628, 595, 571, 614, 633, 604, 610, 522, 598, 605, 598,
634, 615, 645, 626, 598, 607, 607, 598, 595, 625, 551, 587, 648, 603,
624, 596, 605, 608, 590, 634, 617, 650, 597, 615, 638, 600, 603, 619,
607, 582, 573, 563, 604, 647, 605, 573, 615, 576, 576, 607, 573, 616,
589, 582, 589, 624, 587, 624, 605, 635, 645, 654, 618, 642, 603, 546,
586, 611, 561, 664, 605, 673, 645, 648, 559, 627, 671, 620, 617, 563,
596, 638, 588, 621, 620, 581, 575, 599, 603, 602, 643, 613, 643, 599,
619, 580, 632, 604, 639, 640, 604, 583, 612, 606, 583, 539, 590, 626,
610, 626, 603, 590, 679, 593, 654, 658, 587, 677, 622, 575, 648, 647,
607, 595, 620, 646, 629, 657, 637, 599, 618, 610, 627, 577, 625, 617,
581, 569, 588, 566, 596, 641, 633, 640, 605, 613, 606, 619, 624, 614,
641, 578, 610, 557, 621, 617, 599, 572, 609, 613, 587, 577, 571, 622,
642, 611, 619, 581, 620, 630, 561, 614, 547, 593, 608, 672, 631, 607,
```

602, 616, 655, 635, 611, 651, 573, 609, 616, 622, 600, 612, 597, 616,
619, 562, 610, 578, 588, 611, 564, 640, 599, 601, 588, 666, 603, 598,
608, 572, 558, 625, 607, 598, 612, 609, 593, 591, 613, 637, 607, 621,
662, 642, 614, 610, 642, 617, 600, 625, 594, 602, 595, 606, 669, 628,
621, 557, 590, 644, 589, 574, 611, 642, 560, 601, 594, 606, 563, 569,
637, 603, 594, 610, 643, 599, 598, 603, 620, 687, 619, 578, 602, 621,
632, 585, 600, 622, 604, 586, 611, 615, 625, 586, 594, 581, 571, 600,
647, 613, 625, 620, 580, 611, 575, 596, 606, 616, 697, 607, 607, 589,
564, 619, 592, 647, 566, 598, 567, 600, 616, 588, 617, 611, 628, 625,
656, 630, 646, 572, 601, 609, 609, 586, 532, 601, 573, 595, 602, 630,
605, 652, 598, 577, 612, 597, 638, 581, 591, 566, 592, 598, 588, 566,
602, 634, 599, 596, 644, 611, 652, 602, 588, 614, 621, 585, 622, 612,
552, 601, 600, 614, 593, 571, 600, 622, 589, 635, 589, 630, 607, 613,
620, 608, 582, 589, 613, 615, 568, 614, 621, 612, 590, 595, 631, 598,
604, 618, 629, 597, 622, 618, 556, 631, 604, 609, 633, 648, 613, 633,
637, 674, 569, 585, 630, 582, 572, 622, 584, 669, 579, 623, 645, 701,
645, 588, 594, 577, 566, 599, 603, 609, 606, 643, 592, 596, 626, 599,
658, 584, 622, 615, 608, 608, 595, 585, 594, 587, 649, 632, 624, 591,
588, 576, 600, 616, 599, 590, 633, 631, 573, 613, 620, 655, 627, 607,
605, 615, 649, 634, 641, 589, 581, 604, 644, 581, 630, 605, 605, 591,
600, 617, 602, 606, 588, 603, 629, 631, 587, 601, 610, 619, 577, 650,
644, 600, 609, 618, 547, 658, 623, 617, 609, 575, 659, 580, 574, 616,
640, 645, 678, 602, 595, 599, 575, 581, 656, 632, 607, 634, 610, 622,
608, 636, 618, 618, 638, 617, 582, 627, 636, 603, 602, 591, 595, 619,
630, 566, 635, 607, 603, 642, 605, 621, 587, 621, 594, 572, 556, 706,
620, 617, 604, 605, 646, 574, 644, 578, 587, 621, 584, 581, 617, 600,
603, 616, 591, 654, 640, 614, 609, 625, 602, 633, 593, 573, 616, 616,
634, 606, 599, 588, 642, 601, 624, 555, 615, 630, 663, 613, 575, 657,
636, 604, 638, 626, 624, 634, 618, 619, 607, 632, 588, 620, 553, 683,
619, 592, 590, 569, 640, 616, 624, 603, 639, 606, 615, 590, 604, 686,
588, 580, 593, 611, 615, 648, 628, 624, 599, 589, 571, 555, 580, 608,
590, 638, 547, 672, 682, 589, 600, 580, 600, 562, 583, 632, 587, 598,
633, 595, 600, 642, 589, 617, 645, 597, 646, 582, 606, 579, 647, 627,
604, 623, 613, 586, 601, 588, 615, 620, 618, 570, 587, 593, 645, 617,
620, 610, 599, 607, 599, 603, 619, 643, 600, 644, 618, 563, 627, 613,
634, 580, 622, 608, 590, 576, 609, 592, 613, 566, 618, 602, 634, 574,
557, 555, 611, 621, 640, 611, 648, 664, 590, 601, 632, 560, 638, 635,
602, 598, 595, 597, 600, 526, 660, 602, 645, 591, 583, 609, 597, 564,
570, 527, 552, 616, 586, 683, 632, 604, 615, 624, 586, 641, 626, 590,
627, 635, 614, 628, 578, 649, 605, 637, 627, 585, 598, 618, 616, 600,
609, 649, 626, 613, 643, 560, 600, 597, 635, 574, 579, 594, 614, 589,
604, 564, 620, 571, 671, 616, 616, 617, 597, 603, 604, 598, 624, 653,
586, 623, 589, 661, 628, 663, 604, 621, 576, 617, 607, 615, 614, 609,
590, 608, 651, 593, 580, 574, 650, 598, 586, 610, 595, 650, 621, 583,

```
602, 614, 666, 603, 578, 637, 593, 603, 588, 626, 581, 581, 598, 647,
600, 631, 627, 613, 625, 658, 598, 583, 625, 584, 627, 591, 632, 588,
633, 608, 607, 597, 570, 647, 604, 649, 633, 597, 570, 577, 621, 589,
582, 599, 624, 634, 612, 624, 638, 593, 572, 548, 604, 602, 630, 590,
637, 566, 586, 597, 662, 578, 594, 567, 583, 585, 656, 597, 607, 631,
619, 595, 587, 590, 553, 608, 612, 594, 654, 661, 570, 611, 624, 608,
617, 607, 681, 639, 579, 620, 590, 662, 596, 607, 611, 588, 650, 584,
653, 618, 606, 611, 635, 610, 613, 614, 598, 576, 605, 613, 612, 583,
627, 618, 595, 556, 567, 616, 639, 613, 586, 618, 602, 636, 577, 600,
680, 587, 612, 626, 620, 580, 606, 620, 598, 608, 605, 674, 579, 575,
599, 590, 610, 629, 581, 638, 583, 592, 627, 661, 586, 593, 624, 619,
589, 635, 608, 612, 539, 619, 626, 602, 615, 601, 605, 590, 651, 569,
566, 625, 625, 644, 685, 631, 600, 587, 637, 591, 595, 579, 597, 589,
629, 659, 617, 642, 607, 603, 619, 574]
print("max :", max(lii))
print("min :", min(lii))
print("avg :", sum(lii)/len(lii))

print()

print("Boarding by Random (30% only):")
liss = [395, 390, 382, 394, 416, 402, 387, 413, 401, 425, 396, 393,
379, 417, 386, 382, 393, 370, 400, 387, 376, 364, 411, 400, 394, 375,
407, 402, 400, 422, 389, 386, 380, 411, 393, 374, 367, 400, 390, 402,
403, 432, 428, 393, 390, 388, 407, 395, 397, 412, 397, 363, 404, 402,
400, 430, 362, 376, 384, 359, 399, 365, 388, 400, 379, 428, 395, 428,
376, 388, 438, 390, 423, 398, 421, 410, 429, 385, 363, 412, 387, 400,
379, 399, 374, 400, 403, 391, 382, 373, 390, 397, 384, 362, 400, 395,
379, 394, 416, 369, 389, 384, 393, 382, 385, 381, 418, 408, 375, 431,
423, 413, 399, 406, 387, 381, 382, 423, 420, 376, 392, 379, 401, 429,
391, 378, 374, 405, 375, 430, 399, 419, 399, 417, 381, 378, 427, 379,
410, 412, 394, 386, 418, 390, 397, 397, 401, 394, 383, 386, 404, 416,
378, 393, 421, 425, 406, 392, 405, 373, 393, 410, 385, 362, 386, 425,
403, 410, 381, 394, 397, 397, 380, 382, 379, 389, 374, 384, 405, 381,
389, 369, 413, 394, 445, 378, 399, 412, 405, 384, 404, 424, 386, 437,
411, 392, 413, 369, 394, 420, 417, 406, 391, 389, 412, 365, 420, 394,
391, 419, 346, 411, 391, 377, 415, 409, 413, 410, 388, 387, 399, 385,
385, 403, 420, 390, 423, 384, 364, 401, 409, 415, 429, 386, 378, 403,
435, 367, 410, 438, 369, 391, 386, 357, 424, 369, 384, 387, 396, 358,
416, 390, 356, 416, 394, 384, 394, 385, 413, 374, 350, 368, 349, 412,
397, 399, 367, 385, 384, 426, 423, 390, 398, 427, 400, 394, 408, 389,
385, 368, 398, 393, 384, 394, 398, 390, 348, 381, 406, 421, 388, 408,
388, 412, 416, 385, 381, 387, 374, 412, 376, 368, 391, 411, 421, 381,
395, 356, 403, 409, 381, 394, 371, 365, 391, 411, 366, 398, 364, 410,
```

374, 390, 408, 410, 362, 418, 389, 391, 373, 390, 415, 364, 415, 370,
384, 413, 398, 395, 401, 391, 432, 405, 386, 366, 398, 368, 414, 380,
386, 402, 416, 422, 399, 441, 384, 377, 406, 400, 388, 425, 395, 378,
413, 370, 401, 394, 384, 356, 438, 372, 408, 375, 405, 377, 414, 353,
403, 375, 413, 406, 404, 380, 396, 372, 421, 414, 424, 344, 404, 409,
401, 374, 416, 399, 370, 441, 362, 398, 436, 369, 390, 385, 391, 403,
401, 380, 390, 389, 378, 364, 424, 427, 405, 393, 412, 385, 383, 406,
418, 384, 388, 414, 410, 411, 397, 392, 411, 402, 413, 393, 394, 394,
416, 398, 394, 399, 383, 391, 424, 409, 392, 373, 383, 415, 400, 414,
390, 418, 431, 383, 396, 387, 386, 390, 405, 378, 376, 385, 402, 378,
378, 372, 401, 423, 389, 375, 386, 396, 406, 416, 388, 370, 405, 392,
400, 389, 396, 416, 353, 414, 381, 406, 390, 390, 395, 406, 436, 384,
402, 423, 364, 379, 365, 423, 389, 400, 386, 373, 371, 445, 399, 391,
389, 414, 390, 414, 412, 367, 385, 399, 393, 414, 342, 402, 397, 404,
381, 371, 385, 433, 402, 384, 404, 374, 374, 420, 403, 388, 433, 379,
394, 391, 403, 397, 393, 394, 408, 381, 390, 404, 386, 357, 414, 374,
380, 404, 398, 380, 377, 379, 395, 371, 377, 391, 377, 397, 396, 385,
380, 392, 385, 413, 343, 402, 391, 382, 400, 425, 402, 412, 424, 396,
353, 368, 413, 392, 392, 409, 410, 392, 414, 397, 401, 405, 435, 407,
421, 357, 368, 392, 390, 410, 404, 414, 381, 392, 411, 378, 405, 402,
409, 380, 398, 374, 399, 391, 392, 414, 398, 390, 370, 396, 400, 388,
404, 398, 425, 368, 387, 401, 411, 383, 374, 367, 389, 386, 386, 369,
369, 402, 397, 382, 366, 384, 392, 388, 374, 399, 378, 442, 397, 400,
378, 367, 404, 384, 398, 394, 411, 405, 394, 390, 383, 426, 402, 413,
392, 388, 394, 411, 417, 407, 392, 390, 367, 375, 400, 393, 374, 396,
403, 381, 384, 352, 375, 420, 416, 416, 407, 394, 391, 419, 390, 367,
420, 390, 395, 377, 422, 369, 441, 407, 382, 369, 371, 395, 376, 380,
416, 389, 414, 404, 398, 435, 377, 420, 385, 396, 380, 397, 404, 393,
402, 391, 394, 413, 390, 407, 391, 386, 398, 400, 407, 387, 399, 402,
407, 383, 386, 354, 372, 394, 416, 397, 390, 393, 407, 396, 384, 404,
405, 404, 400, 404, 393, 406, 384, 392, 388, 400, 393, 369, 368, 365,
377, 393, 394, 390, 373, 418, 367, 364, 381, 412, 393, 418, 376, 401,
394, 401, 406, 396, 414, 380, 409, 397, 378, 391, 414, 382, 389, 397,
409, 397, 384, 382, 376, 403, 379, 395, 378, 370, 401, 384, 385, 351,
391, 379, 436, 410, 431, 369, 419, 397, 394, 402, 383, 370, 379, 374,
394, 375, 380, 387, 397, 385, 385, 411, 371, 393, 385, 424, 377, 395,
392, 409, 394, 412, 396, 384, 399, 385, 404, 373, 410, 366, 382, 416,
365, 419, 381, 387, 406, 391, 380, 380, 391, 416, 399, 403, 430, 394,
379, 422, 403, 413, 384, 393, 408, 409, 398, 403, 397, 408, 399, 377,
372, 390, 405, 386, 373, 409, 415, 418, 400, 405, 414, 414, 416, 365,
359, 389, 386, 362, 388, 391, 395, 373, 428, 364, 368, 392, 381, 368,
410, 374, 384, 370, 362, 401, 421, 384, 394, 366, 371, 369, 392, 371,
404, 403, 368, 397, 377, 394, 406, 378, 399, 418, 385, 384, 392, 398,
405, 399, 410, 408, 397, 380, 377, 416, 360, 360, 412, 448, 403, 388,

```
404, 378, 396, 356, 406, 400, 412, 401, 387, 379, 368, 393, 387, 377,
393, 373, 363, 396, 388, 378, 443, 394, 394, 400, 411, 403, 398, 420,
413, 370, 392, 382, 355, 373, 403, 373, 403, 368, 396, 421, 439, 357,
387, 404, 402, 428, 398, 402, 370, 386, 380, 391, 403, 379, 403, 395,
383, 380, 377, 394, 374, 399, 389, 402]
print("max :", max(liss))
print("min :", min(liss))
print("avg :", sum(liss)/len(liss))
```