CMPE 365 Lab 4

Riley Becker

October 2018

Part 1

In this problem, we care about having only all the planes which are at the airport at the same time in a gate, not about maximizing the number of planes that can wait at one gate, like in our scheduling algorithm from class. As a result, when a plane leaves a gate we want to replace it as quickly as possible (call this method 1) instead of waiting for the plane that would optimize the number of planes that could use the gate. This was checked as well for small lists of flights (before testing with the files on onQ), and even with a list of 6 flights the method from class gave a higher number of gates than method 1. This was tested with the arrival times

$$[2, 3, 4, 14.5, 8, 10]$$

and departure times

$$[3, 5, 14, 16, 15, 13].$$

In this case, method 1 gives 3 gates and the method from class gives 4. I also tested a different method (call it method 2), coded in main2() in part1.py. In this method, we iterate through the flights and assign each flight to the first available gate. If all gates that we have when a flight comes in are full, we build a new gate for the flight to arrive at. Using both sets of test data from onQ, this method performs exactly the same as the first method, not only using the same number of gates but also assigning the all the flights to the same gates in the same order. From the two tests, it seems that the methods are equivalent. This makes sense because using either method the lower numbered gates house planes as long as they possibly can and the flights that are left over are allotted to the other gates.

Part 2

The number of gates for 20 different situations, averaged over 1000 iterations of the program, are given in the table below. The columns correspond to a constant maximum lateness, and the rows correspond to a constant proportion of planes which are late.

	15 mins	30 mins	45 mins	1 hr
10%	19.975	20.091	20.126	20.24
20%	19.96	20.121	20.307	20.454
30%	19.945	20.179	20.385	20.634
40%	19.98	20.256	20.489	20.743
60%	19.906	20.303	20.643	20.965
80%	19.899	20.221	20.604	20.88
100%	19.96	20.053	20.46	20.692

Table 1: Test File 1

	15 mins	30 mins	45 mins	1 hr
10%	22.021	22.136	22.216	22.411
20%	22.055	22.245	22.418	22.638
30%	22.047	22.307	22.485	22.739
40%	22.116	22.446	22.689	22.901
60%	22.169	22.45	22.737	22.984
80%	22.171	22.51	22.737	23.008
100%	22.1	22.457	22.672	22.882

Table 2: Test File 2

From the data, it seems that the maximum lateness has an effect on the number of gates needed but the proportion of planes that are late does not have an effect on the number of gates required. This makes sense because the maximum lateness of planes is what effects how much extra overlap there is, while the proportion of planes which run late should only shuffle the flights around among the gates. That being said, a very low probability of being late has a lower number of required gates then a medium or high probability of being late, it's just that it plateaus by a probability of 40%.

Note: In the final code, I ensure when I introduce lateness to the flight times that there is 15 mins between the arrival of each flight and its departure. I did check to see if the average number of gates changed if I allowed the arrival and departure times to be the same, and it stayed roughly the same.