Anthony Galczak
CS481 - HW4 Writeup

Code Design

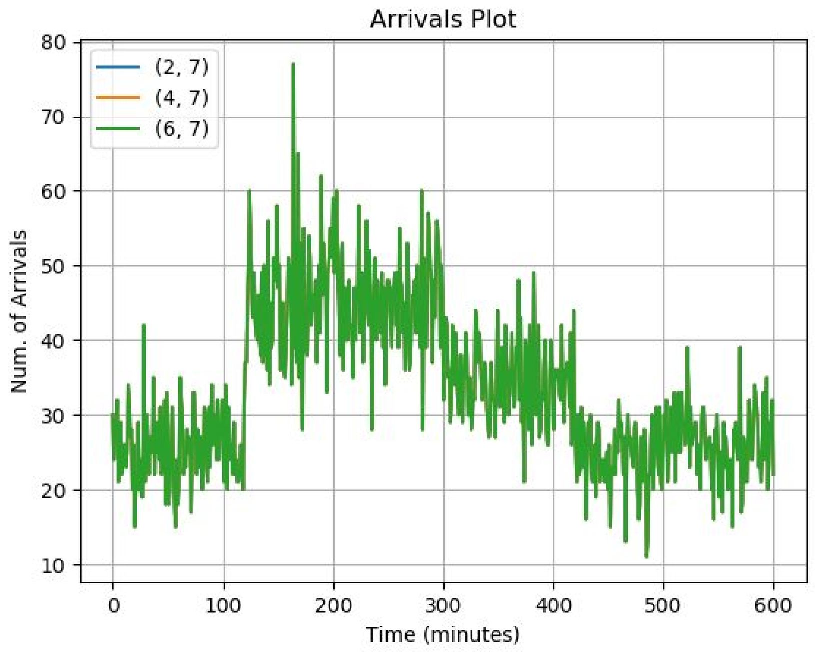
I attempt below to explain my code(sim.c) in plain English, but if you want more details my code is very readable. Variable names are self-describing and the logic is straight forward. I tried my best to avoid any “slick code” or anything that would be unreadable. Additionally, any section that may be a bit complicated has a corresponding comment to explain.

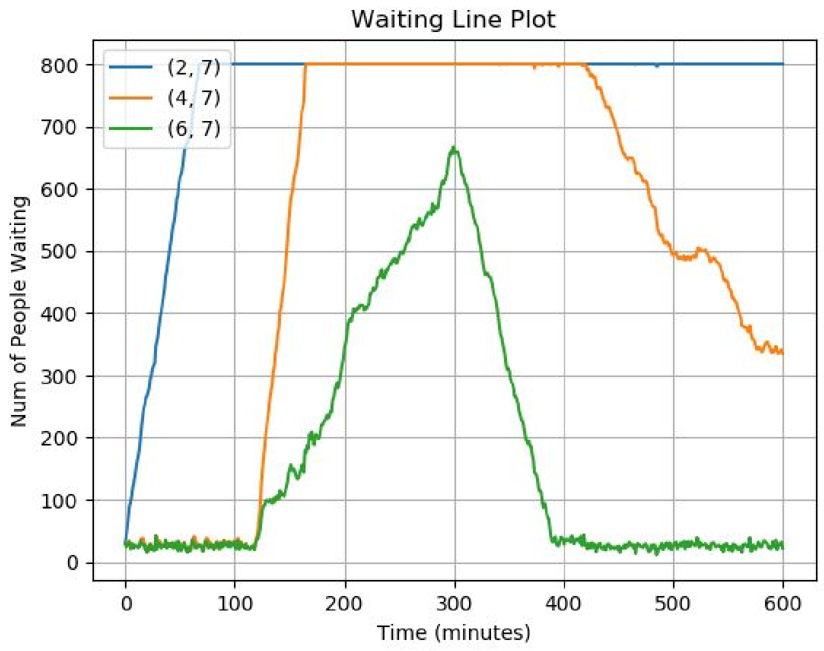
Main thread
First, the code initializes the waiting list to all -1’s, which means the waiting list is empty.
Then we begin to iterate through the time steps of the day, for this program this means iterating over 600 time steps.
At the beginning of each iteration we check the waiting list and if anyone is still in line then we increment their waiting time. i.e. they “waited”.
After this, we generate our arrivals for this time step. This is done via function gen_arrivals().
The main thread now calculates whether or not we can fit the new arrivals into the waiting list. If we can’t, then they are rejected. The remainder gets added to the waiting list.
We now write the iteration status line to an output file called **ride_status.txt**.
Car threads
At this point we are ready to send people on tours in the cars.
I loop over each car and calculate how many to send in the car. When the passengers board is when we calculate directly how much time they waited. Once the car “boards” then we create an individual thread for a car with a struct specifying information to the car thread.
This struct includes the waiting time list and how many people are available to board the car.
The car thread then adjusts the waiting list by shifting it by how many are loaded into the car.
Additionally, if the waiting list was full, we have to fill in -1’s from the right in order to show that these spots are now available.
We continue to load cars (creating a thread for each car) and shift the waiting list until we have looped through all the cars. All cars leave at the beginning of the minute and return at the end of the minute.
Main thread
After all the time steps have been iterated over and all the cars loaded for each iteration, we are now ready to output .csv files containing data for plotting and table generation.
The main thread calls a method output_figure_data() that generates all this data in preparation for main.py to parse it. The end of day report also gets written to **ride_status.txt**.

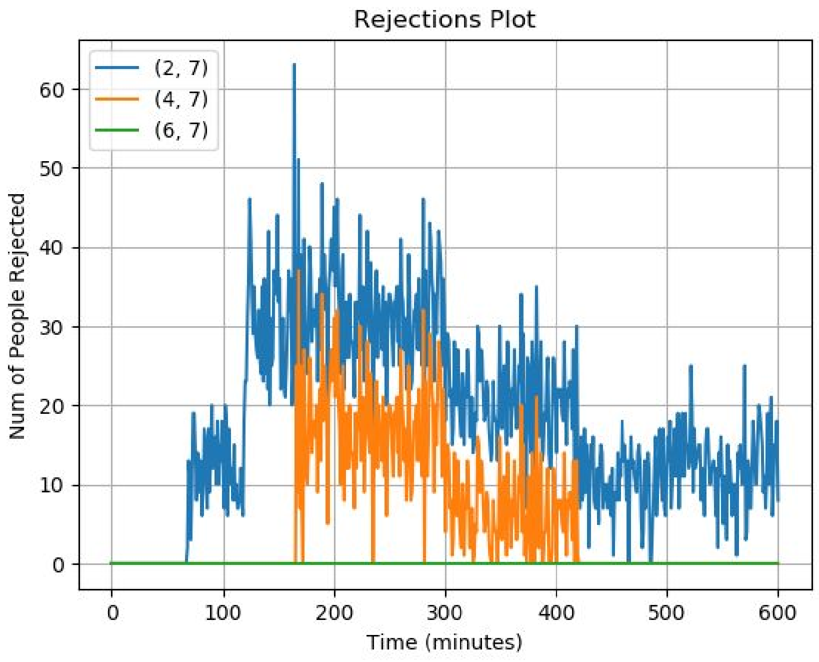
An important aside: There was some ambiguity in the prompt on what happens with the people remaining in the waiting list when the “park closes” at 7PM. I add these people to number of passengers rejected in the total numbers, but I do not include their waiting times in the “average waiting time”. My philosophy is that they waited, but never rode so it shouldn’t affect the average waiting time. The prompt is unspecified on this behavior.

Visualizations

The below plots are auto-generated by an included python script (main.py) that runs sim.c at the selected parameters from the prompt.
For the plots it runs: {N,M} = [(2,7),(4,7),(6,7)]
For the table it runs: {N,M} = [(2,7),(2,9),(4,7),(4,9),(6,7),(6,9)]







(N,M)	Total Arrival	Total Rejections	Rejection Ratio	Average Wait Time	Max Waiting Line	Max Waiting Line Time
N=2,M=7	19999	11599	1.724200	50	800	10:08:00
N=2,M=9	19999	9199	2.174041	38	800	10:39:00
N=4,M=7	19999	3483	5.741889	18	800	11:45:00
N=4,M=9	19999	947	21.118269	10	800	12:20:00
N=6,M=7	19999	0	0.000000	4	667	13:59:00
N=6,M=9	19999	0	0.000000	0	77	11:44:00