Assignment Zero – OOP :

Alon Barak – 213487598

Omer Adar – 325022952

Literature review:

Projects we thought might be helpful for this assignment:

1. Efficient Elevator Algorithm - Tennessee University. https://trace.tennessee.edu/cgi/viewcontent.cgi?article=3380&context=utk_chanhonoproj
2. Smart Elevator – Geeks For Geeks. https://www.geeksforgeeks.org/smart-elevator-pro-geek-cup/
3. Intellevator – IEEE Access. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9003300

The difference between Online algorithms and Offline algorithms:

Most of the time, offline algorithms send the entire data as an input to the algorithm and the method can use the data to optimize and solve the problem better.

On the other hand, online algorithms sends the data piece by piece in a serial order without having the entire data from the beginning.

As a result of not having the whole data from the beginning, online algorithms might make decisions that later on might seem not so smart or optimal. Offline algorithms have the option to make the best decisions according to the given data which make it easier for offline algorithms to reach a better and more efficient solution than online algorithms.

Off-line algorithm for the elevator optimization problem:

Assuming we receive an array of N calls for elevators as an input to the algorithm, U calls up and D calls down (U + D = N).And assuming we have a number of K elevators. Each call have source floor and destination floor.

First, lets divide the array into 2 arrays of UP calls and DOWN calls, lets say u is the length of the UP array and d is the length of DOWN array. We will allocate the elevators for each direction by the ratio of u:d. in case there is only one elevator it will take care of all the calls (up & down). After the division into directions the calls will be divide between the elevators based on the speed of each elevator individually – fast elevators will receive more distant calls than the slower elevator.

On-line algorithm for the elevator optimization problem:

First, we will create an array of linked lists when each linked list representing the road of a specific elevator.

Each linked list is consist of Nodes and each Node will have only two fields: int floor (representing the floor data) and CallForElevator c (represent the call)

Given a call (c) we will create two Nodes :

1. Node for the source of the call (src)
2. Node for the destination of the call (dest)

Up next, we will scan each linked list (elevator) and find the optimistic place to get the new call in the line of our linked list of existing calls.

Then we will calculate the time of each elevator for accomplishing our call (considering the time of existing calls) and save this stat in a variable (Min time).

Now we will find the elevator with the minimum optimistic time for completing the call and embedded the call into the linked list.

Then, the elevator will start its travel through the road using the CMD method.

Ideas for joint tests:

- Testing the allocateAnElevator method by checking whether the method return an integer in range of the number of elevators in the current building.

- Testing the cmdElevator method by asserting that the method actually change the state of the call base on its current state.
- Overall testing to the entire algorithm by creating a call from zero and asserting it reaches it destination floor.

Best results:

- Case 0 : Total waiting time: 236.9897426188186, average waiting time per call: 23.69897426188186, unCompleted calls,0
- Case 1 : Total waiting time: 380.9897426188186, average waiting time per call: 38.09897426188186, unCompleted calls,5,
- Case 2 : Total waiting time: 8135.792822120193, average waiting time per call: 81.35792822120193, unCompleted calls,12
- Case 3 : Total waiting time: 34344.53828433316, average waiting time per call: 85.86134571083291, unCompleted calls,14
- Case 4 : Total waiting time: 51060.455368642084, average waiting time per call: 102.12091073728416, unCompleted calls,13
- Case 5 : Total waiting time: 430231.1211570522, average waiting time per call: 430.2311211570522, unCompleted calls,171
- Case 6 : Total waiting time: 312714.88209694874, average waiting time per call: 312.71488209694877, unCompleted calls,91
- Case 7 : Total waiting time: 890539.1211570507, average waiting time per call: 890.5391211570507, unCompleted calls,449
- Case 8 : Total waiting time: 661971.8820969487, average waiting time per call: 661.9718820969487, unCompleted calls,298
- Case 9 : Total waiting time: 177338.3400743125, average waiting time per call: 177.3383400743125, unCompleted calls,41