בס"ד

Object oriented
Ex1

Yehonatan Barel- 318446333
Nadav Moyal-208514265

how to run the programme:
in our programme we have 3 files

1. AllInOne - that contains all of the classes
2. Ex1 - the algorithm
3. test_Ex1 - test

in order to run the programme you need to put these 3 files in the same folder.

Literature review:
The end of a traffic jam in the elevator ? - this video describing the next generation of elevators,"smart elevator that provides a fast, efficient and cost-effective solution using a smart algorithm that optimizes the operation of the elevator
Unlike before, in this elevator the choice of the target floor is made directly from the user's source place.
The elevator system sends the elevator closest to the user, taking into account parameters such as (elevator status - up / ,down , standby , the distance of the closest elevator to the user),taken from "Mako - Nexter" site.
Mako - Nexter

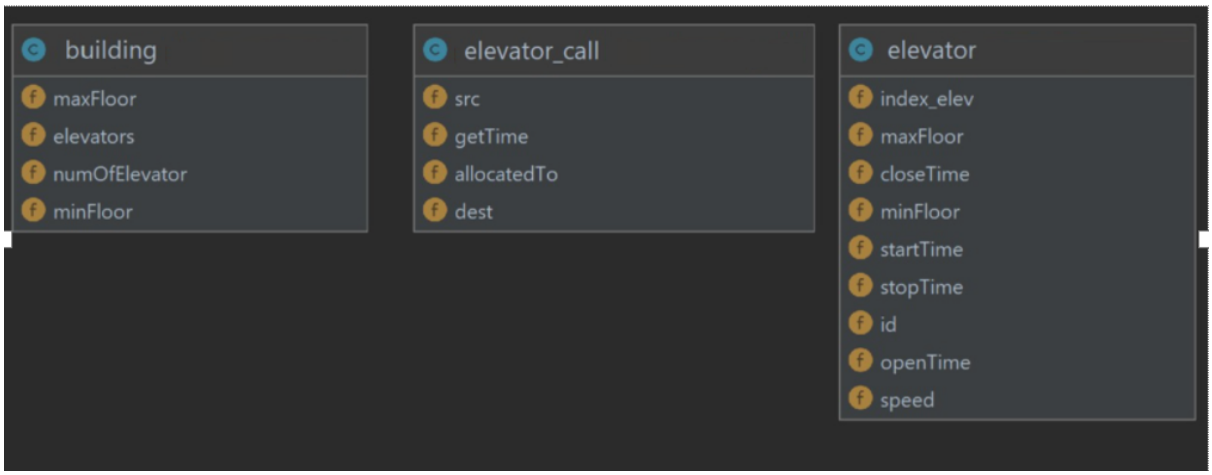
Visualization of the elevator algorithm:
https://www.youtube.com/watch?v=xOayymoIl8U&ab_channel=SpanningTree

A Wikipedia page detailing the possible methods of operation regarding the elevator algorithm:
https://en.wikipedia.org/wiki/Elevator_algorithm

Description of the exercise:
Our mission is to write a code that gets: a json file that represents a building, a csv file that represents elevator calls (without an elevator assignment)
And by an algorithm to export a csv file containing the most efficient assignment of elevators.
This is an offline algorithm, it means that we know all the calls for the elevator from the start.

Explanation of our class:
first of all , we received the information from the json and the csv file by "pandas" to 3 simple class:
1.building
2.elevator
3.elevator call

as we can see clearly at the next diagram:



Explanation of our algorithm:
As we know, we get as input a csv file that represents a calls for an elevator, and we need to return the csv with the best choice of elevators.
Brief explanation
We calculate the time that a certain elevator will work on a given call and we also calculate the time wasted for the scenario in which there is a call but there is no elevator that is currently available to handle it and the handling of that call will be delayed, we check this several times and take the best case.

How it actually works ?
First we check the base case, when we have only one elevator, and if it really is this case we put zero in the all csv. (only one elevator).
We find the next things that will helps up in the algorithm:
1.Find the average speed of the elevators.
2.Find the fastest elevator/s .
3.Find the number of elevators in the building
4.Calculate how many times we will repeat this algorithm,by the number of elevators and the amount of the calls.

Now we are going in a loop on the algorithm x times (we calculated x in section 4 from above), and we are starting a new loop that runs over all the length of the calls file (csv file). We choose which elevator will be assigned for each call .
If there are more than five elevators , check what the speed of the elevator that we choose and check if it is faster than the average speed , if it isn't choose another elevator, if this choice is also not fast enough so choose again.
Now we doing an extra check if we have a call with a huge distance between the src to the dest, so if we have a call with distance that more than a 75% of the building ,than send to this call the fastest elevator/s . (if we have more than one we run over a list that have all the fast elevators and every time it choose an another fast elevator.)

Now we doing a 2 main checks:

1.Floors check: Calculate how many floors each elevator will go in all the calls together.

2.Time check: Calculate how much time is wasted when the elevator is busy with agiven call and late to the next call.

Floors check : We run in a loop on the elevators, and in this loop we start another loop that goes over every call.

In every call we check if the elevator assignment is the same index of the elevator in the outside loop.

If this is it, we sum the floors that he has to reach in the call between src to dest.

and when the insider loop ends , it starts a new loop that sum of the floors that he has to reach in the call between dest to the next src .

In every iteration it will calculate the estimated time of all the floors that every elevator will reach (it considers the speed , stop time, start time , open time , close time).

Time check: We run in a loop on the calls , and in every call we take the "call time", we also take the "call time" of the next call, and the distance of the floors that the elevator will reach.

We calculate the approximate time of each elevator call according to the floors and the features of the elevator (it takes into account the speed, stopping time, start time, opening time, closing time).

so we have 3 thing:

"call time" ,"call time" of the next call , time that takes the elevator to reach the first call.

Now it remains for us to test whether the difference between the first call time and the second call time.

And will the time it takes for the elevator to make the first call be greater than the remainder?

If so, then we have a deviation = delay of the elevator.

$$if\ (|callTime1 - callTime2| < time\ it\ takes\ for\ the\ elevator\ to\ make\ the\ call)$$

So we do that for all the list of calls , and this is one of the main checks that we are doing.

After that in every iteration we sum the time that we get from those 2 checks (Floors check,Time check) ,save it in variable,and insert the list of elevator inlays to an array.

The algorithm returns the minimum time of all the iterations , and bring the list of elevators that makes the minimum time.

Run time table of our algorithm:

First ID (0)	Second ID (1)	Building B1-B5	Call Case	Average Waiting Tim	Number on incompl
208514265	318446333	B1	a	112.92	0
208514265	318446333	B2	a	56.7	0
208514265	318446333	B3	a	31.05	0
208514265	318446333	B4	a	26.29	0
208514265	318446333	B5	a	16.42	0
208514265	318446333	B3	b	547.9726964	176
208514265	318446333	B4	b	240.6366361	43
208514265	318446333	B5	b	62.786	0
208514265	318446333	B3	c	554.774217210004,	122
208514265	318446333	B4	c	251.516474	13
208514265	318446333	B5	c	58.943	0
208514265	318446333	B3	d	581.6006772	77
208514265	318446333	B4	d	241.2957473	20
208514265	318446333	B5	d	59.948	0