Topic 1: Elevator Simulator

Introduction

In a building, there is usually a need for more than one elevator to satisfy the need for carrying people and other people between floors. This creates a challenge in the cooperation and utilization between many elevators.

Therefore, this leads to a scheduling problem. Scheduling the elevators are essential due to the need of minimizing unnecessary stops while dropping off and picking up passengers.

Millions of people use the elevator every day and we may have took the scheduling algorithms behind the elevator for granted.

As we may not know, it is proven by Seckinger and Koehler that the elevator scheduling problem to be a **NP-hard** problem (given 1 elevator and without capacity constraints), due to the large state space of solution and large number of constraints to be factored in, even though it can be reduced to a time dependent travelling salesman problem.

Imagine if there are more than 1 elevators, how complex it would be?

You may read more on Elevator Group Control System (EGCS) about this.

There are some known algorithms that can be used to solve the elevator scheduling problem (for specific use cases) such as the elevator algorithm. Quoted from a comment on ***stackoverflow*** on the elevator algorithm:

The algorithm looks deceivingly simple, yet it is surprisingly very tough to implement, even with a good set of data structures in hand.

Problem Statement

As a sign of appreciation to one of the most important invention over the past 150 years, the elevator, you are tasked to write a program that simulates an elevator controller controlling a **single elevator*** in a building. What building? Your decision!

** Refer to the **Grading Criteria** section*

The elevator installed in the building is **destination dispatch** based, which means when a passenger issues a service request, the passenger will have to **specify the destination**, before the request is being accepted in order to allow room for more efficient scheduling.

Given constants:

- The building has **11 floors** (G - 10)
- Distance between each floor is **3 meter**
- Elevator moves at an average speed of **3m/s**
- The opening and closing of the door uses **5 seconds** each
- Passengers used an average time of **4 seconds** to board/ leave the elevator

Assumptions that you may make:

- At **t=0**, the elevator is **idle** at the **ground floor** with its **doors closed**.
- For every request, there is only **one** passenger
- There are **no limits** for the elevator capacity
- The lifts will **never** break down and suffer down time
- Each passenger is **willing to wait** patiently to be served/ reach their destination
- Free electricity!

Sample Input

For the simulation input, it will consists of the number of requests ***n***, followed by ***n*** lines of request details consisting the request ID, request time, source floor and destination floor.

```
2           # Number of requests
0 0 0 10    # Request ID, request time, source floor, destination floor
1 20 5 10
```

Note: You may modify the input format according to your liking as long as it consists of the above parameters.

Sample Output

You are required produce two outputs, a **log file** consisting of all the actions/ decisions made by the elevator controller, and a **report file** that holds the statistics of the elevator after serving all the requests.

The sample output given are based on a **very very stupid** elevator controller design!

Can you do better? :laughing:

Output 1 - Log File

```
0: Service request (Request ID: 0) received from floor G to floor 10
0: Door opening
5: Door opened
9: 1 passenger(s) entered the elevator
9: Door closing
14: Door closed
14: Heading to floor 10
20: Service request (Request ID: 1) received from floor 5 to floor 10
24: Reached floor 10
24: Door opening
29: Door opened
33: 1 passenger(s) left the elevator
33: Door closing
38: Door closed
38: Heading to floor 5
43: Reached floor 5
43: Door opening
48: Door opened
```

```
52: 1 passenger(s) entered the elevator
52: Door closing
57: Door closed
57: Heading to floor 10
62: Reached floor 10
62: Door opening
67: Door opened
71: 1 passenger(s) left the elevator
71: Door closing
76: Door closed
```

Output 2 - Report File

```
### Elevator Statistics ###
Service request processed: 2
Passengers served: 2
Total floors travelled: 20
Total time taken: 76

## Request ID - 0
Total time utilized: 38

## Request ID - 1
Total time utilized: 56
```

Note: Again, the sample outputs given are based on an **inefficient design**! Surely, you can do better than it, right? :smiley:

Also, you may modify the output format depending on your liking, as long as you have the content listed above!

Before Getting Started

Here are some suggested references that might help you in figuring out this task:

1. https://stackoverflow.com/questions/12009556/datastructure-for-elevator-mechanism
2. http://www.comscigate.com/uml/DeitelUML/Deitel01/Deitel02/ch08.htm
3. http://www.i-programmer.info/programmer-puzzles/203-sharpen-your-coding-skills/4561-sharpen-your-coding-skills-elevator-puzzle.html?start=1
4. https://www.reddit.com/r/IAmA/comments/b1bpp/i*program*elevators*foraliving*ama/?st=jebjeg5w&sh=9ade88f6
5. http://play.elevatorsaga.com/ ***(This game is quite fun!)***
6. http://file.scirp.org/Html/5-8101634_20548.htm

Some Ideas To Explore

- Adding another lift for the building?
- Elevator capacity?
- Electric energy consumption?
- Each request has a maximum waiting time?
- The lift may break down/ suffer down time?
- GUI display/ simulation?
- Fancy report file design?