Elevator algorithm

Name: Ziv Morgan
ID: 209904606
Date: 31/10/2021

Online vs offline algorithms:

The main difference between an online algorithm and an offline algorithm is that with an offline algorithm we know how many stops we will need to do before we dispatch an elevator. This is a major issue when dealing with long elevator rides with fast elevators, on one hand, it would be much faster to take our fastest elevator to the highest floor, instead of taking multiple slow elevators to high floors, but if we have a large amount of elevator calls along the way, the option of taking multiple slower elevators might be better since not all of them would need to stop.

Consider the following scenario, we reacive requests from floors $-10,\ -9,...,0$ to go to floors $90, 91,..., 100$ and we have 10 slow elevators that pass 5 floors per second and 1 fast elevator that can pass 10 floors per second, in addition, asume that opening/closing the doors takes 2 seconds. If we were to use all 10 of the slow elevators it would take us $4 + 20 + 4 = 28$ seconds to complete all 10 requests. But if we were to use the fast elevator? Well it would take us $10*4 + 10 + 10*4 = 90$ seconds to complete the task.
The solution to the problem is pretty simple, when we try to calculate the time it takes to complete a task we should simply add the time it takes to add more people and dispatch the right elevators. But this solution doesn't really help us with an online algorithm since we can't know how many more stops we might need to do along the way to the final stop.

Online algorithm:

The simplest solution to this problem is simply using FIFO, we handle the first request that comes in each time even if we skip a floor with requests. Unfortunantly if we purely use FIFO to handle the requests we might have an issue where two people request an elevator from the lowest floor to the highest and vice versa, then calls that come in from the middle of the building some time later must wait a long time although putting them onto the elevator could lower there waiting time by alot. The solution I propose is a priority queue where each request is sorted in a queue determined by a custom priority function that takes into account the length of the trip and the time the request came in.

Offline algorithm:

Since we could calculate the exactly how much each addition to an elevator route can impact our average waiting times we could make even a not so efficiant algorithm like a basic scan algorithm work optimally as shown in the simulation video and aplication.

Relevent info about elevator algorithms:

https://www.youtube.com/watch?v=xOayymoIl8U
https://www.youtube.com/watch?v=siqiJAJWUVg
https://www.youtube.com/watch?v=TDww3MjL-0A