Ting Ting Ye

Cs323

Prof. Yao

Assignment 2 Write-Up

Physic Experiment:

a. First we have,

   N = amount of steps

   M = amount of student

   m[][] = m amount of student who is doing signed up to do certain step of the experiment

For example, schedule experiment (6, 3, m[][]) will be 6 steps that will be distributed to 3 students depending on what the student has signed up for.

This is a greedy algorithm so there are two cases, one in which the student didn't sign up for the experiment and the other is they did. We want to greedy algorithm to find the most consecutive experiment so we have the least about of students doing the experiment. We also want to be able find a student for all the steps of the experiment to complete.

b. The greedy algorithm that will give the optimal way to schedule experiment is by using the student with the most consecutive amount of experiment steps. because by doing that you can reduce the amount of different students that will work on the whole experiment.

c. Code! Collaborated with Daniel Chang

d. O (m * n) because this greedy code will be checking every single student's sign up table to find the longest consecutive experiment.
e. Proving the ALG is not optimal by contradiction.

Assume there exist some OPT: <S1',S2', … SL'> where L>k for only the purpose of comparing.

Since we already have the Sign Up table all the steps has already been sorted.

Claim: 1<= i<=k is the smallest index where Si does not equal Si' then < S1, S2, … Si-1> and <S1',S2', … Si-1'> are the same.

By design Si are the students with the most consecutive amount of steps. In particular, Si < Si+1

So, < S1, S2, … Si, Si+1', …, SL'> is also an optimal solution (new solution).

Now, if we modify the next index where this new solution differs from our OPT and then the next the same way. We'll get an optimal solution where Si =Si' for all i=1,…,k

ALG : < S1, S2, …, Sk>

OPT: <S1',S2', … Sk+1', …, SL'>

But by design if there is a student who has most consecutive steps, he/she will be the student to do all the steps. There shouldn't be another student who can do more steps than that student. That will be the reason why there is less switches between the students. Therefore, Sk+1' doesn't exist. So, our algorithm gives the optimal solution.

Public Public Transit:

a. The solution to this problem is the same as the dijkstra algorithm. However, it consists of more calculations with the data provided of the frequency and the first train. With the frequency and the first train data, the way to take the train will vary because

depending on how often the train with arrive and when the train starts running the time it takes from point A to point B might increase.

b. Complexity of myShortestTime: $O(V^2)$ because of the two nested for loops.

c. The shortestTime method is an implementation of the Dijkstra's Algorithm

d. I used the existing code to start with everything I needed to for my implementation because myShortestTime is very similar to shortestTime. I had to add in another for-loop to be able to do the calculations taking account for the frequency and the first train running to find the lowest amount of time. Once I have that I can input it into my Time array and lastly I would have the best route.

e. Complexity of shortestTime: O (E + Vlog V)

f. Code!

g. Code!