CS323 Assignment 2

Zihao Shao

1. Experiments Scheduling

(a) Since we are going to schedule steps of the experiment with the least number of students needed, we have the least number of switches. For instance, we have m students and n steps we can split the problem into the lower half steps[1,2,3,...,n/2] and the upper half steps [(n/2+1),...,n].

We find the most optimal students to do the lower half, then call student L and we find the students to do the upper half student U. We can then put the two groups of students together to get the optimal solution for the original structure [1,2,3,...n].

(b)

For each step 1 to n:

For each student 1 to m:

If student sigh up for the step and no need to switch: Put the student into result

(c) /

(d) O(mn)

(e) we call our optimal solution ALG <s1,s2,s3,...,sk>. Then we assume that there is an algorithm more optimal than ALG and prove by contradiction.

Assume there exists some optimal solution OPT <s1,s2,s3,..,sL> where L < K.

Claim: i (1<=i<=k) is the smallest index where si != si then <s1,s2,s3,...si-1> and

<v1,v2,v3,...vi-1> are the same. So by design vi is the least number of students that OPT said is needed to finish the experiment. The finish of vi <= start of vi+1 and finish time of si >= finish of vi.

So, <s1,s2,...,si-1,vi > is also an optimal solution. However, by design of ALG , if there was a solution that could be created with the least amount of switches then it would happen. So OPT algorithm is missing or skipping steps. This contradicts our assumption that there is such an optimal solution that can schedule the experiment with less switches.

2. Public, Public Transit

(a) An algorithm that can be used is Dijkstra's algorithm to find the shortest path. We have to check for some cases like if the station start and end are the same we say 0. Then we check the best route to get to the other stations by checking the weight edges from length[][]. If we make it to the certain station before it arrives we increase the minute to the first arrival of that train and then add the travel length. And If we missed the train we add the time till the next train arrives. Keep doing this method for each station connection till we get to the station we need and return the total minutes needed.

(b) O(n^2)

(c) Dijkstra's shortest path algorithm

(d) I need to store the current shortest paths vertices I used just when I need to check if there is a better path.

(e) "shortestTime" is O(V^2) because worst case it would have to go through all V

vertices to determine the best path. The time complexity of the optimal implementation:

O(E log V), where E is edges and V is vertices