Eric Chan 23308963

Greedy Graphs Writeup

Experiments Scheduling

a) The optimal substructure to schedule the steps with the least switching is to schedule students who can do the most consecutive steps for steps not yet done in order.

b) While there are still steps to schedule.

   Starting at the first step not yet scheduled, find a student who can do that first step and keep track of the most consecutive steps he/she can do including that first step. Keep track of the student and how many consecutive steps of the student who can do the max consecutive steps.

   Schedule the student with the max number of consecutive steps and update the scheduled steps by adding the max consecutive steps to it to find the next step not yet scheduled.

c) Code.

d) m students, n steps. O(m*n).

   At the worst case, none of the students can do more than one step consecutively (n).

   Algorithm iterates through every student to find max consecutive steps (m).

e) Let A = {a1, a2, … , ak} be the solution generated by my algorithm and let O = {o1, o2, … om} be an optimal solution to this problem where "a" and "o" are students. Assume these solutions differ at an ith element which would mean the ith element of the optimal solution contains a student who can do a different number of steps than the ith student of the greedy solution. Replacing {ai,...ak} with {oi,...om} in the greedy solution would yield a solution no better than the greedy solution by itself since for it to be better, oi must be able to do more consecutive steps than ai to eliminate a switch but since the greedy algorithm schedules based on the max consecutive number of steps, oi would be equal to ai. If oi did fewer steps than ai, then there would be one more switch in the optimal solution to schedule the steps not in oi and is therefore not an optimal solution.

Public, Public Transit

a) While shortest path S to T not found. For each edge, find first instance of when the train will arrive after the start time then find how long it would take to get there from source using Dijkstra's shortest path. Save result into a lookup table.

b) Worst case is to find shortest path to every edge, O(V^2). Dijkstra's O(V^2). Total O(V^4)

c) shortestTime is implementing Dijkstra's shortest path algorithm. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

d) When computing first instance after start time, if(start/freq(e) > 0) {start/freq(e) + 1} else{start/freq}. Add result to new table shortest time to some edge e after using Dijkstra's.

e) O(V*V). Two for loops. Outer: 0 to V-1. Inner: 0 to V. (V-1)*V is approximately V^2.

f) Code.

g)