Elijah Desrosier Assignment 2

1. Experiments Scheduling

Given: n number of steps, m number of students that give you a list of steps (sorted) they can participate in. Assume there's a lookup table where you can find if student X signed up for step Y in O(1), so no need to factor that into your runtime.

Find: An optimal way to schedule students to steps such that there is the least amount of switching as possible.

(a) Describe the optimal substructure of this problem

A problem is said to have an optimal substructure if an optimal solution can be constructed from optimal solutions of its subproblems. In this problems case the problem can be constructed by first scheduling the student who has most steps to complete and following him by students who have the next most after him that don't overlap with previous students steps. The subproblem will aim to do most consecutive tasks per student and therefore minimizing the number of switches between students with a total number of steps we can call n.

(b) Describe the greedy algorithm that could find an optimal way to schedule the students

The typical greedy algorithm approach is that you make the locally optimal choice at each stage with the intent of finding a global optimum. In the case of this problem the greedy algorithms approach we'd use is one that minimizes the number of switches between students who are signed up for experiments. The way we'd do this is by scheduling the students with the greatest number of steps and following him by the next student with the next greatest number of steps for the steps that are remaining. By picking the students with greatest number of steps we effectively minimize number of switches required.

(d) What is the runtime complexity of your greedy algorithm? Again, you don't need to factor in the setup of the lookup table, just your scheduling algorithm.

The runtime complexity of the greedy algorithm is O(n^3). With a best case of O(n) when the first student does all of the steps.

(e) (e) In your PDF, based on your answer to part b, give a full proof that your greedy algorithm returns an optimal solution.

1)Prove greedy algorithm gives an optimal solution
By design of greedy algorithm we select student with greatest amount of steps in hopes of globally optimal solution each selection therefore proof by contradiction would be.

Number of switches by greedy algorithm be ALGO: {U1, U2, U3, …, UK}

And number of switches by optimal solution be OPT: {V1,V2,V3,…, VL} where L < K
And the number of students with greatest scheduled jobs are picked first. Let i be the smallest index where Ui != Vi and {U1, U2, …, Vi+1, …, VL} is also a solution and {U1, U2,.. Ui-1} is the same as {V1, V2, …, Vi-1} by design of algorithm Ui is the smallest amount of switches possible.
By using the cut and paste method we see
ALGO: {U1, U2, U3, …, UK}
OPT: {U1, U2, .., Uk-1,…, VL}
By design of our algorithm if there exists a path to switches less then UK it would happen. Contradicting that its possible for a solution to have less switches then the greedy algorithm. Therefore our greedy algorithm will give us an optimal solution.

2. Public, Public Transit

(a) Describe an algorithm solution to this problem. Feel free to talk about how you would adapt an algorithm we covered in class.

The algorithm we would use to solve this problem would be Dijkstra's algorithm. Dijkstra's algorithm is an algorithm that deals with going from a source to a destination and can find the shortest path between two nodes or stations while also accounting for the distance or weights between the destinations. How it works is Dijkstra finds he shortest path by building a set of nodes that have minimum distance from the source node tracking the path of each station stopped at eventually giving you the shortest path. Concerns of the algorithm would be how Dijkstra's cannot properly interact with a graph that provides negative weights and how the algorithm will implement a greedy approach that is fundamental is getting a correct answer. Things to consider would be the wait time for the train at each stop from one station to another, surely this could be tracked and taken account for.

(b) What is the complexity of your proposed solution in (a)?

The time complexity of Dijkstra's algorithm is O(V^2) since it will loop and go through all nodes twice before coming up with shortest path.

(c) See the file FastestRoutePublicTransit.java, the method "shortestTime". Note you can run the file and it'll output the solution from that method. Which algorithm is this implementing?

The algorithm that is implemented in shortestTime is Dijkstra's algorithm.

(d) In the file FastestRoutePublicTransit.java, how would you use the existing code to help you implement your algorithm? The existing code only handles one piece of data per edge, so describe some modifications.

I would use existing code to help implement my algorithm by using functions like FindNextProcess to help calculate the minimum vertex cost.

(e) What's the current complexity of "shortestTime" given V vertices and E edges? How would you make the "shortestTime" implementation faster? Describe any algorithm changes or data structure changes. What's the complexity of the optimal implementation?

The runtime complexity of the optimal implementation is O((|E|+|V|log| V|) by use of Fibonacci Heaps and time complexity of O((|E| + |V|) log |V|) by binary heaps. However the time complexity of our approach is O(V^2) because of costs of searching through vertex's and are simple use of Dijkstra's algorithm.
https://www.quora.com/What-is-the-complexity-of-Dijkstras-algorithm