Kaven Vohra

Professor Yao

Project 2 Writeup

Experiments Scheduling

(a) Describe the optimal substructure of this problem

The optimal substructure can be seen as we try to solve the optimal solution that minimizes the number of switches required to complete all of the steps. We can keep track of which students we will switch to by seeing which students have the most consecutive steps.

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

The greedy Algorithm would consist of finding the student with the most consecutive steps to complete steps of the experiment. When a student cannot complete a certain step we first check if a student who completed previous steps can complete the current step this ensures the least amount of switches required at each step. Students with the most consecutive steps at a particular step are chosen to participate in the experiment.

(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 run time of the algorithm is O(m*n). We are checking each student and for each student we are looking through all of the steps to see if there is a consecutive step (in the worst case).

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

Say we have ALG (our algorithm) and OPT (the optimal algorithm)

ALG returns $x_1, x_2, x_3, \ldots, x_4$ --> 'a' switches

OPT returns $x_1', x_2', x_3', \ldots, x_4'$ --> 'b' switches

Lets say a > b, assuming OPT is better than ALG.

We can use the Cut and Paste method to prove that ALG is the OPT

Say k is the first switch where ALG and OPT differ in number of switches

ALG OPT

$x_1, x_2, x_3, \ldots, x_K$ And $x_1', x_2', x_3', \ldots, x_K'$

Using Cut and Paste:

$$x_1, x_2, x_3, \ldots, x_P, \; x_{P+1}', x_{P+2}', x_{P+3}', \ldots, x_K'$$

By design, we are always taking the students with most consecutive steps to insure the least amount of switches in ALG so, if we were to substitute from 1 → P and P+1' → K', we would still have the same amount of switches. No matter what part we cut and paste ALG will always be OPT.

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.

An algorithm solution to this problem would be using Dijkstra's algorithm. This algorithm starts from a specific vertex (station) and gets the shortest path needed to get from one station to the next. By modifying the algorithm we are able to use the frequency when calculating the time from the starting station to the target station.

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

The Complexity of the algorithm would be O(n^2) since that's the worst case for Dijkstra's algorithm

(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?

Shortest Time implements 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.

The code in shortestTime method was useful in my implementation since I was able to use the Dijkstra's algorithm however, I was able to get the shortest path at every previous station and when calculating the time, use that shortest time in conjunction with the frequency to find the fast way from one station to the target station. The utility of knowing the details of the path was the key to being able to determine the route.

(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?

As it is, the complexity of shortestTime is O (V^2) since it follows Dijkstra's shoertest path algorithm. However, this can be reduced to O(E Log V) with the use of a binary heap in a priority queue implementation. This can be even further reduced to O(E + V Log V) by using a Fibonacci heap. The complexity is a result of a loop which mimics a BFS, which has a runtime of O(V+E) and the binary heap that takes O(Log n).

(g) Code! In the file FastestRoutePublicTransit.java, in the method "myShortestTravelTime", implement the algorithm you described in part (a) using your answers to (d). Don't need to implement the optimal data structure.

(g) Extra credit (15 points): I haven't set up the test cases for "myShortestTravelTime", which takes in 3 matrices. Set up those three matrices (first, freq, length) to make a test case for your myShortestTravelTime method. Make a call to your method from main passing in the test case you set up.