Sarmad Karatela

Assignment 2

CS323

1. **Experiment Scheduling**
2. Optimal Substructure:

An optimal solution of the given subproblem can be obtained by using optimal solution of its subproblems. For this problem, an optimal solution can be obtained by scheduling a student that can do the most consecutive tasks in an increasing order until all the tasks have been taking care of. We can describe this optimal solution as Student(*i)* that shows the largest continuous step, starting at *i* {1…,(i-1)} and let *j* {(j+1),….,k}be the last step that Student(*i)* takes where *i* ≤ *j* ≤ *n*. An optimal solution can be find for those subproblems.

1. Greedy Algorithm:

To find the Greedy Algorithm we will start with step 1, taking the student with longest consecutive steps in an increasing order. This will minimize the number of switching that will take place between the students. But if there are more than one students that has the longest consecutive steps, we will just take the first one and assign that experiment to that student.

1. O(n3)- worst case for finding the student with the most consecutive steps.
2. Proof:

Let’s suppose two algorithms

ALG: <a1, a2…. as>

OPT: <a1, a2…. Ak>

Assume: s > k

Let *ai* represent a student that can perform *n* number of consecutive steps. While *k*  indicates the total number of students, while *s-1* represents the number of switches that occurs after completion of all the steps. Since we know that ALG takes a greater number of switches between the tasks. We need to assume that there exist an optimal solution (OPT) that takes fewer numbers of switches of students than ALG. OPT takes the student with the most consecutive steps. After the *ith* index we have different students performing different number of steps. We can use cut and paste method to get the longest consecutive tasks for rest of the students. This optimal solution claims that this will take the minimum number of switches between the student in order to complete all the steps. Therefore the assumption that ALG exists is contradicted and OPT is the only optimal solution.

1. **Public, Public Transit**
2. A modified Dijkstra’s Algorithm can be used to find the shortest path to the station. Since it’s a single source problem, we can modify the algorithm keep track of the paths that were taken. We need to track the wait time that can occur while transfers. To find the next available train, we need to use first(e) , freq(e) and the starting time to find the next available train on this shortest path. Using the same information, if the train has already passed, we need to figure out how log until the next train arrives.
3. O(n2) since that’s the worst case for Dijkstra’s Algorithm.
4. The algorithm that is being implemented is Dijkstra’s Algorithm.
5. Since the original code only handles one piece of data per edge, we would have to add another array to store the wait time that will keep track of the previous nodes in the shortest path between these two vertices.
6. the complexity of shortestTime is O (V2 ) since it follows Dijkstra’s shortest path algorithm. This can be reduced to O(E log V) by using binary heap in a priority queue. Where E is the number of edges and V is the number of vertices.

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>