

Experiments Scheduling

Problem A)

This problem exhibits optimal substructure because the optimal solution is achieved by scheduling the student with the most uncompleted steps. Then the sub problem of scheduling the student with the most uncompleted which do not include the steps completed by the first student. Then the sub problem of scheduling the student with the most uncompleted which do not include the steps completed by the first student and second student, etc.

Problem B)

The greedy algorithm that can solve this problem is achieved by greedily choosing the student with the most steps that are able to be completed out of the ones that are left. This process will then be repeated until all steps are marked off as completed, resulting in the least amount of students needed.

Problem D)

The runtime of my greedy algorithm is $O(n*m)$ where n is the number of students and m is the number of steps.

Problem E)

Hypothesis:

The algorithm returns S_1, S_2, \dots, S_n with z different students

The optimal solution returns A_1, A_2, \dots, A_n with v different students

With $z > v$

There exists i such that $S_i \neq A_i$ which can occur in only one scenario being that the optimal solution makes a switch but my algorithm does not.

If the algorithm doesn't switch students, it will continue, until S_b where the algorithm finally switches. This would make the algorithm result in: $S_1, S_2, \dots, S_b, A_{b+1}, A_{b+2}, \dots, A_n$. Eventually this would result in the algorithm having at most the same number of switches.

Public Transit Problem

Problem A)

This problem is simply a modified version of Dijkstra's algorithm. Dijkstra's algorithm finds the shortest path between the source node and all other nodes in the graph. However the difference between this problem and Dijkstra's is that we will need to account for the wait time that may occur if a train is not available at the current time because it is not at the station. If the first train hasn't arrived the wait time will be the first arrival time subtracted by the current time. If the train has arrived at least once but isn't available, then the wait time is the frequency of that train subtracted by the $\text{currentTime} - \text{firstArrivalTime} \% \text{frequency}$.

Problem B)

The worst case scenario occurs similarly to Dijkstra's algorithm's worst case which is when the graph provided is a dense graph. The time complexity of this is $O(|V|^2 * \log|V|)$. However in the solution provided above, we also need to factor in the computing the wait time for every transfer which could at worst occur V^2 times. This results in a worst case time complexity of $O(|V|^2 * \log|V| + |V|^2)$.

Problem C)

It's implementing Dijkstra's algorithm.

Problem D)

In order to use the existing code to implement my algorithm, the `shortestTime` method would also need to take into account any additional wait time that may be necessary depending on whether the train is present or not at the current time.

Problem E)

The time complexity of the `shortestTime` algorithm given V vertices and E edges is $O(|V|^2 * \log|V|)$, as stated in Problem B. If the graph used a Fibonacci heap instead of an adjacency matrix, the time complexity can be brought down to $O(|E| * \log|V|)$ which is significantly better.