Nyran Bonilla

1. Experiments Scheduling
   1. The optimal substructure for this problem is to finish all steps in experiment with the least amount of switching. We schedule steps to students that have the longest consecutive steps that they can complete. With whatever steps are left over we look at the other students to do them if they are assigned to it.
   2. The greedy algorithm to this problem is to schedule the student with the most consecutive steps with the remaining steps. As a step is assigned, we find a student with the next maximum consecutive steps assigned. We repeat this until all steps are assigned.
2. O(mn)
3. ALG chose: {S1,S2, S3, … , Sk}

OPT solution: {W1, W2, W3, … , WL}, where L < K

< S1,S2, S3, … , Sk>and < W1, W2, W3, … , WL} are the switches chosen.

If ALG and OPT have the same solution switches up to Si and Wi, 1 < i < L, then we can say <S1, S2, … , Sj , Wj+1 , … , Wi> , where 1<j<i, are the same and won’t make the optimal solution any worse. However, if OPT stops at L while ALG isn’t completed yet, then we can claim that OPT left out some students or steps because our algorithm schedules all students and accounts for all steps by the maximum amount of consecutive steps; Therefore, by using the cut-and-paste logic our algorithm produces the most optimal solution.

1. Fastest Route Public Transit
   1. The algorithm solution for this problem would incorporate Dijkstras algorithm that we learned in class. We would need this to find the shortest path from the source to destination stations. Once given that we can find the shortest time using the frequency table of the frequency the trains arrive at a station and the arrival times table to analyze when the train gets to these stations. There are 2 cases that must be accounted for and that is if you are early for the next train and you must wait and you missed a train.
   2. The worst case scenario of the Dijkstra algorithm is O(V2) to find the shortest path. Once that is found, our algorithms worst case is O(n) since we are looping through each route given and using that to find the shortest time. Since n will never be greater than V, the time complexity (worst case) of the algorithm is O(V2).
   3. Dijsktras algorithm
   4. We can use the shortestTime function to help us given that it’s a Dijkstra implementation. But we can make some modifications by using waiting time to help calculate the shortest time. This can be done by keeping track of the paths and stations visited.
   5. The current time complexity of shortestTimes is O(V2). Given what we learned in class, we can make this function more efficient by using a adjacency lists and binary heap