Experiments Scheduling Questions

a. Optimal Substructure: To optimally schedule the students in a way that involves minimum switching, we choose the student that can complete the most steps out of N steps where N is a constant. After assigning a student to those steps, we look for the next student that can complete the most steps out of the remaining steps, and this process repeats until there are no more steps. Thus, our subproblems are finding which students can complete the most steps out of the remaining steps. After repeating this process until there are no more steps, we will have obtained the optimal solution.

b. A greedy algorithm that can optimally schedule the students will repeatedly check which steps are available to be assigned and it will choose the student that is able to complete the most steps out of the steps that are available. This process will continue until there are no more steps to be assigned to the students. Since the algorithm chooses the student that can do the most steps at every iteration, it minimizes the number of switches needed to complete all steps, providing the optimal solution.

d. $O(n^3)$

e. Let ALG be the solution obtained by our algorithm and let OPT be the optimal solution where

ALG : $< U_1, U_2, \ldots U_K >$

OPT: $< V_1, V_2, V_L >$ where $L < K$

Our algorithm always chooses the student that can complete the most steps at every iteration. Because $L < K$, up to some index i, $U_i = V_I$. After index i either $U_i != V_i$ or V is null. However, this is not possible because the algorithm must choose the student that can complete the most steps at every iteration, if $U_i != V_i$ or V is null then OPT chose the wrong student or it missed a student, thus this solution is not optimal. Therefore, by contradiction our algorithm is correct.

Public, Public Transit

a. Given a start station, my algorithm will look at all the stations that connect to the current station that we're in and will choose the next station to move to by checking which station has the lowest wait time. Wait time is defined as the sum of waiting for the train to arrive + time to travel to the next station. After finding the next station with the lowest wait time, the algorithm will repeat the process again to find the next station until it reaches the destination station. During this process, the algorithm will also check to see if there is a new path found to a station that has a lower total wait time. This ensures that the algorithm always chooses the path with the lowest wait time to a station which gives us the optimal solution of getting from start station to destination station with lowest wait time.

b. $O(V^2)$ where V is number of stations

c. This algorithm is implementing Dijkstra's algorithm

d. The original code compares edge weight to find the optimal path, instead of only considering edge weight my algorithm will calculate the total wait time between all the vertices. Wait time is

defined as the sum of waiting for the train to arrive + time to travel to the next station. This algorithm will work the same as Dijkstra's algorithm but instead of looking at edge weight, it will consider total wait time between vertices instead.

e. The current time complexity of shortestTime is $O(V^2)$, this can be improved to $O(E + V \log V)$ with a Fibonacci heap.