Experiments Scheduling

a) The optimal substructure for this problem would be that either a particular student does a certain number of steps or does no steps at all. Evaluate each student and determine the maximum number of steps they can do. First find the student that has the maximum number of steps and assign them first. Similarly, find the next student that has the greatest number of steps and assign them accordingly as long as their steps don't overlap with the previous student.

b) A greedy algorithm that could find an optimal way to schedule the students would be to see which student has been given most steps and assign them first. Then, consider the rest of the students by seeing which students have steps that overlap but also ensure that the minimum number of switches are being made.

c) Code in PhysicsExperiment.java

d) The runtime complexity of my algorithm is $O(n^2)$.

e) Proof:
ALG $<t_1, t_2, \dots, t_K>$
Assume that there exists some OPT $<s_1, s_2, \dots, s_L>$ where $L > K$
Claim: i is the smallest index where t_i is not equal to s_i, then $<t_1, t_2, \dots, t_{i-1}>$ and $<s_1, s_2, \dots, s_{i-1}>$ are the same unless a switch occurs.
Now, if we modify the next index where this new solution differs from OPT $<s, \dots, s_L>$, and then the next the same way, we'll get an optimal solution where t_i is equal to s_i for all, which therefore concludes that the greedy solution is making the same number of switches as the OPT. This contradicts our assumption that there is such an OPT that can assign students with the least number of switches. Therefore, the greedy algorithm gives an optimal solution.

Public, Public Transit

a) Since the objective of this problem is to get from a single starting to station to another station in the shortest amount of time, an ideal algorithm that could provide a solution would be Dijkstra's Algorithm. Dijkstra's Algorithm involves solving the single source shortest-path problem on a weighted directed graph.

b) The time complexity proposed in (a) is $O(V^2)$.

c) The method shortestTime implements Dijkstra's Algorithm.

d) Using the existing code, I would implement my algorithm by taking into consideration the possible waiting time while traveling from one station to another and attempting to store that information in some type of data structure. Not only would I have to determine the shortest amount of time to travel from Station S to Station T, but I would also have to

consider how long I would have to wait in order to catch the next available train, which would affect the overall minimum travel time.

e) Given V vertices and E edges, the current complexity of the shortestTime method would be $O(V^2)$ since it is based on Dijkstra's Algorithm. One possible approach towards optimizing this implementation would be to modify Dijkstra's Algorithm by using a binary min heap or a Fibonacci Heap, which would result in a runtime of O(V lg V + E).

f) Code in FastestPublicTransit.java

Resources used:
Textbook Lecture Notes on Shortest Path
file:///Users/mchanni97/Downloads/Ch24%20(2).pdf

The Activity Scheduling Algorithm Proof which was completed in class was used as a reference when writing the proof for the Physics Experiment Problem.
file:///Users/mchanni97/Downloads/Interval-Scheduling-Proof%20(1).pdf