Experiment Scheduling

a) Optimal substructure $c[1,n] = \begin{cases} 0, \ if\ A = \emptyset \\ min_{s_k \in S}\{c[1, x-1] + c[x+1, n] + 1\}, if\ A \neq \emptyset \end{cases}$

b) The greedy algorithm to find the optimal way to schedule students is to let the same student help with multiple steps of the experiment before needing to switch to another student. In other words, you want to take the student with the largest sequence of consecutive steps.

c) This algorithm is coded in the PhysicsExperiment.java file

d) The runtime complexity of this algorithm is $O(m * n)$, where m is the length of each student's list of selected steps of the experiment that they offered to help with, and n is the number of steps in the physics experiment.

e) Let $E = \{1, 2, \dots, n\}$ be the set of steps that are part of the physics experiment, $S = \{1, 2, \dots, n\}$ be the set of students that can help to participate in certain steps of the physics experiment and each student in *S* has a sorted list of steps they can help with *e. g.* $S[1] = <1,2,3,5> = \{1,1,1,0,1,0\}$ for n = 6 steps, and *A* = minimum set of students of *S* that help with steps in *E*. This set *A* will be an optimal solution for the problem at hand.
Suppose we have a student *k*, denoted s_k, that is part of *A* that can help with the x^{th} step. By including s_k in an optimal solution, we are left with two subproblems:
   1) Finding students that can solve the 1st to $x - 1^{th}$ steps of the experiment, call this set S_1
   2) Finding students that can solve the $x + 1^{th}$ to n steps of the experiment, call this set S_2

   Let $A_1 = A \cap S_1 \ and \ A_2 = A \cap S_2$, where A_1 is the min set of students for steps 1 to $x - 1$ and A_2 is the min set for steps $x + 1$ to n. We have: $A = A_1 \cup \{s_k\} \cup A_2 \rightarrow |A| = |A_1| + |A_2| + 1$. We can use the cut & paste argument to show that the optimal solution A must include the optimal solutions to the subproblems S_1 and S_2.

   Suppose we find a set A_2' of students in S_2 where $|A_2'| < |A_2|$. We could use A_2' rather than A_2 in the solution to the subproblem for A. We could construct: $|A_1| + |A_2'| + 1 < |A_1| + |A_2| + 1 = |A|$ students to solve steps. This is a contradiction for the assumption that A is an optimal solution. The argument for S_1 is symmetrically similar to the above.

   Let $c[1,n] = \begin{cases} 0, \ if\ A = \emptyset \\ min_{s_k \in S}\{c[1, x-1] + c[x+1, n] + 1\}, if\ A \neq \emptyset \end{cases}$ be the optimal substructure to the problem. The greedy choice is to take the student with the largest sequence of consecutive steps before needing to make the switch to another student. We add this greedy choice to the set A to solve the subproblem.

Public, Public Transit

a) An algorithm solution for this problem would be to use Dijkstra's Shortest Path algorithm to find the shortest path between the nodes and traverse the shortest path to obtain the travel time between the specified start and end nodes. If your arrival time is earlier than the arrival time of the train, subtract the difference and make note of this as the wait time.
b) By using Dijkstra's Shortest Path algorithm, its runtime will be $O(|V|^2)$, where E is the number of edges and V is the number of vertices. The other operations will be of constant time so they do not affect the final runtime of the algorithm.
c) The shortestTime method is implementing Dijkstra's Shortest Path algorithm.
d) You could use the shortestTime method to get the shortest path tree that you could traverse to find the shortest distance between two given nodes. You can keep track ofthe distances traveled by keeping a global variable and adding the length of each edge to that variable to determine the final length of the shortest path between two nodes.
e) The current complexity of shortestPath given V vertices and E edges is $O(|V|^2)$. To make the implementation faster, you could use data structures such as priority queues or binary heaps. The optimal implementation's runtime would be $O(|E| * \log(|V|))$.

Eldrid Gonsalves
CS 323 , Assignment 2 writeup