Karina Ibragimova
Collaborator: Federico Toscano

1. Experiments Scheduling

   (a) Optimal Substructure: The optimal solution is to schedule the student that signs up for the most steps. The sub problem(s) would be to schedule the next student with the most steps that do not include the first students steps and so on.

   (b) Greedy Algorithm: Choose the student with the most steps and repeat until all steps are taken. This leads to minimum students picked and least switches between students steps.

   (c) See PhysicsExperiment.java

   (d) Runtime Complexity of greedy algorithm: O(n steps * m students)

   (e) Proof that greedy algorithm returns optimal solution: Greedy Algorithm gives solution <s(1), s(2), … , s(k)> and m number of students.
   Let's say there exists an optimal solution (OPT) <o(1), o(2), …, o(j)> and h number of students where h < m (resulting in less switches for OPT).
   i is the smallest index where s(i) != o(i). Up to (i-1) both greedy and OPT are the same until there is a switch in OPT. If OPT does not make switches it continues and let's say at s(b) there is a switch:
   <s(1), s(2), …, s(b), o(b+1), o(b+2), …, o(j)>. If we modify the next index and then the next, OPT results in same number of switches as greedy which contradicts that h < m. By design of greedy algorithm, switch only occurs after a student with most possible steps is picked. Therefore, greedy algorithm does return just as an optimal solution as OPT. Greedy Algorithm gives an optimal solution.

2. Public, Public Transit

   (a) Algorithm solution to problem: Dijkstra's Algorithm (which computes the shortest path) in addition to computing the wait time when a train is not available at the current time. If the first train has not arrived, the waiting time results in the first arrival time subtracted by the current time. If the train arrives (at least once) but is not available, then the waiting time results in the frequency of that train subtracted by the current time minus the first arrival time modulo the frequency.

   (b) Time Complexity: Dijkstra's Algorithm time complexity in addition to the wait time so in total it is O(|v|^2 * log|v| + |v|^2).

   (c) Algorithm: The shortestTime method implements Dijkstra's Algorithm.

(d) Implementation: In addition to the existing code, the shortestTime method needs to account for additional waiting time depending on if the train comes or not at the current time.

(e) Complexity of shortestTime: O(|v|^2 * log|v|). To make this time complexity faster though, if we use a Fibonacci heap instead of the given adjacency matrix, then the time complexity would be O(|e| * log|v||v|).

(f) See FastestRoutePublicTransit.java

(g) See Extra Credit in main method of FastestRoutePublicTransit.java