Experiments

a) The optimal solution is the solution with the fewest amount switches in between experiments. Therefore, the optimal substructure would be the student with the most consecutive experiments they can do. Using a greedy algorithm with k total experiments, we choose the student with the most consecutive tasks first. If there are m total tasks and student (i) has the most consecutive tasks gets experiments (1 to m) then the next student will be chosen by the next consecutive tasks from (m+1 to k). This shows optimal substructure.

b) Greedy optimal way would be to schedule students with the most consecutive tasks so in the end there are fewer students with experiments to fulfill

d) Worst case with n number of students and a worst case n times lookup in the signuptable would be O(n^2) time.

e} Suppose our algorithm is ALG and there exists an Optimal algorithm ALG

ALG returns < k1_ ….. k_n> where n = switches

OPT returns <k^1_1 …....k^2_l> where l < n and l = switches

With the assumption OPT is better than ALG

If OPT returns sol with student k^1_1 with the most consecutive number of tasks, we can replace that with k_1 from ALG and it would not worsen the total number of switches. We can use the same cut and paste logic for the next students in ALG and place into OPT. Since ALG uses only next highest amount of consecutive tasks as well, we will end up with the same number of switches. But since OPT claims fewer switches, we reach a contradiction. Therefore our ALG is optimal.

Public Transit

a) An algorithm we used in class would be Dijkstras algorithm. Where we start off with a source node, check the connected nearby nodes to find the shortest path. This would allow us to keep a record of all shortest paths from source to destination but would also have to include wait times from trains based on their frequency of arrivals.

b) Worst case for a Dijkstra algorithm is O(V^2). V is the number of vertices so ours will be the number of stations plus the other constant operations so O(n^2).

c) It is implementing Dijkstras algorithm

d) using a secondary array shortPath I can store the times it takes from the previous stations to the destination station. This will allow for look up of the path taken. Also wait time along with train frequency need to be taken into account.

e) Dijkstras algorithm can be reduced to O(E log V) if a binary heap in priority queue is used. Also can be reduced to O(E + lVog V) in a fibonacci heap where the loop runs O(E + V) and building a binary heap takes O(log n) (https://www.quora.com/What-is-the-complexity-of-Dijkstras-algorithm)