1.

a) Optimal substructure:
at each step (iteration through students' steps), we pick the student that can do the most consecutive steps starting from the current step. We continue to do so until we have completed all steps, thus picking as few students possible to do as many steps possible.

b) We first iterate through the students, and at every iteration we check whether that student can do the current step at hand, if that's the case then we also check how many consecutive steps can they do from there on and then compare it against other students that can also perform that step, at the end of the iteration we pick that student that can do the max amount of steps from current step and then reiterate through students to complete the rest of the steps, thus minimizing the number of switches between students.

c) done.
d) O(n).
e) if we took experiment 1 as an example:
first, our greedy algorithm (ga) will look to see which student can do step 1 out of 6, finds out that student 1 can do it and that it can also do 3 consecutive steps which turns out to be more than any other student (if counting

from 1), then steps 1-3 will be assigned to student 1. Next our ga will look for step 4 to be done and will find that student 2 can do it but with only one consecutive step which is less than student 3 who can do it step 4 as well but with 3 consecutive steps, so our ga will pick student 3 and then all steps 1 through 6 are fulfilled with only 1 switch (from student 1 to student 3), hence optimal solution.

2.

a) An algorithm solution for this problem would be an adaptation of Dijkstra's algorithm. The algo would maintain two sets – one of vertices that are a part of the shortest path tree and one with vertices that aren't. Starting by adding S to the first set, the algo would check against the two relevant matrices and the second set and find the shortest time it would take to get to the next (adjacent) station from S. we update the shortest time it takes to get each station (relaxation). We keep doing that until we reach our target T. then return the time from S to T.

b) O(V^2).

c) Dijkstra's Algorithm.

d)The biggest modification I would have to make is in the way the algo calculates the shortest (or rather fastest in this case) path. The time it takes to get from one station to another, now depends on the time you would have to wait for the next train plus the time it takes it to get you to the next station, and so in determining the fastest path between two stations, we have to take both factors into consideration.

e) Currently the complexity is O(V^2). If we were to use a priority queue with a binary heap implementation, we can reduce the time complexity to O(Vlg V + E lg V).

f)