Physics Experiment:

a).Describe the optimal substructure:

The goal of the problem is to schedule students for each step with a minimum switching. And for each step, my algorithm will first check a student who can do the step and whether the need to switch. Then it will put the student who can do the step and do not need to switch into the result table.

b). Describe the algorithm:

```
For each step 1 to n:
    For each student 1 to m:
        If student sigh up for the step and no need to switch:
            Put the student into result

Return result
```

d). What is the time complexity:

O(mn)

e). Proof:
Let's say S is Switch of students

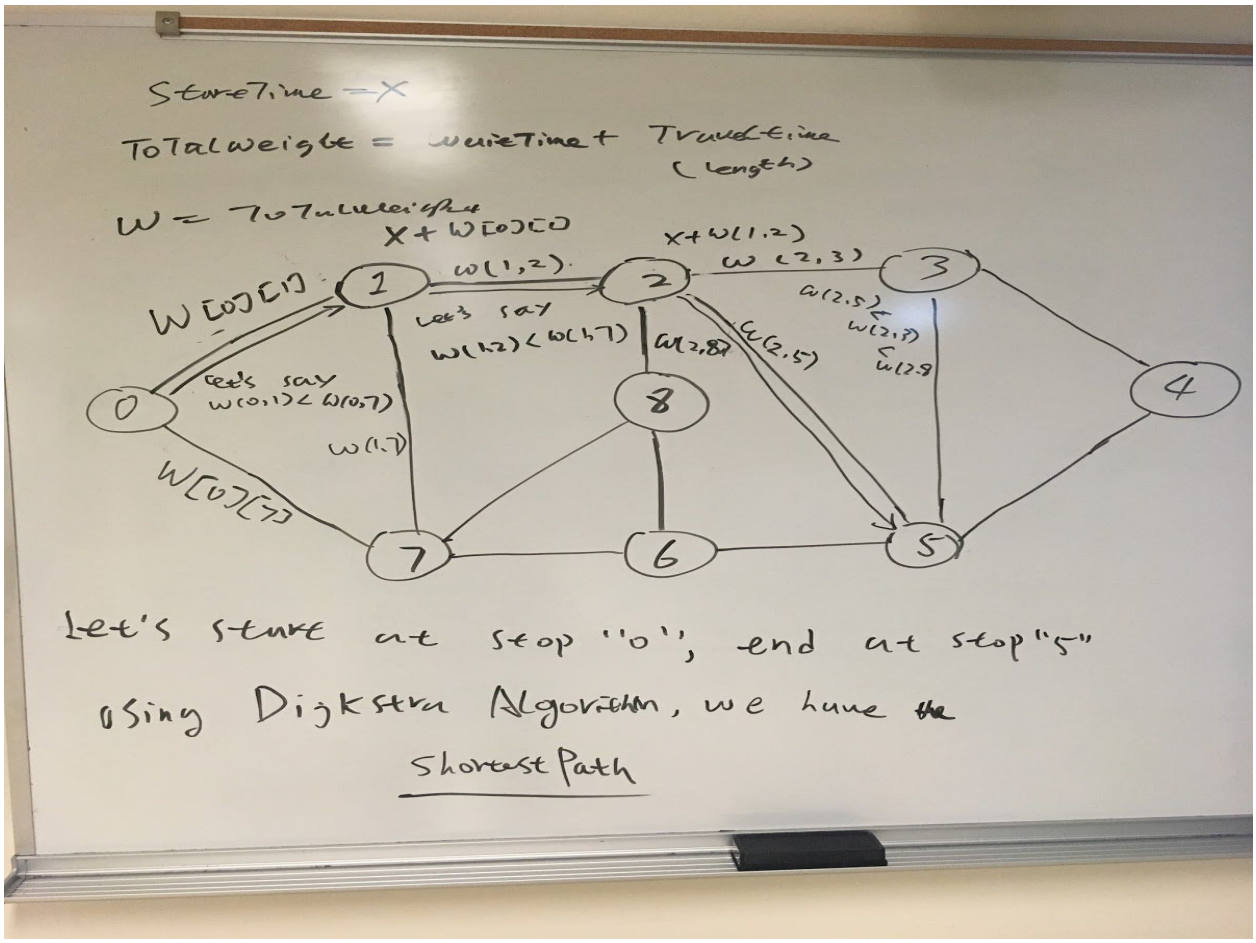
ALG: S1, S2……..Sk,
OPT: s1, s2…….sl, Where l<k, Claim: l<=i<=k, i is the student who did most the steps where Si != si. {S1, S2 ……. Si, si+1……..sl} is also a sulotion. Becasue S1, S2……..Si-1 = s1, s2……..si-1 by design of algorithm, Si is the student who did the most steps, that is compatible with {s1, s2…….si-1} in particular f(Si) <= f(si), {S1, S2 ……. Si, si+1……..sl} is a solution, becasue S(si+1) >= f(si) >= f(Si).
Cut-paste: ALG: S1, S2……..Sk, OPT: S1, S2…….Sk, sk+1……...sl by design of the algorithm, if there exist steps arrangeable with S1…….Sk, it will be arranged. This contradicts that there could be a solution, that arrange more steps that algorithm. Therefore the greedy algorithm gives us an optimal solution.

Public transit:

a). Describe the algorithm:

myShortestPath(startTime, S, T, length[][], freq[][], first[][]):

Using freq[][] and first[][] to get the waitting time,
Then add it to length[][] to get the total weight,
Using the Dijsktra Algorithm to get each optimal subsolution,
After each visted update the start time,
According to Dijsktra Algorithm, we will get the shortest paths



b). What is time complexity:

O(V^2)

c). Algorithm of shortTime() method:

It's a Dijsktra Algorithm.

d). My implementation by using existing code:

The existing method shortTime() is a good tool to implement my algorithm. Because it using Dijsktra Algorithm even though it just handles one piece of data. I just need to adjust it a little bit to make it handle the data appropriately. First I use the freq and first data to calculate the actual time that cost of the edge between two nodes and update the length on each edge between the unvisited node. Then using the Dijkstra to get the cost for each terminates then return the request one.

e). Analysis and optimize the shortestTime():

1. Time complexity is O(v^2)
2. I would use the adjacency list to optimize the algorithm.

3. First using minheap to implement the adjacency list, then store the map data into a adjacency list instead of the graph. And initialize the source as the root of minheap, while minheap is not empty: remove the node with least cost from minheap as the unvisited node, then for each reachable node of it if it meets the condition of Dijkstra then update it.

4. The time complexity of the optimal implementation:
O(E log V), where E is edges and V is vertices

Reference:
https://www.geeksforgeeks.org/dijkstras-algorithm-for-adjacency-list-representation-greedy-algo-8/