Name : Rifa Tasfiya
ID: 20301126 Sec: 03
Course Code : CSE221

Task-3

Here, number of places = N → node/vertex
number of roads = M → edges

For problem-1,

The time complexity for the first "for loop" is O(N) because this loop traverse through all the vertice/places.

Again, as my provided solution to problem-1 is based on normal queue data structure, the time complexity for the "while loop" will be O(N). Now, finding the desired element/places from the queue, we have to traverse through the queue for N times. As a result, the time complexity becomes $O(N^2)$. Then inside the while loop, there is a loop which traverse through the edges/ roads for all the places. Hence, the time complexity for this loop is O(N+M).

∴ The time complexity is $= O(N) + O(N^2) + O(N+M)$
$= O(N^2)$

For problem 2,

The time complexity for traversing through all the places is $O(N)$

From problem 1 (previous part) we have acknowledged that

Time complexity for the while loop is $O(N^2)$

And, inside the while loop, there is a loop which traverses through the edges/roads for all the places. So, ~~the~~ time complexity for this loop is $O(N+M)$

As like problem 1, I used normal queue to implement the Dijkstra function and find the minimum number of titans. In other words, this problem-2 finds the shortest path.

$\therefore$ Time complexity for this problem $= O(N) + O(N^2) + O(M+N)$

$$= O(N^2)$$

Now, if the number of titans in each road is exactly 1, there is O(N+M) algorithm namely BFS to solve this problem.

In this case the input will be,

```
3
1 0
2 1
 1 2 1
 5 6
  3 5 1
   1 2 1
    2 3 1
    1 4 1
     4 3 1
      2 5 1
```

Basically, we do not consider the weight of the graph while implementing BFS algorithm. The BFS algorithm (Breath First Search Algorithm) is applicable for unweight graph or the graph which contains the edges of same weight. The BFS algorithm finds the lowest number of ~~roads~~ nodes / shortest path to reach the destination.

Task-4

In the input table, we can see that there are traffic levels for each Vertex 1 and Vertex 2. Now, if we represent the input table to the graph, we will get a weighted graph (with different weight values).

But we know that BFS is not applicable in such cases. BFS is applicable for unweight graph. Also, we can make the graph having same weight to behave like unweight graph and implement the BFS algorithm to find the shortest path.

Since, the above mentioned criteria does not match for this problem, the BFS algorithm is not applicable for this scenario.