Task-3:

There are N places (nodes) and M roads (edges). In both task 1 & task 2, I have used min-priority queue. So,

for task 1, the time complexity is $O(|V|+|E| \log V)$, where V is N & E represents M. So, it is $O(N + M \log N)$

Similarly, for task 2, the time complexity is $O(N + M \log N)$

If the number of titans in each road is 1, the algorithm to solve the problem with $O(N+M)$ complexity is BFS algorithm.

The inputs can be given as →

```
5   6      (N & M)
3   5
1   2
2   3      Edges.
1   4
4   3
2   5
```

Here, the number of titans (1) is avoided as in every case it is 1.

If we operate BFS algorithm, it will run in $O(N+M)$ and give the output $1 \rightarrow 2 \rightarrow 5$.

Task-4:

Although BFS also gives the shortest path from source to destination, we cannot use BFS in every case. BFS works fine when the weights assigned to every edge of the graph is equal. Because it just iterates through the graph and counts the steps required to reach each node. But when a weighted graph comes into play, BFS is misleading as it does not consider the weights of the edges.

In Dijkstra, we can use a priority queue to find the minimum weight of the edges and put it in our path. So, it takes a little more time. But it provides us with accurate results.

So, as in the given problem, the traffic levels (weights) are given, we cannot use BFS as it will not give us the shortest path considering the traffic level. So, we have to use Dijkstra algorithm to consider the traffic and find the shortest path.