Task 3

I used Dijkstra algorithm in the solution of problem 1 and problem 2. I used adjacency list for creating the graph. We know that the time complexity of Dijkstra algorithm with adjacency list is $O(E \log V)$ where,

E = total number of edges
V = total number of vertices

Now, we are considering there are N places and M roads. N places = N vertex and M ~~place~~ = road = M edge.

In worst case scenario, every vertex will be connected with all other vertex by $(n-1)$ edges.

Now, we used mean hip for priority ~~quee~~ queue which has $O(\log N)$ time complexity for push; pop functions. So it will take $O(\log N)$ time.

There are M roads for which ~~heap~~ priority queue will be used. So, the time complexity will be $O(M \log N)$.

Now, for all the vertices, the total time complexity will be $O(NM\log N)$.

If we consider the tighter bound, we can say,

$O(NM) = O(M)$

So, the time complexity of my algorithm will be $O(M \log N)$

If the number of titan in each road is exactly 1, we can consider this graph as a weightless graph. So we can use a modified version of BFS algorithm which will have a time complexity of $O(N+m)$.

The modification we need to do is while doing we need to store the prvions node of each node. This will allow us to get the path.

So, by using BFS, we can get to our destination with $O(N+M)$ time complexity.

Sample input is.

```
4   5     ← verten   edges
1   2
2   3       edges
2   4
3   4
1   3
1         ← source
4         ← destination
```

Since number of titan is same, we can ignore that input.