Task-3

Dijkstra algorithm pseudocode is in the following:

```
dijkstra(G, s):
    for each v ∈ V[G]:                    O(V)
        do d[v] ← ∞
           p[v] ← None
    d(1) ← d[s] ← 0
    d(1) ← Q ← V[G]
    O(V) ← while Q not empty:
        O log(V) ← do u ← extract_min(Q)
            Uses heap
            data structure
            for each vertex v ∈ Adj[u]:    O(E)
                do if d[v] > d[u] + w(u,v)
                   then d[v] ← d[u] + w[u,v]
                        p[v] ← u
```

works as BFS. So, O(V+E)

∴ Time complexity = $O(V) + O(1) + O(V \log V) + O(V+E)$

$= O(V \log V)$ [By ignoring the lowest terms]

As in task 1 and task 2, dijkstra algorithm has been used. So, for both task 1 and task 2 the time complexity will be $O(V \log V)$.

$= O(N \log N)$

[Here, N = number of places/vertices]

If the number of titans is set to 1, then the graph will as an BFS. Moreover, the time complexity of BFS is $O(V+E)$ which will be $O(N+M)$ according to this problem statement. Thus, we can solve this problem using BFS. The inputs for BFS

algorithm will be graph (list of all vertices) and the start vertex.