Task-3

Part-1

In both task-1 and task-2, I have used Dijkstra algorithm. So, for both task-1 and task-2, the time complexity is same.

Firstly, adjacency list was used for creating a graph and we know the time complexity with adjacency list is $O(E \log V)$

- E → Edges
- V → vertices.

Compute:

Let's assume, N Places = N vertices

M roads = M edges

In the algorithm, we used min heap for priority queue, which runs in $O(\log n)$

time complexity to complete its task.

The total Roads are M.So, the priority will be used M times. The total time complexity for priority queue will be O(M log N).
Ledges ↙ vertices

Again for all the vertices the total time complexity is O(MN log N).

After considering the tight bound, the time complexity become O(M log N), (Am)
↓ edges ↙ vertices

Part 1-2

If we consider the number of trains for every road as 1, the graph becomes a weightless graph. So, we won't be able to use Dijkstra algorithm anymore. The algorithm we will be using for this task is, modified version of BFS, which is also known as shortest path and has a time complexity of $O(N+M)$ (N: vertices, M: edges).

In the modification, all we need to do is to store the previous node of each node. This will get us the required shortest path. Therefore, by using BFS, we can reach our destination with $O(N+M)$ complexity.

Sample input:

1 —— source

5 —— destination

5 6 —— vertex and edges

3 5
1 2
2 3
2 4
4 3
2 5

edges

weight does not needed in the input since the number of titans is 1 for all cases.