Task 3

The best case of ~~my~~ the Dijkstra's algorithm would give O(1) time complexity as it checks in the first line if the adjacency list is empty.

If there are N nodes, initializing dist[], parent[] and explored [] take O(N) time.

We are using a min heap to give the least weighted path. which takes NlogN time. And traversing throughth the ~~M~~ edges ~~nodes~~ takes log(N+M) time.

The while loop deperds on the number of ~~edges~~ nodes which is ~~N~~, so it's time complexity is O(N). Using a min-heap to extract the minimum edge takes log(N) time in the while loop and the second for-loop runs to traverse the edges so its time is O(M)

Overall time complexity is:

$$O(N) + N\log N + O(N+M)$$

O(NlogN) is the dominant figure so the time complexity of this algorithm is O(NlogN)

The same algorithm is used in Task 2 with a minor change. I used a path string and use a while loop to create a string. from parent[ ] and return the reverse string. The time complexity would be O(N). which adds to O(N)+NlogN+O(N+M) However, the final time complexity is still O(NlogN).

BFS algorithm is used if number of Titan is 1. Its time complexity is O(N+M). The inputs will be N and M where N is the number of place and M number of path and followed by M lines of pair values to create the graph.