10_21101298_Anika Islam

TASK 1

For problem 1 and problem 2, I have used Dijkstra Algorithm. The time complexity of Dijkstra is $O(V^2) \rightarrow O(N^2)$ ~~or O(nm)~~ because of the vertices and edges. Since I have used the min-extract function in problem 1 and heapq library in problem 2, these are working as priority queue / min heap. Thus, reducing the time complexity from $O(V^2)$ to $O((V+E)\log V) \rightarrow O((N+M)\log N)$

If the number of Titans in each road is exactly 1 and the time complexity is $O(N+M)$, then ~~we~~ I can use BFS Algorithm. Time complexity of BFS is $O(V+E) \rightarrow O(N+M)$. ~~and we~~

Pseudocode for Dijkstra Algorithm

```
import math
import heapq
Dijkstra(graph, source):
    dist = [0]*(len(graph)+1)
    prev = [0]*(len(graph)+1)
    visited = [False]*(len(graph)+1)
    queue = [ ]
    for i in graph:
        if i != source:
            dist[i] = math.inf
            prev[i] = None
        heapq.heappush(queue, [dist[i], i])
    while queue not empty:
        u = heapq.heappop(queue)[1]
        if visited[u] == False:
            visited[u] = True
            for v in graph[u]:
                val = dist[u] + graph[u][v]
                if val < dist[v]:
                    dist[v] = val
                    if u not in prev:
                        prev[v] = u
                    idx = queue.index([math.inf,
                                       v])
                    if idx in queue:
                        queue.pop(idx)
                    heapq.heappush(queue,
                    [dist[v], v])

    return dist, prev
```