Task 02

Task 1

```
import heapq

def dijkstra(graph, source):

dist = {}

prev = dict()

for v in graph:
    dist[v] = ∞
    prev[v] = None

dist[source] = 0

pq = []

heapq.heappush(pq, (dist[source], source))

while pq:
    u = heapq.heappop(pq)[1]
    if dist[u] == float("inf"):
        break.
```

O(V)

```
for v in graph[u]:
    alt = dist[u] + graph[u][v]
    if alt < dist[v]:
        dist[v] = alt
        prev[v] = u
        push dis[v]
return dist, prev.
```

$O(E \log V)$

Therefore time complexity $= O(M \log N)$

For graph $= O(M+N)$

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

Or, $\text{Max}(O(M+N), O(M \log N))$

Task 2

Same dijkstra function from Task 1 and

```
def solve(graph, source, n)
    dist, prev = dijkstra(graph, source)
    path = [n]
    while n != source:
        n = prev[n]
        path.append(n)
    path.reverse()
    return path.
```

Here the over all time complexity

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

If the number of titans in each road is exactly 1, then the algo to solve the problem will be BFS.