Task-4

BFS:

visited = [0] * noof Places — O(V)

```
def BFS(visited, graph, node, end point):
    Do visit[int(node)-1] → 1 --- O(1)
    Do queue.append(node) -- O(1)
    while queue not empty:
        Do m ← pop(0) — O(1)
        print m -- O(1)
        if m = end point: break -- O(1)
        for each neighbour of m in graph -- O(E)
            if visited[int(neighbour)-1] != 0:
                                    -- O(1)
                pass
            else:
                visited[int(neighbour)-1]=1
                queue.append(neighbour)
                                    } O(1)
```

For BFS it there is v number of vertex then it takes O(v) time complexity to visit each vertex. But after each vertex is visited we explore those vertex's neighbours. for every vertex there can be different numbers of neighbores. If we consider it as E_{adj_v}. then it visits all the edges, then the time complexity becomes $\sum_{i=1}^{v} E_{adj\_v_i}$ = Total number of edges.

which gives us O(E)

Besides, to mark the visited edges we need an array or list which takes O(v) times to create.

∴ Time complexity = O(v) + O(E)

= O(v+E)

DFS:

visited = [0] * no.of places -- O(V)

printed = [] -- O(1)

DFS VISIT (graph , node)

Do visited ([int (node) - [] = 1 -- O(1)

printed . append (node) --- O(1)

For each node in graph [node] --- O(E)

if node not visited --- O(1)

DFS_VISIT (graph , node)

DFS (graph) , end point) --- O(1)

for each node in graph :

if node not visited :

DFS_VISIT (graph , node)

print printed till end point --- O(1)

For DFS, the time complexity, to check if we have visited the vertices $O(v)$ time for v number of vertices while creating a check up array. Where we use queue to visit vertices and explore neighbours in BFS. Here we use stack, which is done by recursion. but it also goes to the child of every parent parent vertices and stops, if a leaf is found. For v number of vertices it visits all of its processors one after. If vertices v the number of predecessors is E_{pre_v} then the time complexity is $\sum_{i=1}^{v} E_{pre\_vi}$ which

is the total number of predecessor for all vertices. It is same as total number of edges in BFS. which gives us $O(E)$

$\therefore$ time complexity $= O(V) + O(E)$
$= O(V+E)$

For matrix this value will differ but both of them will have same time complexity. the time complexity will be $O(V^2)$

In particular case the DFS is the shortest path. It takes less city to reach destination.

But the time complexity might be faster in BFS as the program ends exactly when we find end point unlike in DFS where we go through all.