CSE 221 2010537

TASK - 05

(i) For BFS :

```
visited = [0] * nodes
queue = [ ]
BFS (visited, graph, node, end_point):
    visited [int (node)-1] = 1
    queue. append (node)
    while queue not empty → n or V
        vw = pop()
        print (vw)
        if vw = end-point:
            Break
        For each neighbour of vw in graph →
        if visited [int(neighbour)-1] = 0
            Do visited [int [neighbour)-1] ← 1
            Do queue ← append (neighbour)
```

For Matrix
n / V
n^2 / V 2

For List
outdegree. V
Total = V + E

BFS

∴ Time complexity for Matrix = V x V = V^2

BFS Adj

Time complexity for List = V + E

20101537

(ii) For DFS.

```
visited = [0] * V
stack = [ ]
def visit(start)  ⟶ O(E)
    visited[int(start)-1] = 1
    stack.append(start)
    for node in graph[start]:
        if int(node) not in visited:
            visit(node)

def ~~finish~~ DFS(finish)
    for node in graph ⟶ O(V)
        if node not in visited:
            visit(node).
```

∴ Time complexity for, DFS Adj list = $O(E)+O(V)$

$= O(V+E)$

Time complexity for DFS Matrix = V^2

(#) in DFS we will get to the victory road faster. Because in BFS when we visit a node we visit all its adjacency nodes immediately after that. For example, in BFS Ash will first visit cerulian city then he will visit both lavender town and viridian city. But in DFS he will only visit one of them to get to victory road faster.