Answer to the Question no: 5

For task 01 there is a for loop that has a time complexity of O(v) which also aplices for the task 2 and task 3.

task 02:

```
DFS (visited, graph, node, endpoint)
    Do visied [node-1] => 1
    Do queue. append (node)
    while queue:
        Do m <- pop()
        print m
        if m = endpoint break
        For neighbour in graph [m]
            if visited [int(neighbour)-1] <=> 0
                Do visite [neighbour-1] <- 1
                Do queue. append (neighbour)
```

Here The while loop will triverse thrugh the nodes O(V) times and for loop will loop throughout nodes taking O(E) times.

Therefore the time complexity for each and every node and edges triversed will be $O(V^V)$

Task 03:

```
DFS_Visit(graph, node):

    Do visited [int(node)-1] ← 1
    printed. append (node)

    For each node in graph [node]

        If node not visited
            DFS_Visit(graph, node)

DFS (graph, endpoint)

    For each node in graph

        If node visited
            DFS_visit (graph, node)
```

DFS_Visit Function will traverse through edges taking O(E) times. For adjecency matrix it will run for all vertecies and edges taking O(V) times. Therefore the time complexity will be O(V+E) or O(V)

Here we can see the time complexity of both adjacency list matrix is same for each task. Hence in our case the endpoint at the end of the map. We know BFS is a parent-priority based algorithm and DFS travel less places than BFS Therefore Gray will reach before me by appling DFS.