1

Name : Rifa Tasfiya
ID: 20301126 Sec: 3
Course Code: CSE221

Task-5

Time Complexity of BFS (Task-2):

If the graph is represented as adjacency list and there are V number of nodes/vertex and E number of edges in the graph,

→ We can discover all neighbors of each node by traversing its adjacency list just once in linear time. (Each neighboring vertex is inserted once into a queue and also each visited vertex is marked so it can not be visited again).

→ So, the sum of the sizes of adjacency lists of all the nodes/vertex is E in case of directed graph (As each vertex maintains a list of all its adjacent edges).

Therefore, time complexity in case of adjanceency list $= O(V)+O(E)$
$= O(V+E)$

Again, if the graph is represented as an adjacency matrix (Basically, a VxV array),

→ In order to discover all the outgoing edges of each node, we have to traverse an entire row of length V in the matrix. Point to be noted, each row of the adjacency matrix corresponds to a node in the graph. That row includes ~~includes~~ information about edges that emerge from the vertex.

Therefore, time complexity in case of adjancency matrix $= O(V\times V)$
$= O(V^2)$

2

Time complexity of DFS (Task-3):

Let; V = number of vertex and E = number of edges in the graph.

If the graph is represented as adjacency list,

→ Like BFS, we can discover all its neighbors by scaning its adjacency list just once in linear time.

→ So, in case of directed graph, sum of the sizes of adjacent list is E like BFS.

Therefore, the time complexity in this case = $O(E+V)$

Again, if the graph is represented as adjancency matrix $(V \times V)$,

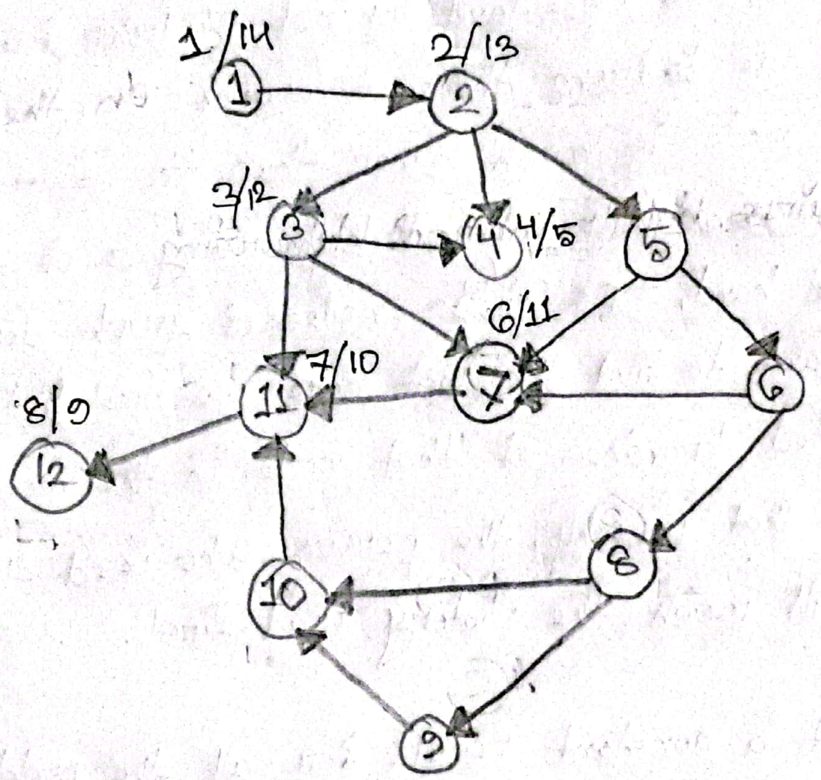
→ Like BFS, each row stores information about edges emerging from the node and to discover all its outgoing edges we have to scan an entire row of length V.

So, time complexity of the ~~B~~DFS in this case = $O(V^2)$.

$\boxed{\text{P.T.O}}$

③

Gary (who used the DFS algorithm) gets to the victory road first because he requires less number of places to reach the victory road.



Performing BFS:

Queue	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~7~~	~~11~~	~~6~~	~~12~~	8
Dequeue	1	2	3	4	5	7	11	6	12	
Steps :	1	2	3	4	5	6	7	8	9	

Performing DFS:

	1	2	3	4	7	11	12
Steps :	1	2	3	4	5	6	7

4

Here, when I used the BFS algorithm, I had to visit 9 cities. On the other hand, Garry (who used DFS algorithm) had to visit 7 cities.

We know, BFS visits nodes level by level. It looks for nodes based on their distance ~~to~~ from the root/source. So, if the level of distination node is much higher than the source node, then the BFS algorithm will requiired high number of traversing. On the other hand, DFS visits nodes of graph depth wise. It always want to find the last node we visited to go in the other unvisited branches of that node in the most efficient way possible. That is why, the person who used the DFS algorithm will reach the Victory road first.

But it is not a constant result for all the problems. It depends on the graph's level and depth. For this secenerio, the person who used the DFS algorithm will reach the victory road first.