Lab-3

Task-4

BFS time complexity:

for adjacency list

Adjacency is a map of keys, where every ~~matrix~~ vertex is a key and points to a list of vertices which are incident from or adjacent to that key vertex.

In order to perform BFS, you have to put any vertex in queue and make it as visited, pop the queue[0], pick the

Starting vertex, explore all its adjacent vertices, make them as visited and put them all in the queue and ~~similarly pop the queue~~ similarly pop the queue[0] and explore all the non-visited vertices until the queue becomes empty. For every vertex, you are traversing through only its adjacent non-visited vertices (nothing but edges)

~~SO, the time complexity~~

visited = [ ] , queue [ ] —— O(1)

BFS(- - -) :

- - - - } O(1)

while queue not empty :

ans = queue.pop(0) —— O(1)

print(an)

output = + - -

if an = endpoint break

} $O(c_1)$ for vertices

for each i of an in graph :

if i not in visited

} $O(c_2)$ for edges

Therefor, the time complexity,

~~$O(1) + O(c_1V + c_2E)$~~

$O(1) + O(c_1V + c_2E)$

$= O(V+E)$

V → vertices, E → edges.

For matrix:

In matrix representation, for every vertex, you have to traverse through all the vertices and check whether there is a non-visited vertex.

since, for every vertex, we are traversing through all vertices,

the time complexity is,

$$O(V^2)$$

DFS time complexity:

Adjacency list:

In worse case, DFS and BFS will give same the complexity in terms of adjacency list.

Since its a directed graph its ~~vertices~~ edges will be

called once and its all vertices will traverse through.

therefore, the time complexity is will be O(V+E).

for matrix:

- An adjacency matrix is a VXV matrix.

In worst case, the algorithm will have to traverse through entire VXV matrix.

∴ the time complexity is,

$$O(v \times v) = O(v^2)$$

Granny will get the victory road first. Because, ash have to go through 7 different cities between source to destination city. Or the other hand, granny have to go through 5 different cities between source

to destination city,
Although BFS known as shortest path, in this case DFS's depth search luckily found the destination city shorted. That is why Guru gets to the virtual road first.

Ash:

1 → 2 → 3 → 4 → 5 → 7 → 11 → 6 → 12

source

Guru:

1 → 2 → 3 → 4 → 7 → 11 → 12

source destination