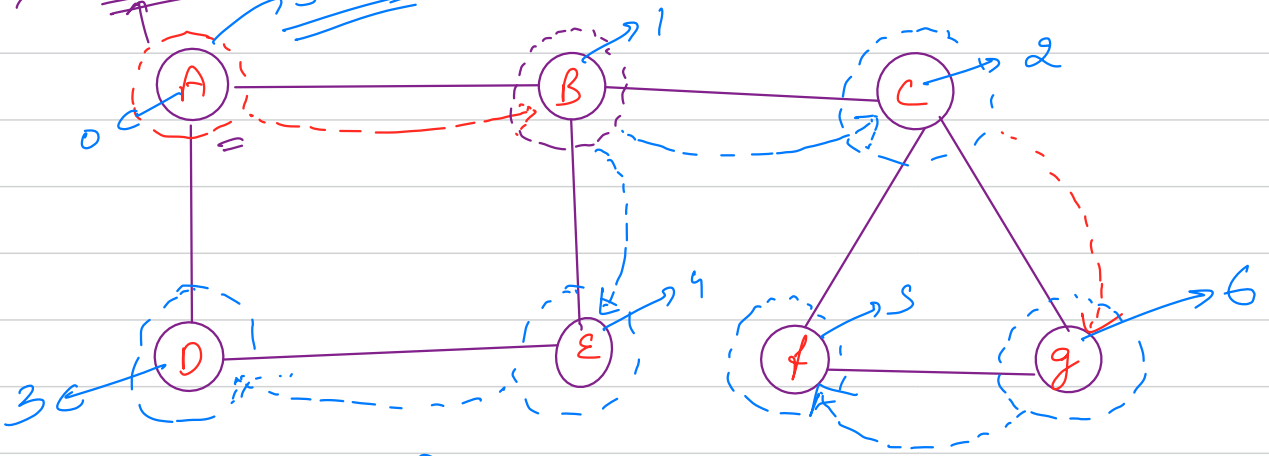
⇒ Graph Traversals → These are algorithms, using which we can read / traverse a graph.

↳ Depth First Traversal
↳ Breadth first Traversal

Depth first traversal- In a depth first traversal, we pick any node as a source & recursively move to any one neighbour, & explore the depth first traversal from that neighbour before moving to next neighbour.

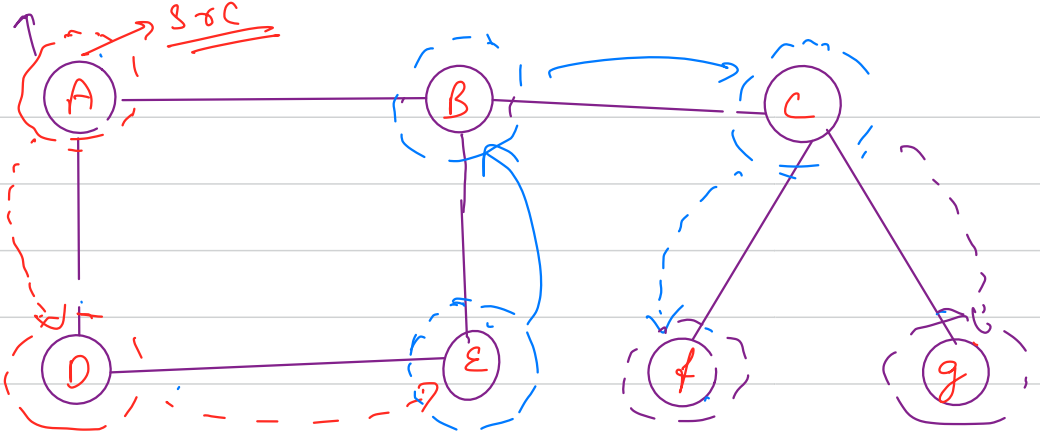
visited

source node → we will start the traversal from node A



A, B, C, G, F, E, D

DFT



A D E B C f G ≈ DFT

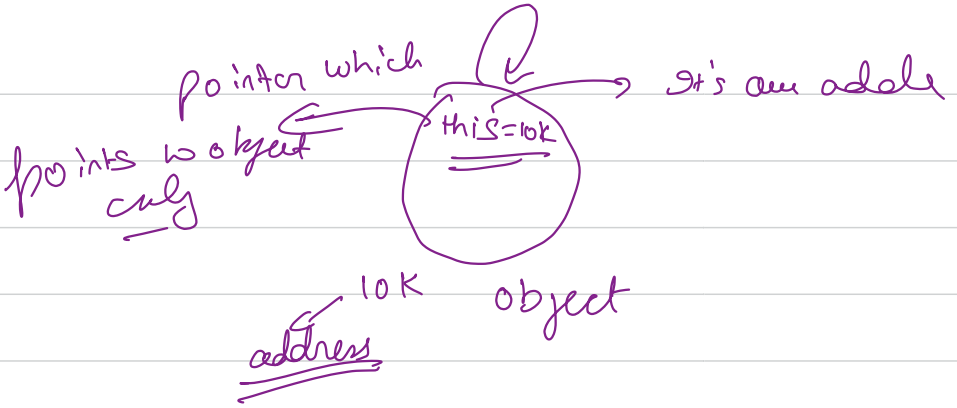
Time Complexity → $O(V+E)$

Space Complexity → $O(m)$

recursion's call stack

m → length of the largest path btw 2 nodes in a graph

visited → unordered_set


pointer which
points to object
only
this=10K
it's own adds
10K
address
object

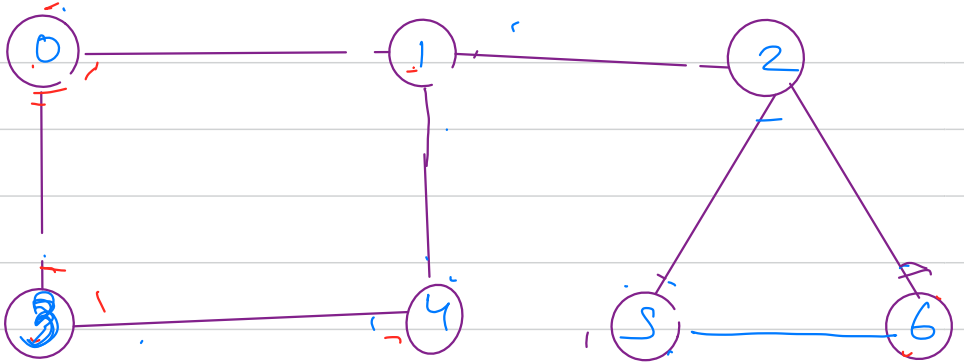
```
    void dfsHelper(int src, unordered_set<int> &visited) {
        // mark the current node as visited
        visited.insert(src);
        cout<<src<<" ";
        for(int neighbour : adj[src]) {
            if(visited.count(neighbour) == 0) {
                dfsHelper(neighbour, visited);
            }
        }
    }

    void DFS() {
        unordered_set<int> visited;
        dfsHelper(0, visited);
    }
};
```

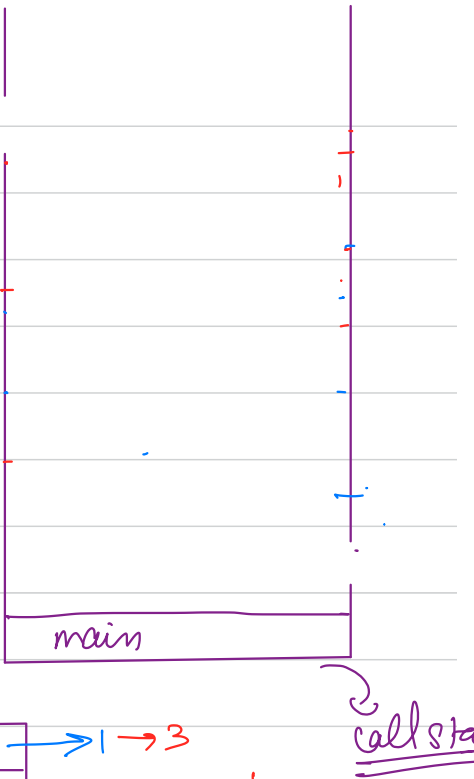
pass by reference

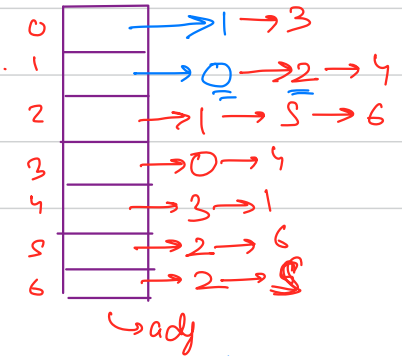
Console

0 1 2 5 6 4 3

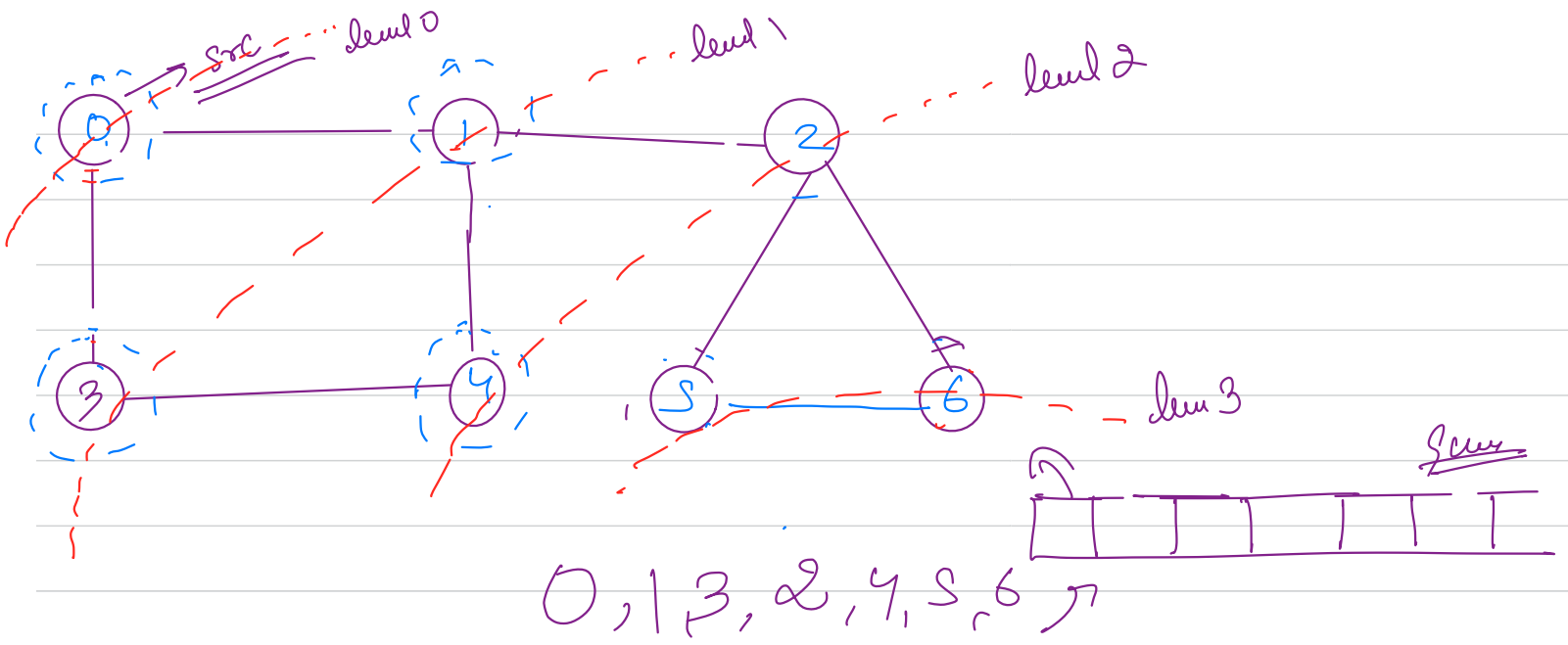
{0, 1, 2, 5, 6, 4, 3} visited







Breadth first Traversal → from any node travel all the immediate neighbours first & then pick any neighbour & do the same.

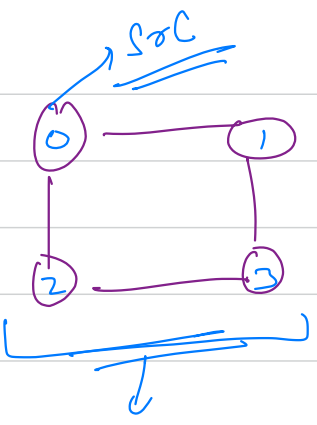


whosoever comes first gets processed first → FCFS

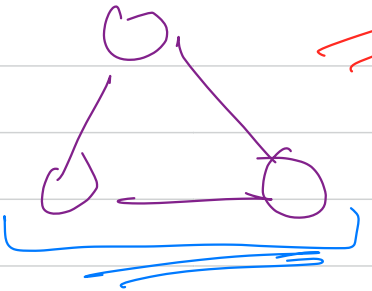
Queue

DFS

BFS



connected component



another connected component

2

Q: Given a graph, calculate the no. of connected components.

Sanket.Singh--