1 → Red

0 → yellow

2 → white

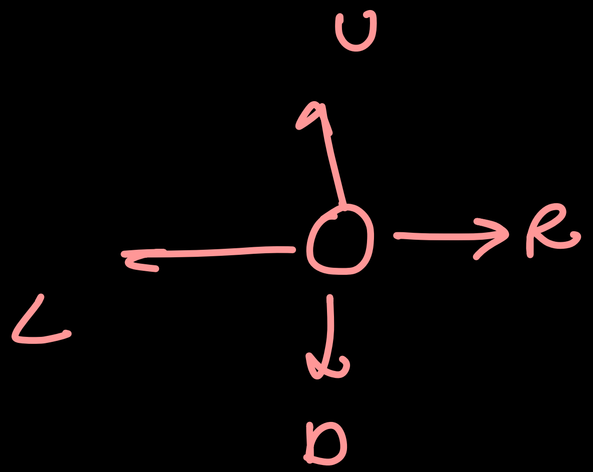
grid based

	0	1	2
0	~~1~~ 2	~~1~~ 2	~~1~~ 2
1	~~1~~ 2	~~1~~ 2	0
2	2 ~~1~~	0	1

Sr = 1
↳ source row

Sc = 1
↳ source col

color = 2

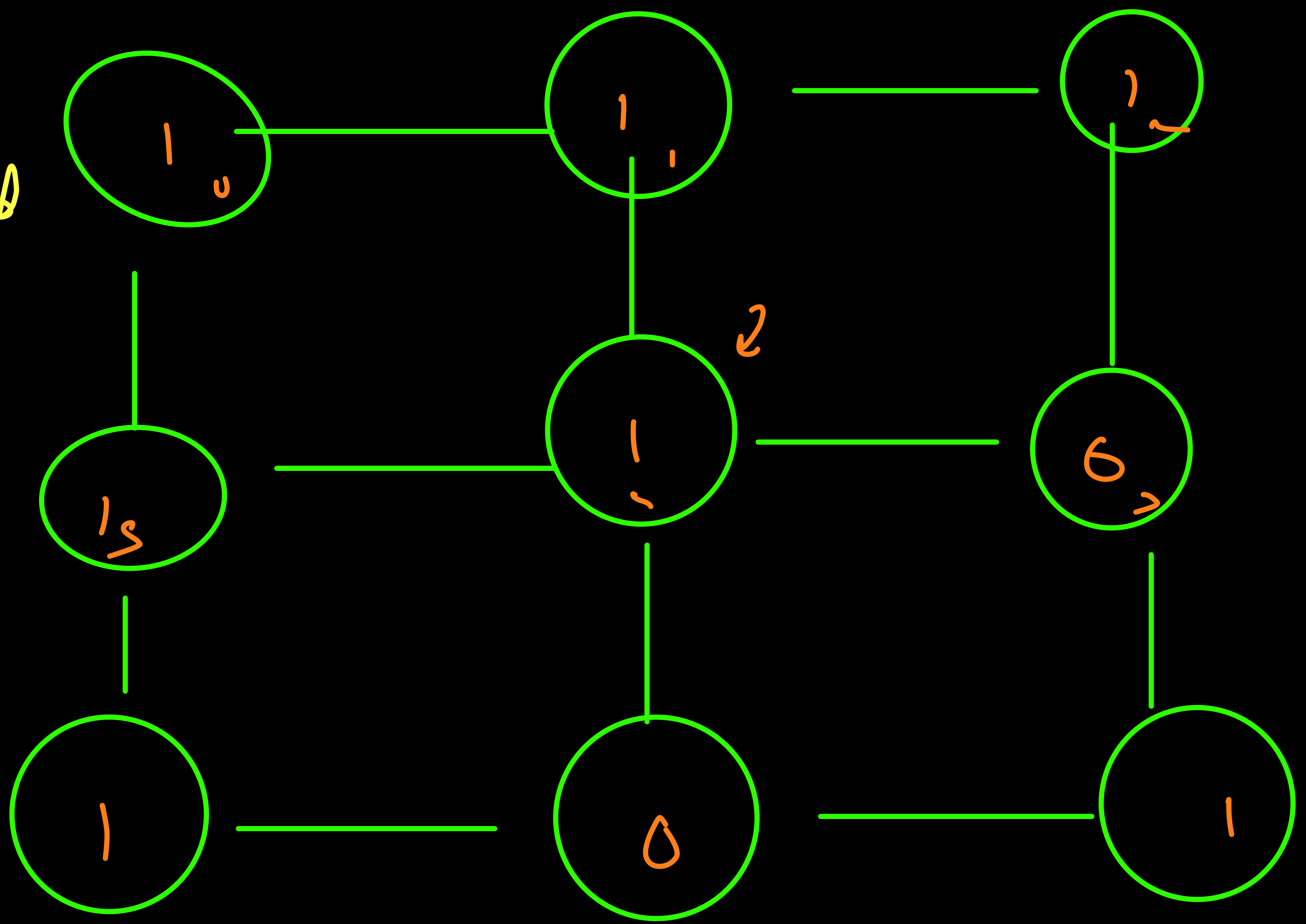


Problem is giving an intuition of recursion

↳ DFS

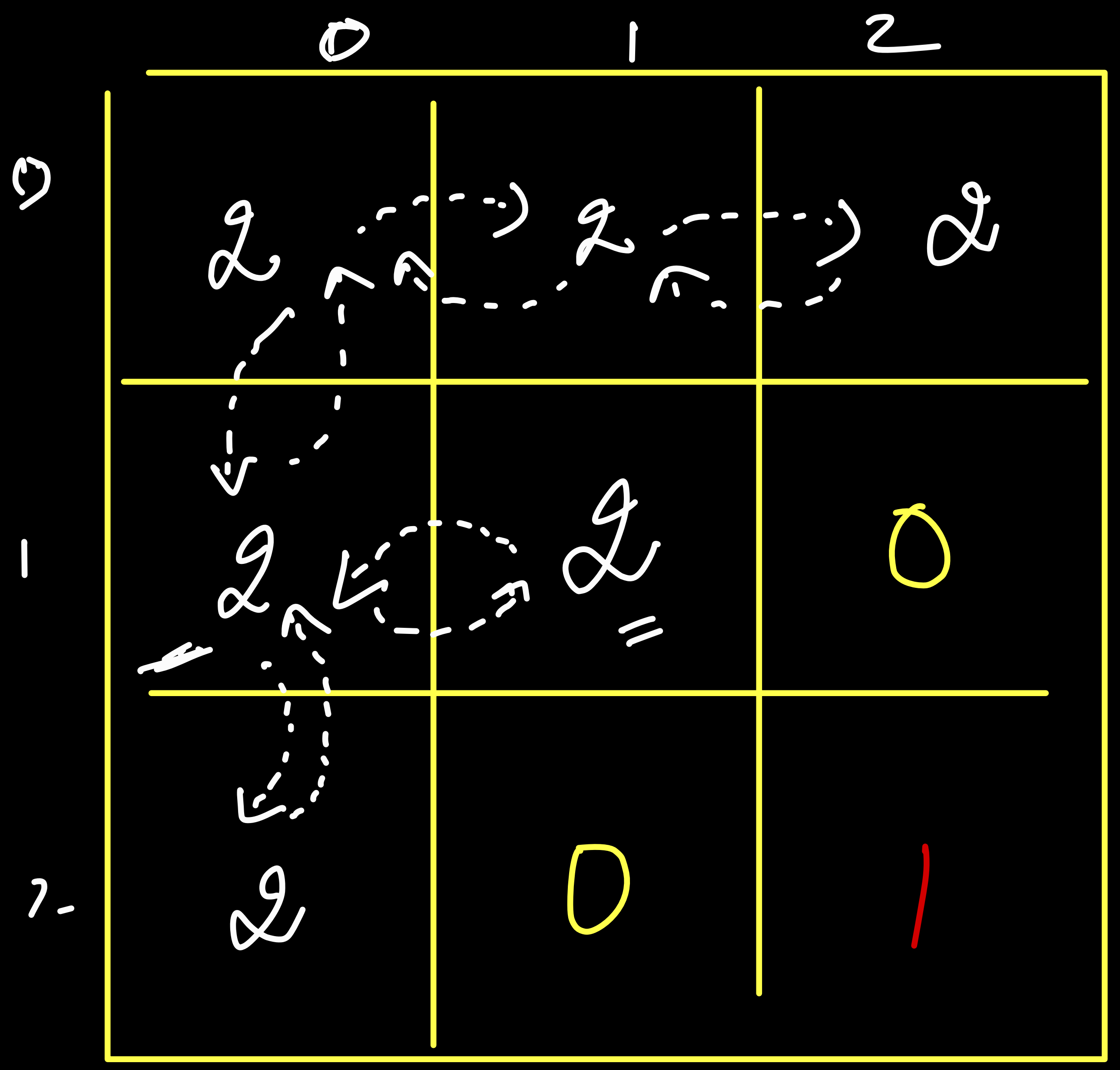
↳ BFS

most of the grid based dfs/bfs can be solved without creating a graph of the grid.



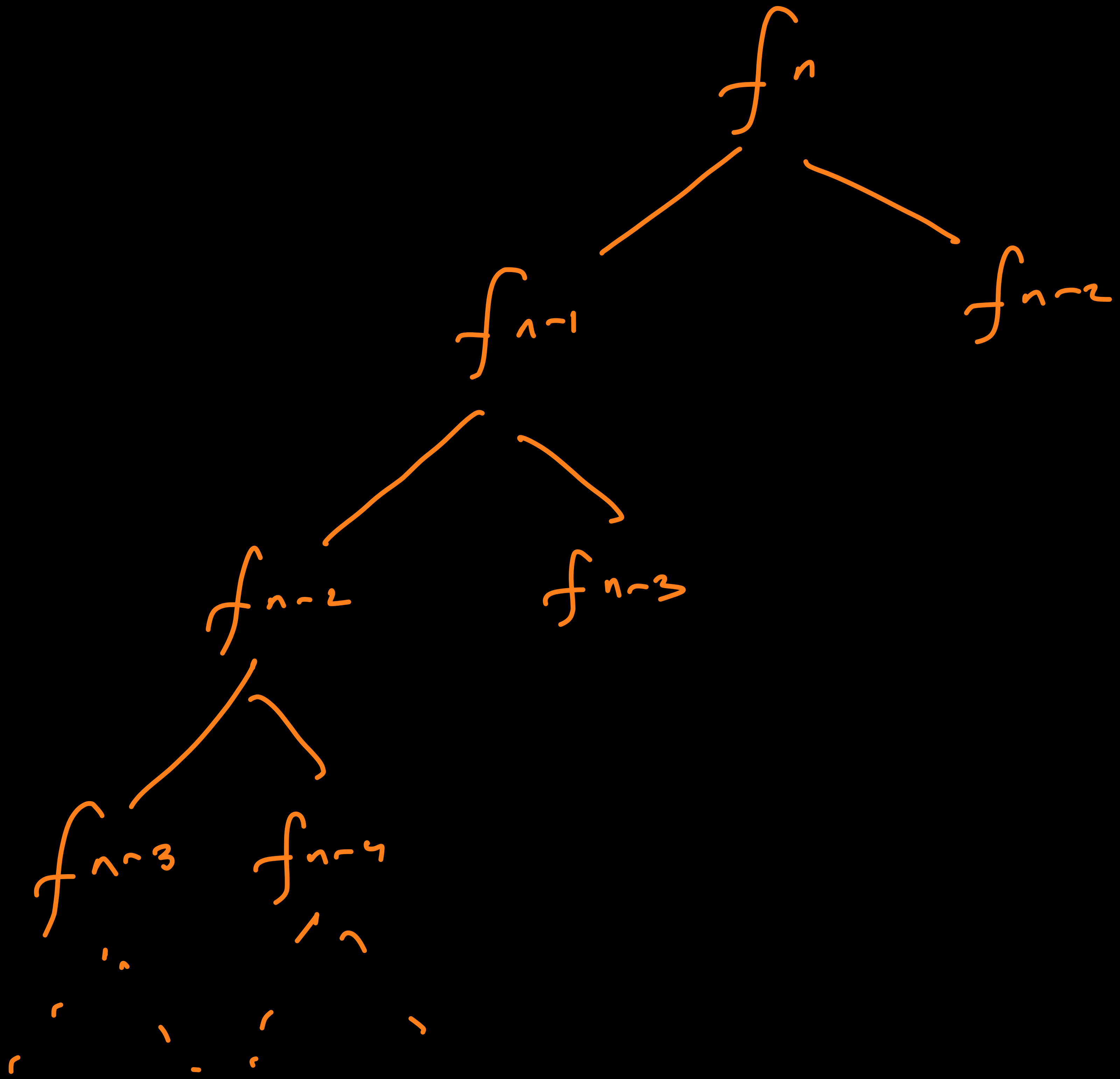
dfs

initial color = 1



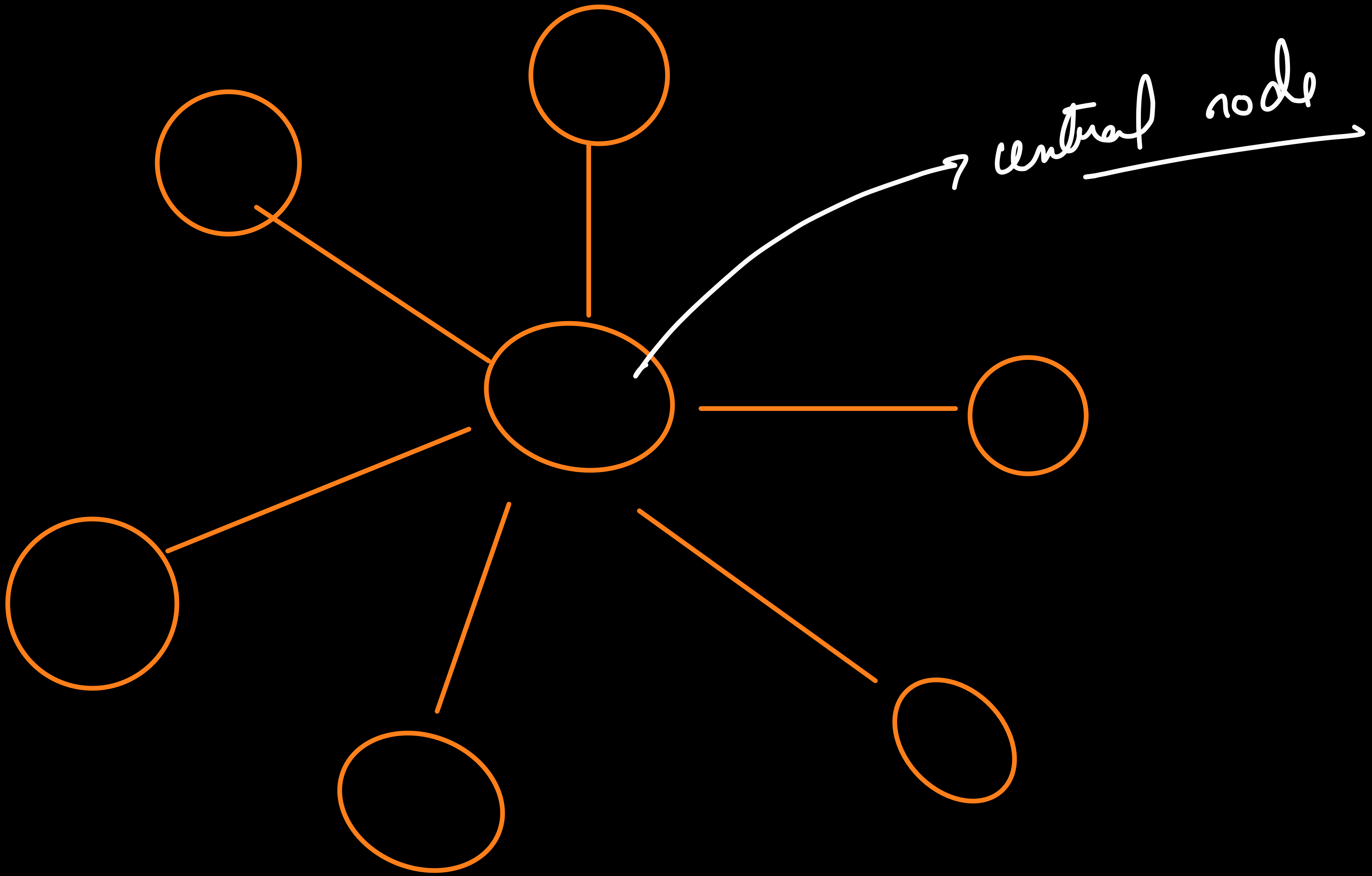
	0	1	2
0	2	2	2
1	2	2	0
2	2	0	1

$$f_n = f_{n-1} + f_{n-2}$$

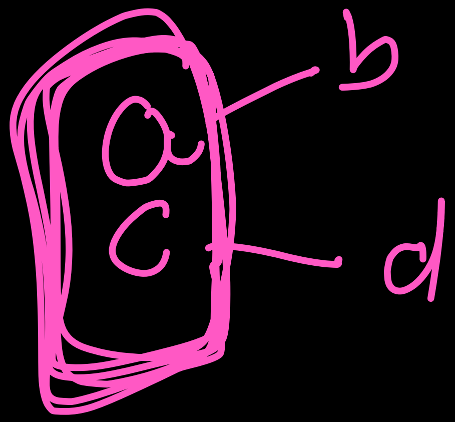


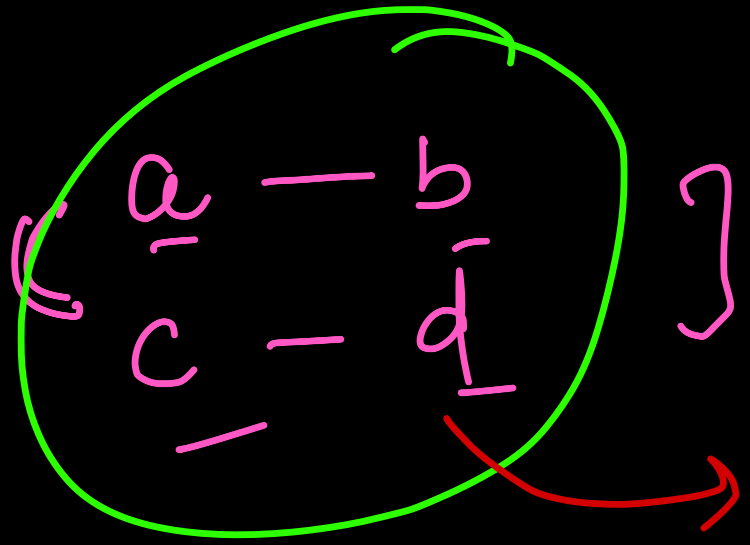
Generally a recursive sol^n is more or less DFS

Leetcode 1791 → Bruteforce → Calculate frequency of vertices in the edge list. O(V+E)



↳ pick any 2 edges, my graph is a star graph

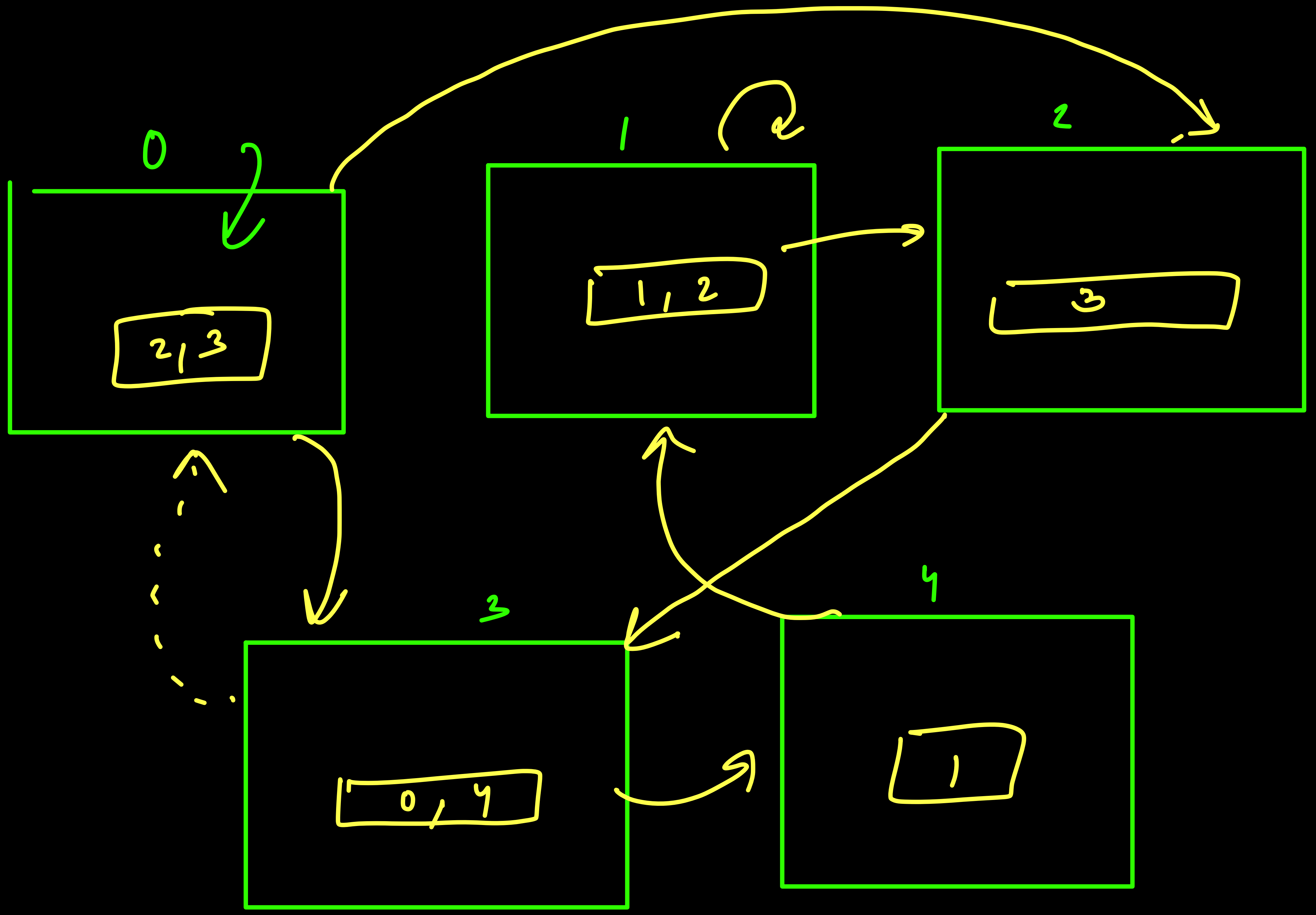




3 uniq

one of the nodes is cmnl

$$(a == c \,||\, b == c) \; ? \; c : d$$



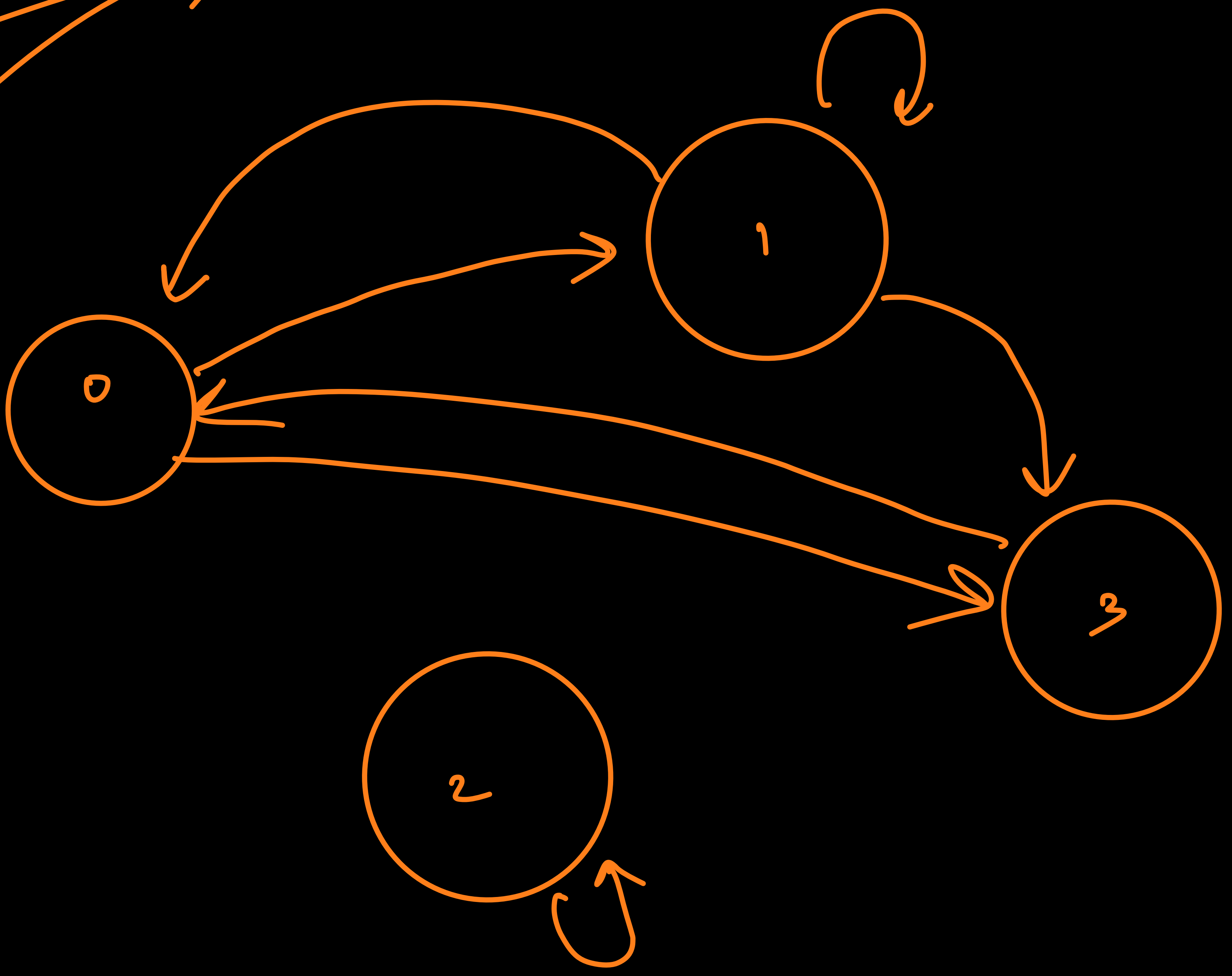
the set of keys inside a room tell us the neighbours of the node (room).

adj list

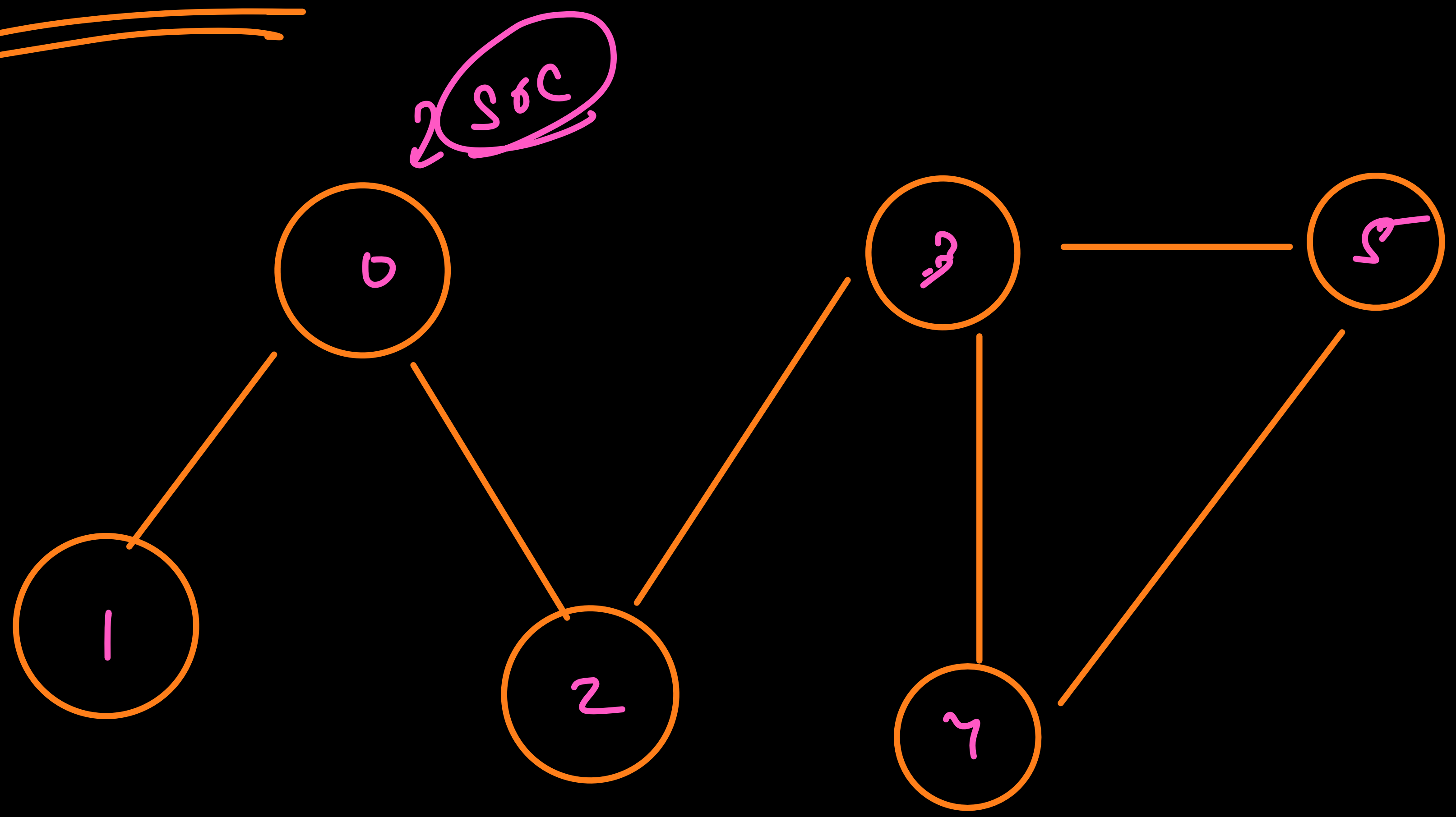
$$\left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ [1,3] & [3,0,1] & [2,] & [0,] \end{array}\right]$$

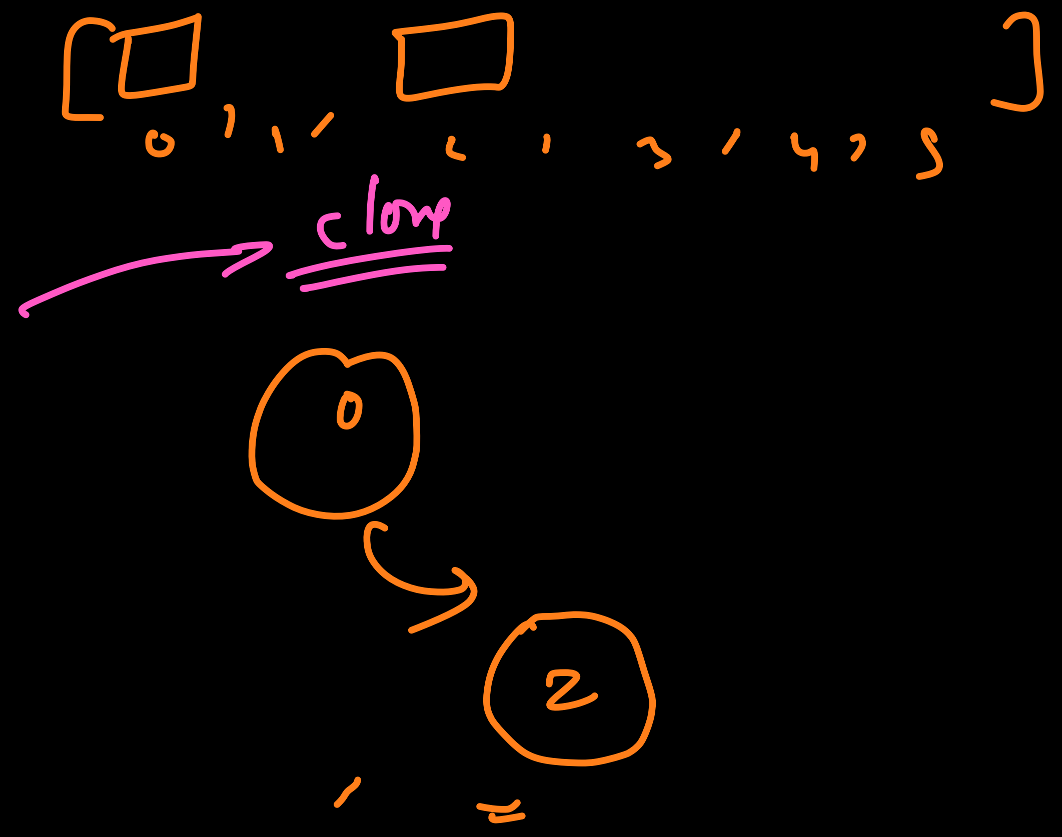
visited

Read the graph



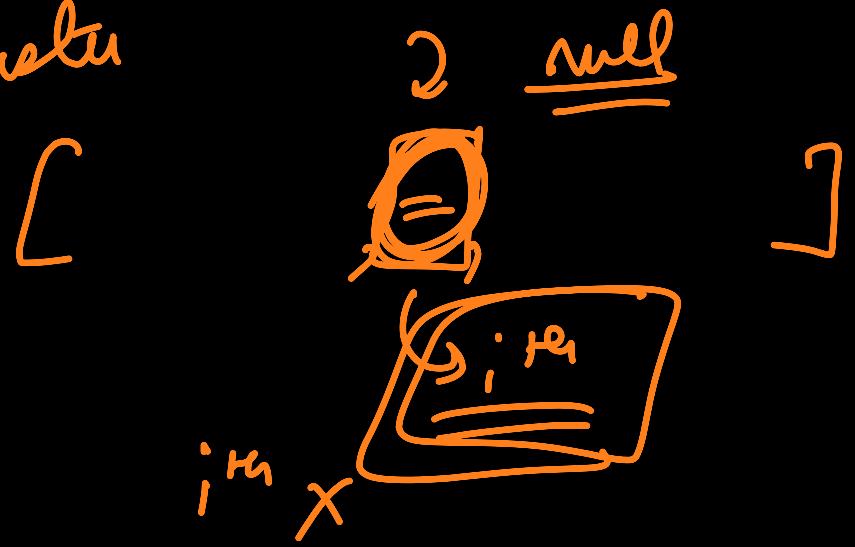
leetcode 133



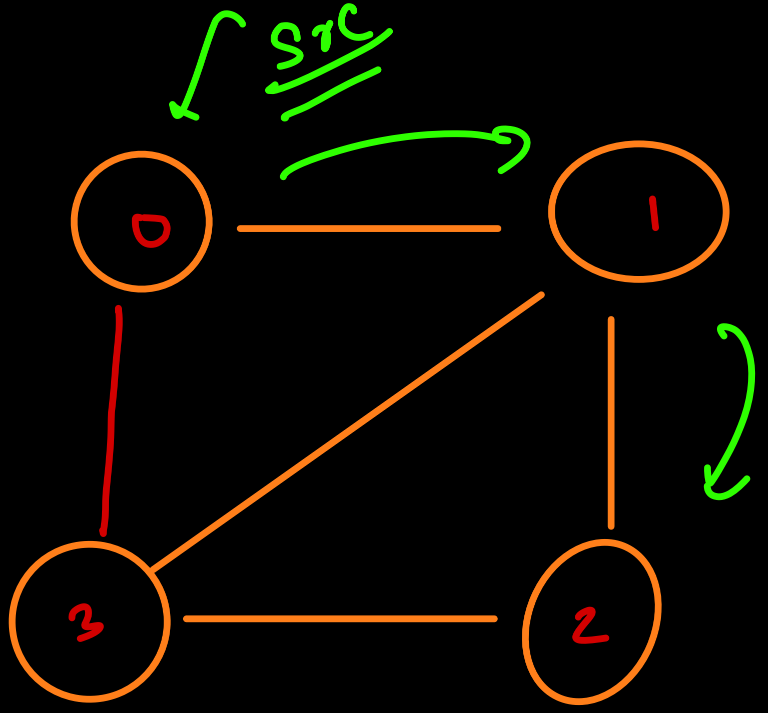


How to identify if we already created a node ?

vector < Node * > nodeRegister



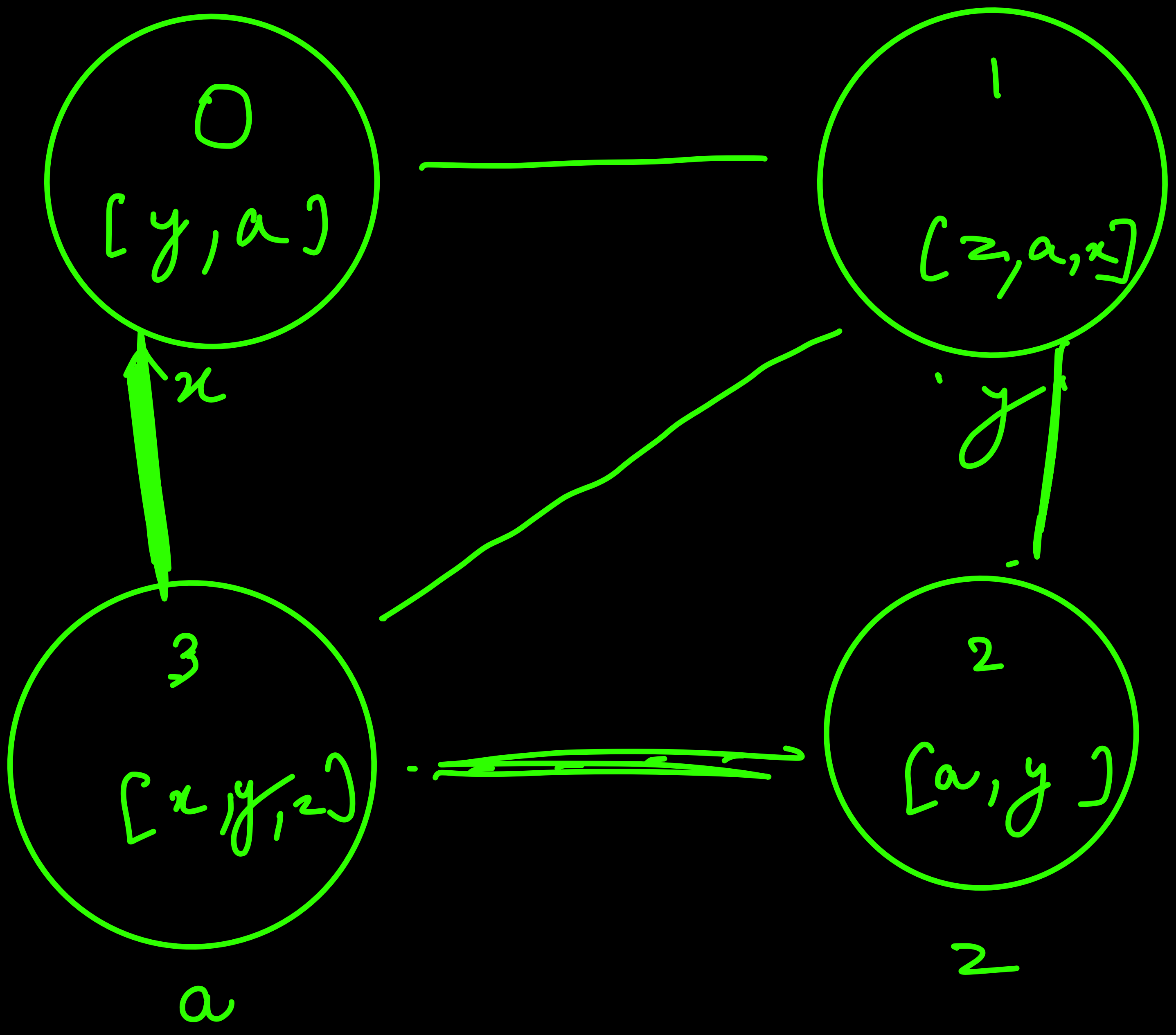
nodeRegister(i) → address of the newly created ith node



[x , y , z , a]

0 , 1 , 2 , 3

node kyila



dfs (node, clone)

```
dfs(node, clone)

    for (neigh : node)
        if (! nodeRegte [neigh->val]) {
                    ——— create
                    ——— new
                    ——— Recur
        } else
                    ——— neigh add
        }
```

JOIN THE DARKSIDE