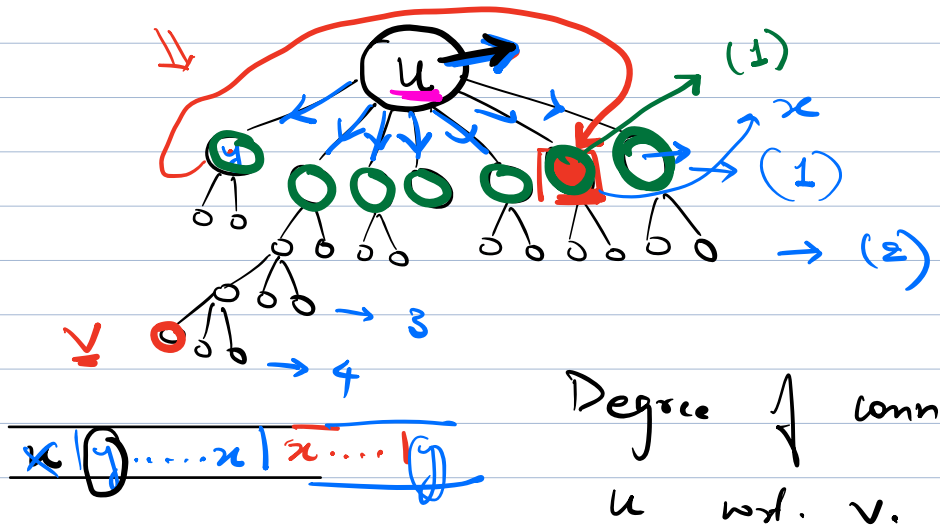
LinkedIn → 1st / 2nd / 3rd .....

↑ Degree of connection.



Degree of connection of u wrt. v.

1) DFS

2) BFS → ✓✓

BFS ⇒ Reach dest via shortest possible path

Breadth first Search

DS: Queue

1) Add source to queue & mark as visited.

2) While (! queue.isEmpty) {

1) Deque the first element.

2) Enque all unvisited neighbours of this element & also mark them as visited.

N → [1 to N]

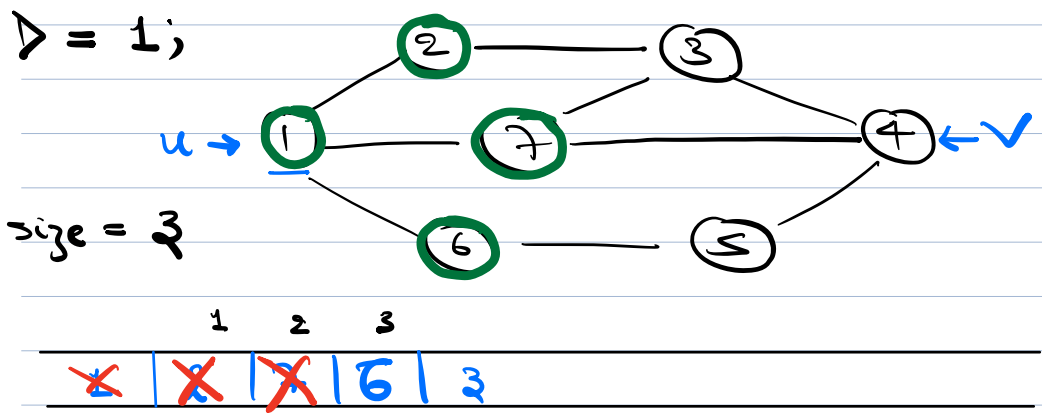
visited boolean array of size N+1.

---

undirected / unweighted.

Given a graph & 2 nodes u & v.

Find the shortest distance b/w u & v.



1 T
2 T
3 T
4 F
5 F
6 T
7 T

Code (9:45)

bool visited[N+1] = {false};

```
bfs ( source , dest ) {
    q ⇒ queue
    q.push (source);
    visited [source] = True;
    int distance = 0;
    while ( ! q.isEmpty() ) {
        int size = q.size();
        for (i = 0; i < size ; i++) {
            v = q.dequeue();        // Adjacency list of v
            for ( all unvisited u connected to v ) {
                if ( u == dest ) {
                    return (distance + 1);
                }
                visited [u] = True;
                q.enque (u);
            }
        }
        distance ++;
    }
}
```

V & E

T.C. = O(V + E)

Depth first Search

→ Select one neighbour. & explore that completely & then go to the next neighbour.

Code

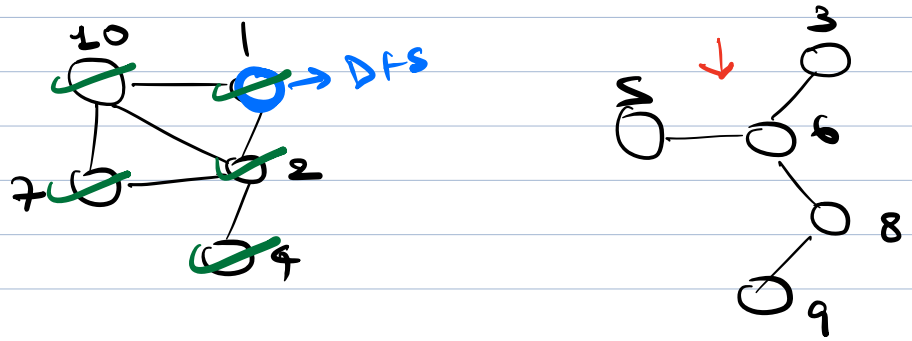
```
bool visited [N+1] = {false};

dfs ( source ) {
        visited [source] = True;
        for ( all u connected to source) {
                if (visited [u] == false) {
                        dfs (u);
                }
        }
        // logic
}
```

T.C. = O(V+E)

Q If BFS or DFS is called on any one node of the graph. Is it guaranteed all the nodes will be visited.

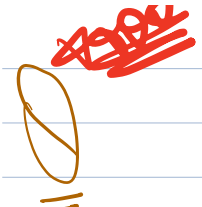
NO



Call BFS / DFS for all unvisited in a loop

```
visited [N+1] = { false };

for (i = 1; i <= N; i++) {        // Looping over all nodes
    if (visited[i] == false)
        bfs(i);
}
```



Count the total no. of connected component in a graph.

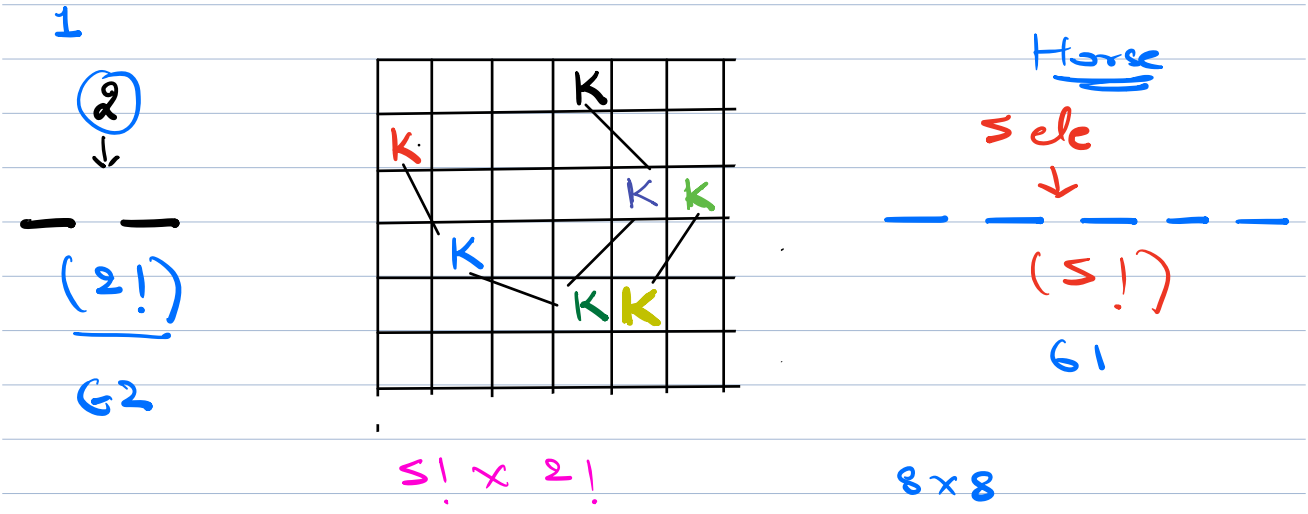
```
count = 0;
for ( all nodes u) {
    if (visited [u] == false) {
        count ++;
        bfs / dfs (u);
    }
}
```

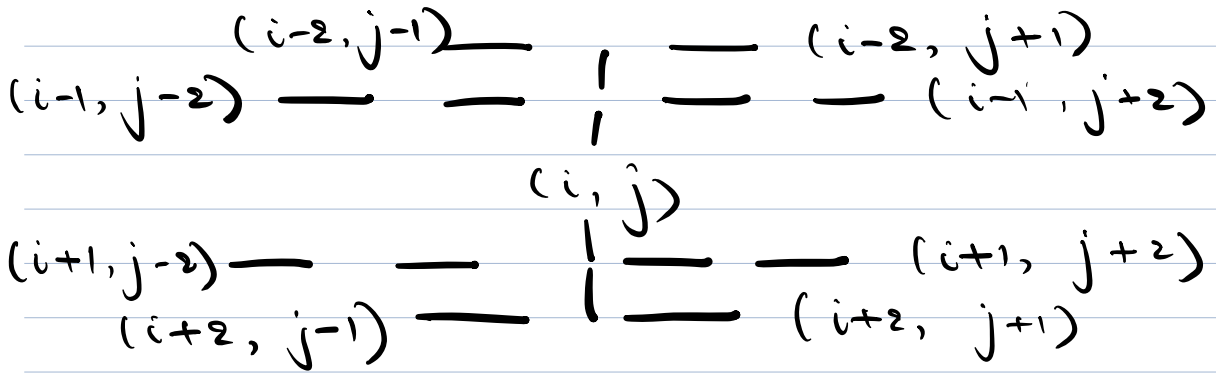
Given a NXN chessboard.

Google K Knights placed on the chessboard.

Swap 2 knights if they are reachable from each other

find the total no. of ways to arrange these knights.





Soln

ans = 1

Iterate over all connected components: {

find count

ans = ans x (count!)

}

H.W. ⇒ Code

Given a 2D matrix. (city)

Cell ⇒ Residence (R)
Cell ⇒ Hospital (H)

For every residence, find the min distance from a hospital.

	0	1	2	3
0	R	R	R	H
1	R	R	H	H
2	R	H	H	R

N x M

⇒

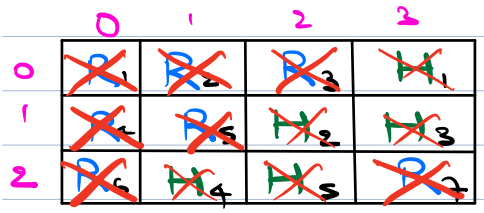
3	2	1	0
2	1	0	0
1	0	0	1

N x M

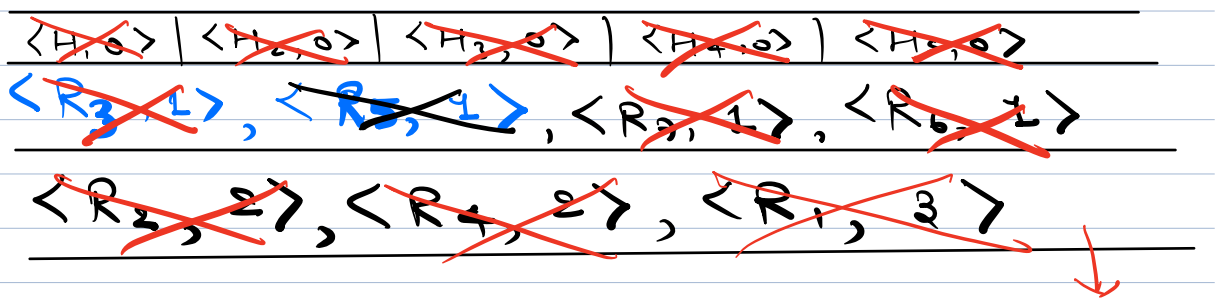
O(N x M) residences.

Soln 1) Call BFS from every residence.

$$T.C. = O(N \times M) \times O(N \times M)$$
$$= O(N^2 M^2)$$



3	2	1	0
2	1	0	0
1	0	0	1



$\langle H_1, 0 \rangle \rightarrow \langle R_3, 1 \rangle$

$\langle H_2, 0 \rangle \rightarrow \langle R_5, 1 \rangle$ (+1)

$\langle H_3, 0 \rangle \rightarrow \langle R_9, 1 \rangle$

$\langle H_4, 0 \rangle \rightarrow \langle R_6, 1 \rangle$

$\langle H_5, 0 \rangle \rightarrow$ No unvisited neighbour

$\langle R_3, 1 \rangle \rightarrow \langle R_2, 2 \rangle$ (+1)

$\langle R_3, 1 \rangle \rightarrow \langle R_4, 2 \rangle$

$\langle R_7, 1 \rangle$ No unvisited neighbour

$\langle R_6, 1 \rangle$ No unvisited neighbour

$\langle R_2, 2 \rangle \rightarrow \langle R_1, 3 \rangle$

T.C. = $O(N \times M)$

$\Rightarrow$ Multi Sourced BFS

Code