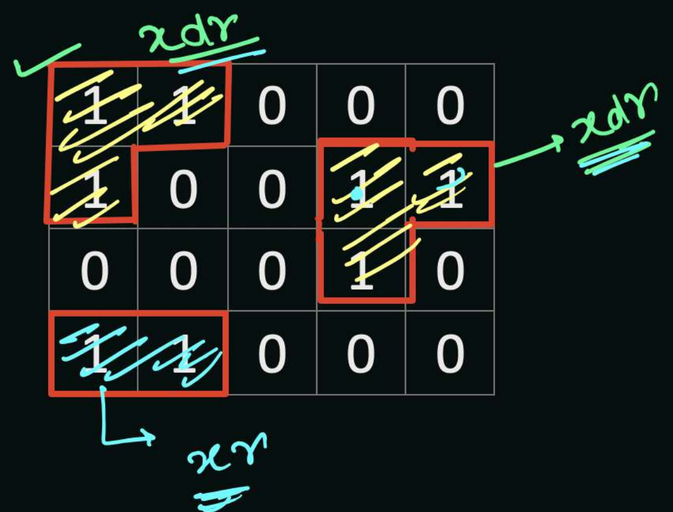
Number of Distinct Islands

Saturday, 25 September 2021 1:05 PM



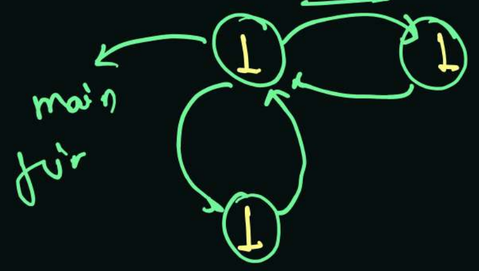
→ xdr
→ xr
] 2 islands

① add direction, in which we are moving and store string in a Hashset.

String = xdr

Call → To, left down Right

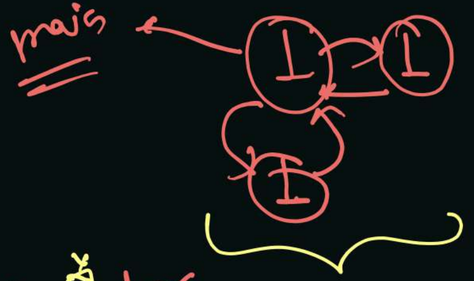
For Ex



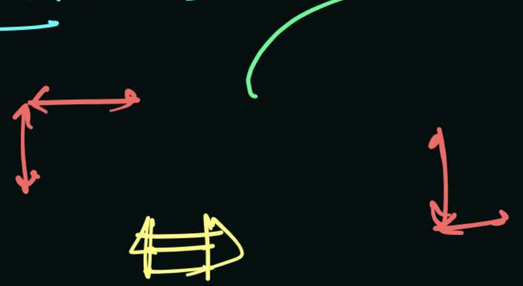
String = xr

maintain track of backtracking.

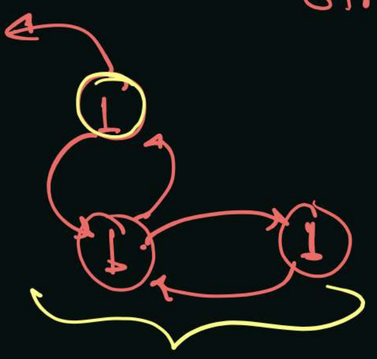
Drawback →



String = xdr

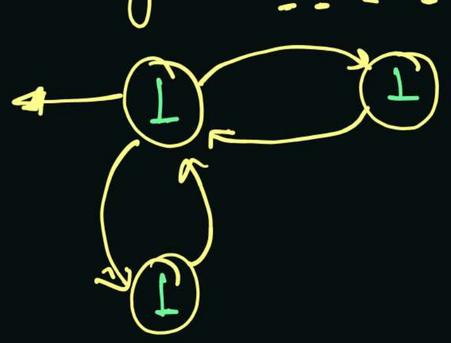


String = xdr



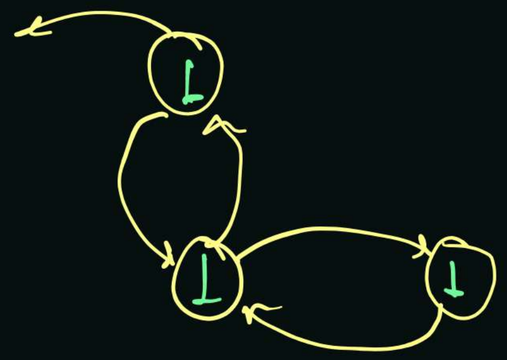
Backtrack Top → left → down → Right

String = xdzrzzz



"xdzrzz"

String = xdrzzzz



"xdrzzzz"

Return → add 'z' in string

String builder → append

StringBuilder concatenation → O(1)

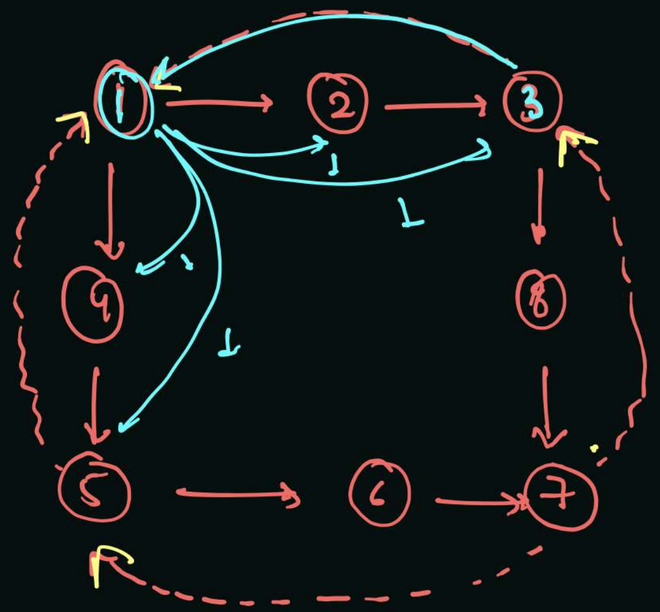
String - concatenation → O(n)

Bus Routes

Saturday, 25 September 2021 7:04 PM

src, dst

Find Min no. of bus from source to dst



src = 1

Hint

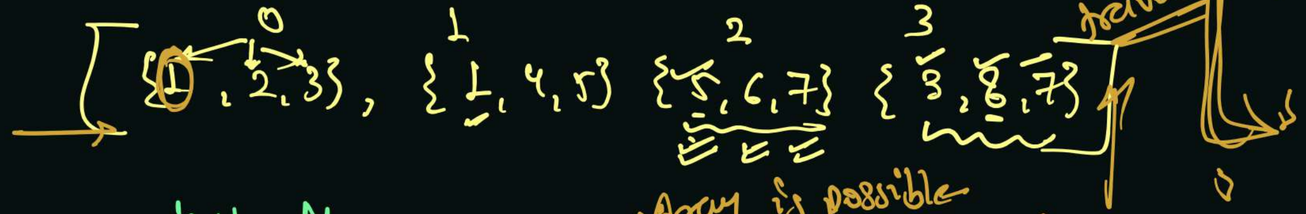
HashMaps → bus stop vs. bus No.

1, 2, 3, 4, 5 — first bus

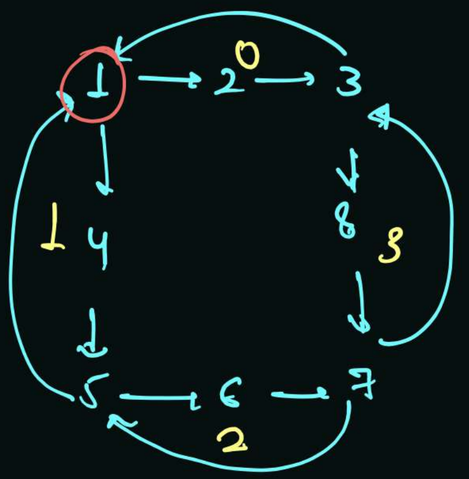
6, 7, 8, — second bus

Min no. of bus from src to dst

HashMap< Integer , AL<Integer> > map



while ( ) {
level order traversal
}



bus stop → bus No.

1	→	0 , 1
2	→	0
3	→	0 , 3
4	→	1
5	→	1 , 2
6	→	2
7	→	2 , 3
8	→	3

Set → Array is possible

visited bus No. → 0, 1, 3, 2

visited bus stop Set (Set) → 1, 2, 3, 4, 5, 8, 7

src = 1 , dst = 7

BFS

BFS

level = 0 1 (2)

1 | 2 | 8 | 4 | 5 | 8 | 7 | 6

0 1 2

rem = 1 : bus = { 0, 1 }

rem = 2 : bus = { 0 }

rem = 3 : bus = { 0, 3 }

rem = 4 : bus = { 1 }

rem = 5 : bus = { 1, 2 }

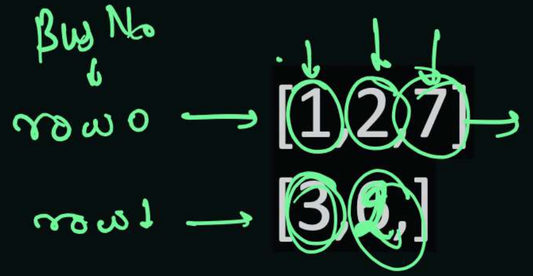
rem = 8 : bus = { }

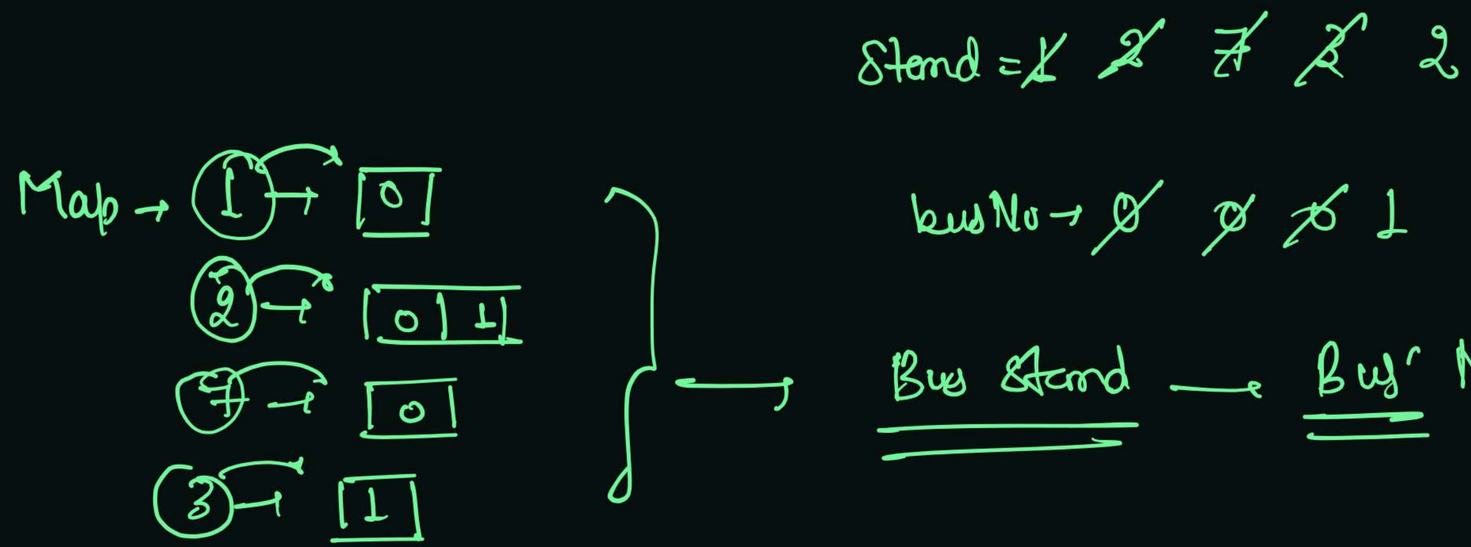
rem = 7

rem == dst

level is Result

```java
for(int r = 0; r < routes.length; r++) { // r -> bus number
    for(int c = 0; c < routes[r].length; c++) {
        int stand = routes[r][c];
        if(map.containsKey(stand) == true) {
            // key is already present, so just add value i.e.
            map.get(stand).add(r);
        } else {
            ArrayList<Integer> busNo = new ArrayList<>();
            busNo.add(r);
            map.put(stand, busNo);
        }
    }
}
```





```
qu.add(source);
int level = -1;
while(qu.size() > 0) {
    int size = qu.size();
    level++;
    while(size-- > 0) {
        int rem = qu.remove();

        if(rem == target) return level;

        for(int busNum : map.get(rem)) {
            if(visBus.contains(busNum) == true) {
                continue;
            } else {
                for(int busStand : routes[busNum]) {
                    if(visStand.contains(busStand) == false) {
                        visStand.add(busStand);
                        qu.add(busStand);
                    }
                }
                visBus.add(busNum);
            }
        }
    }
}
return -1;
```

```
HashMap<Integer, ArrayList<Integer>> map = new HashMap<>();
makeMapWithStandAndBus(routes, map);

HashSet<Integer> visBus = new HashSet<>();
HashSet<Integer> visStand = new HashSet<>();
Queue<Integer> qu = new LinkedList<>();
```

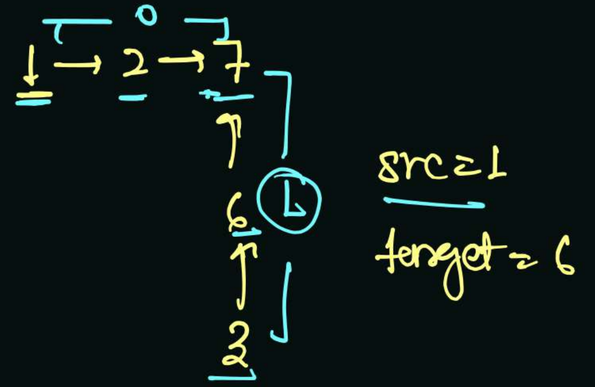
Result

1 → 0
2 → 0
8 → 1
6 → 1
7 → 0,1

TRICK

visRoute = 0, 1
visStand = 1, 2, 7, 8, 6
level = 1 2

Modified BFS



1 2 7 8 6

rem = 1
BusNo → {0}

rem = 2
BusNo → {0}

rem = 7
{0, 1}

rem = 3
BusNo → {1}

rem = 6

stand associated with route

size = 1, 2

Minimum Cost to connect All Cities

Saturday, 25 September 2021 7:04 PM

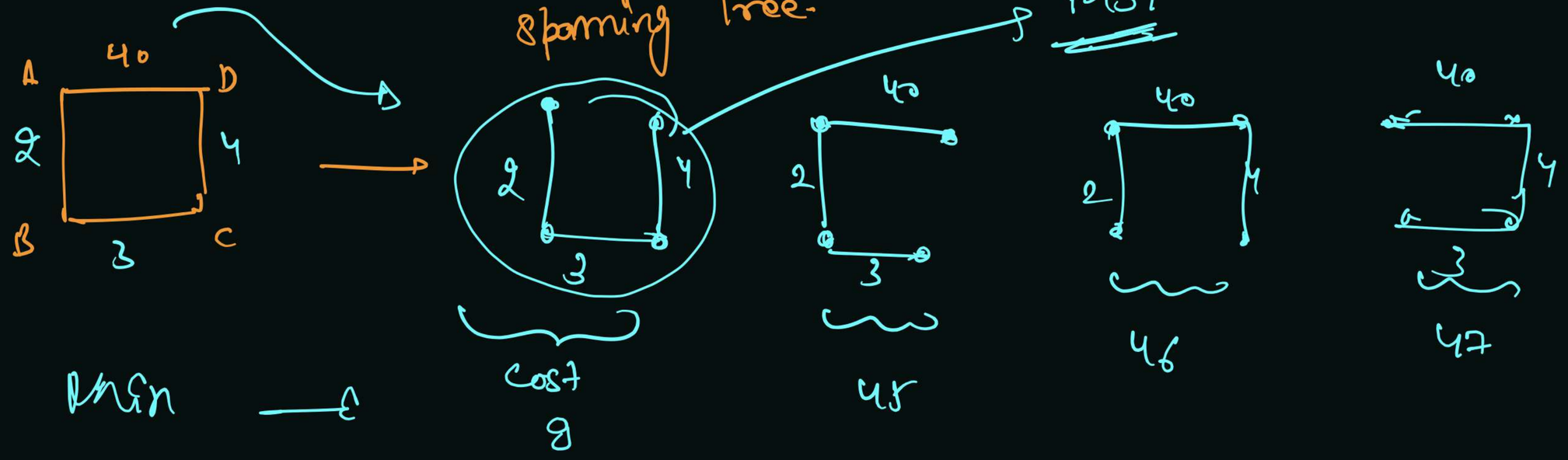
Prims Algo.

Minimum Spanning Tree → connected and acyclic

Minimum possibility.

Algo =

BFS → Replace queue from priority queue Min priority on weight



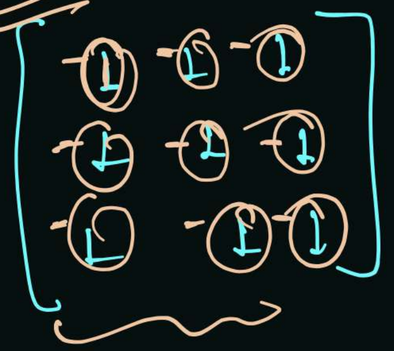
Zero One Matrix

Saturday, 25 September 2021 7:04 PM

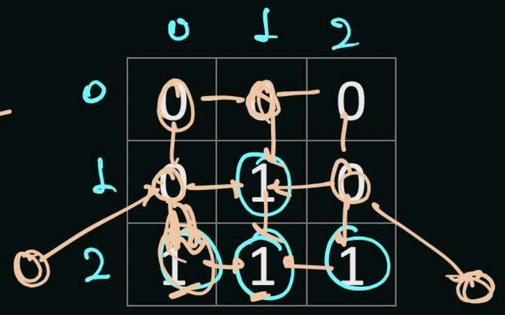
Brute force = BFS from all the 1

- Time → $V(V+E)$ = $O((V)(V+E))$ $V \rightarrow (V+E)$ (V=3)

worst case

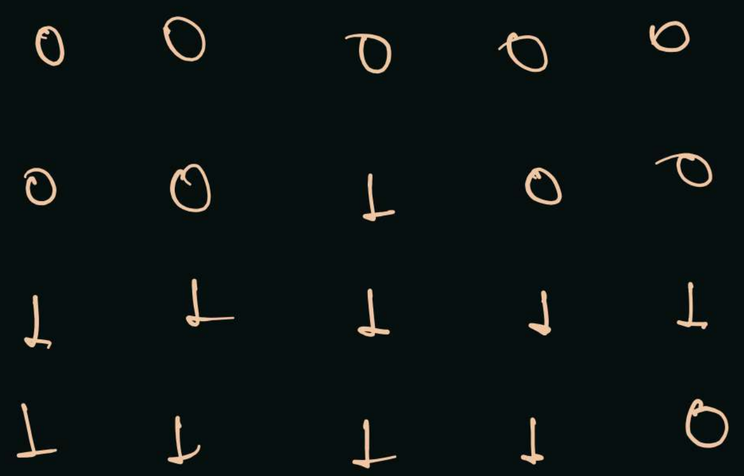


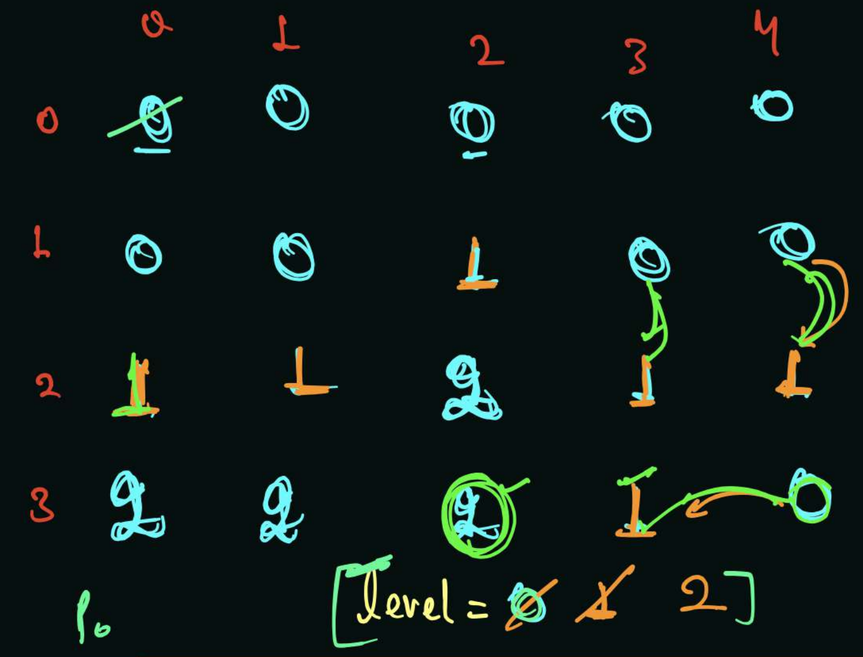
$V+E$

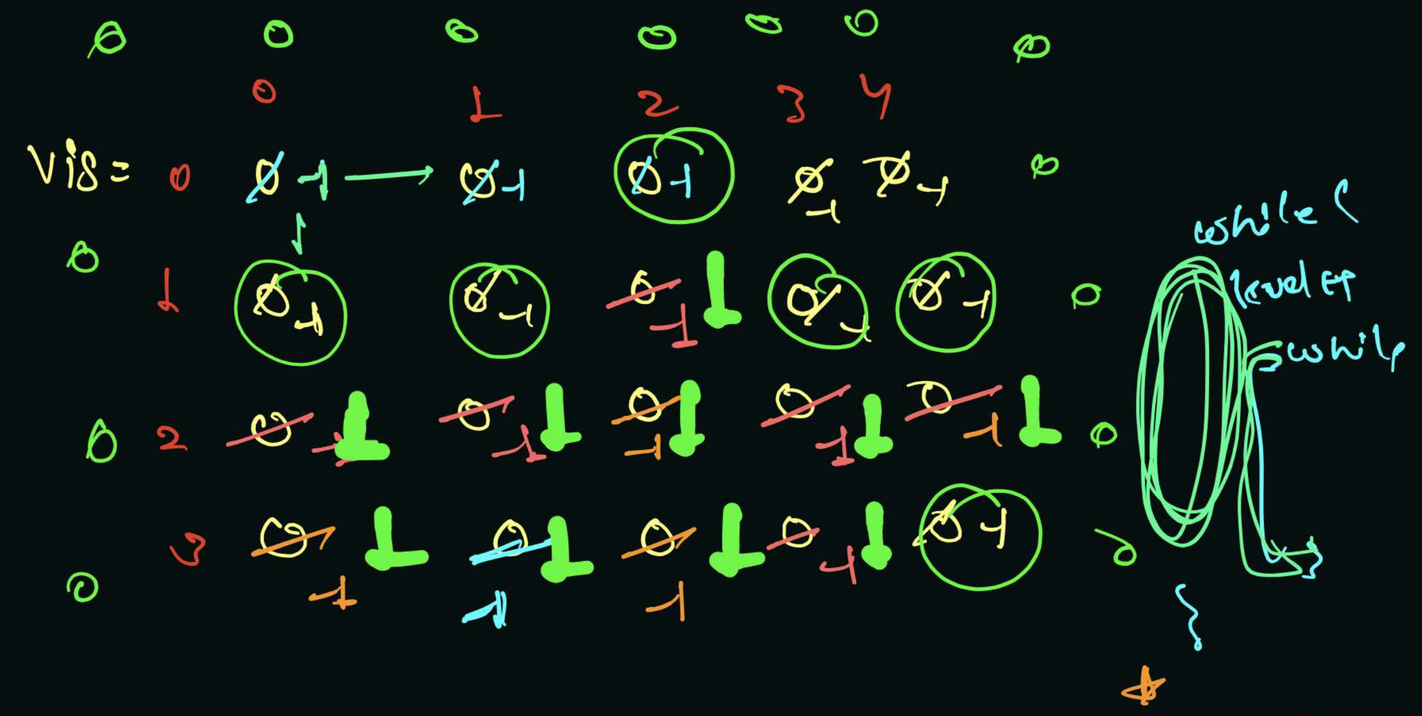


Apply BFS from all 0's (Simultaneously) treat every one as unique.

In all 0's, there is call for single level







(0,0) | (0,1) | (0,2) | (0,3) | (0,4) | (1,0) | (1,1) | (1,3) | (1,4) | (3,4) | (1,2) | (2,0) | (2,1) | (2,3) | (2,4) | (3,3)

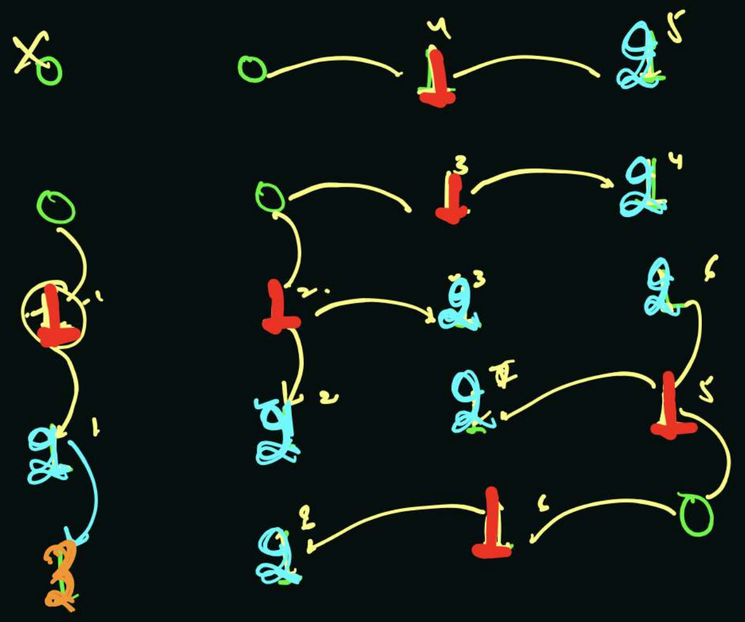
Set level at index

size = 10 6 4

(2,2) | (3,0) | (3,1) | (3,2) |

2

BFS



BFS ⟶ already marked → with Smaller level or equal level.