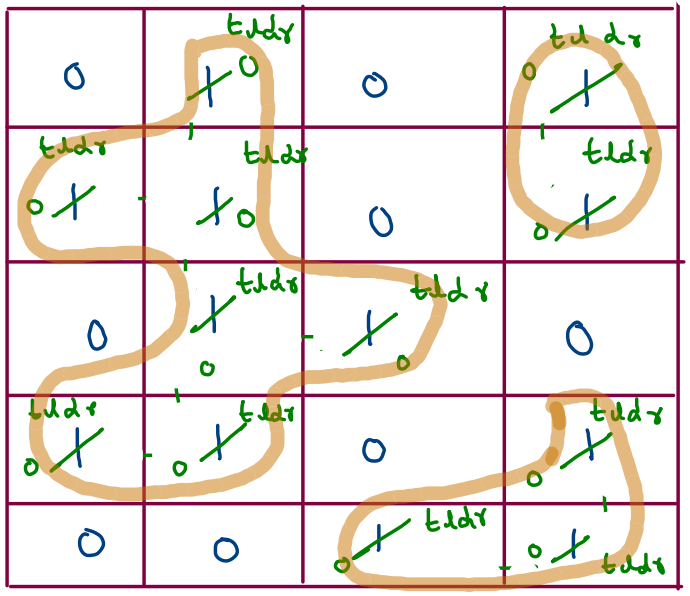
200. Number of Islands

```java
public int numIslands(char[][] grid) {
    int count = 0;

    for(int i=0; i < grid.length;i++) {
        for(int j=0;j < grid[0].length;j++) {
            if(grid[i][j] == '1') {
                dfs(grid,i,j);
                count++;
            }
        }
    }

    return count;
}
```



dfs (0,1)
dfs (0,3)
dfs (3,3)

count = ~~0~~ ~~1~~ ~~2~~ 3

```java
int[][]dir = {{-1,0},{0,-1},{1,0},{0,1}}; //tldr

public void dfs(char[][]grid,int i,int j) {

    grid[i][j] = '0';

    for(int k = 0; k < 4;k++) {
        int ni = i + dir[k][0];
        int nj = j + dir[k][1];

        if(ni >= 0 && ni < grid.length && nj >= 0 && nj < grid[0].length && grid[ni][nj] == '1') {
            dfs(grid,ni,nj);
        }
    }
}
```

860 · Number of Distinct Islands

dlr

0	1	0	0	1
1	1	1	0	1
0	0	0	0	0
1	0	0	1	0
1	0	1	1	1
0	1	0	0	0

t, l, d, r

dlr

1 , 1 1

1 1 1 , 1 1

to solve the above problem, involve backtracking step.

dlbrbbb

1

1 1 1

dlbbrbb

1 1

1 1

```java
public int numberofDistinctIslands(int[][] grid) {
    HashSet<String>hs = new HashSet<>();

    for(int i=0; i < grid.length;i++) {
        for(int j=0; j < grid[0].length;j++) {
            if(grid[i][j] == 1) {
                sb = new StringBuilder();
                dfs(grid,i,j);
                hs.add(sb.toString());
            }
        }
    }

    return hs.size();
}
```

d r b b b

d d r b b b b

d d b r b b b

1	0	0	1	0
1	1	0	1	0
0	0	0	1	1
1	0	0	0	0
1	1	0	1	0
1	0	0	1	1

sb = d r b b b

994. Rotting Oranges

multiple src BFS

0	1 (1)	0	1 (3)
1 (1)	2	1 (1)	1 (2)
0	1 (1)	1 (1)	0
0	1 (1)	2	1 (1)
1 (3)	1 (2)	0	0

0 -> empty cell

1 -> fresh orange

2 -> rotten orange

BFS: min no. of steps

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

```
Pair {
    int i;
    int j;
    int t;
}
```

① remove

② add unvis nbr and mark them.

~~1,1,0~~ | ~~3,2,0~~ | ~~0,1,1~~ | ~~1,0,1~~ | ~~1,2,1~~ | ~~2,1,1~~ | ~~2,2,1~~ | ~~3,1,1~~ | ~~3,3,1~~ | ~~1,3,2~~ | ~~3,0,2~~ | ~~0,3,3~~

while (q.size() > 0) {

1. remove
2. mark*
3. work
4. add unvisited nbr

}

→ marking on removal

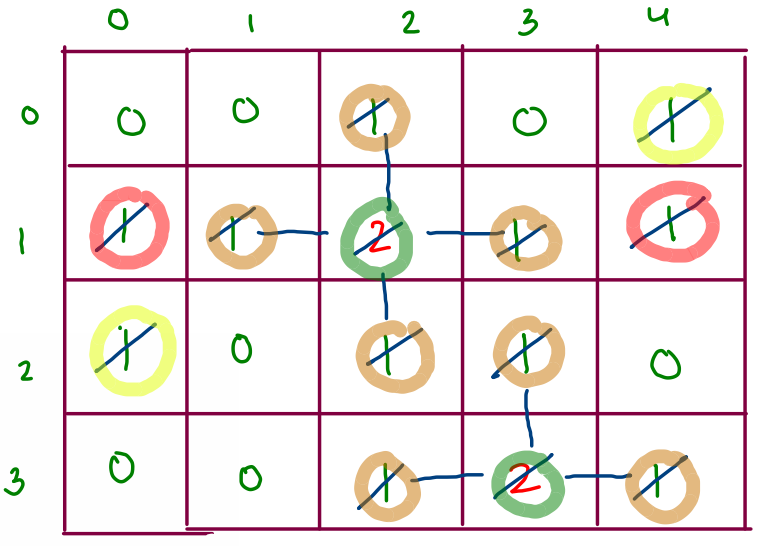
while (q.size() > 0) {

1. remove
2. work
3. add unvis nbr and mark them visited

}

→ marking on addition

```java
for(int i=0; i < grid.length;i++) {
    for(int j=0; j < grid[0].length;j++) {
        if(grid[i][j] == 2) {
            q.add(new Pair(i,j,0));
        }
        else if(grid[i][j] == 1) {
            fo++;
        }
    }
}
```



time = 3

```java
//bfs
while(q.size() > 0) {
    //remove
    Pair rem = q.remove();
    time = Math.max(time,rem.t);

    //add unvis and mark them as well
    for(int k = 0; k < 4;k++) {
        int ni = rem.i + dir[k][0];
        int nj = rem.j + dir[k][1];

        if(ni >= 0 && ni < grid.length && nj >= 0 && nj < grid[0].length && grid[ni][nj] == 1) {
            q.add(new Pair(ni,nj,rem.t+1));
            grid[ni][nj] = 2;
            fo--;
        }
    }
}

if(fo == 0) {
    return time;
}
else {
    return -1;
}
```

fo = ~~11~~ ~~7~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0