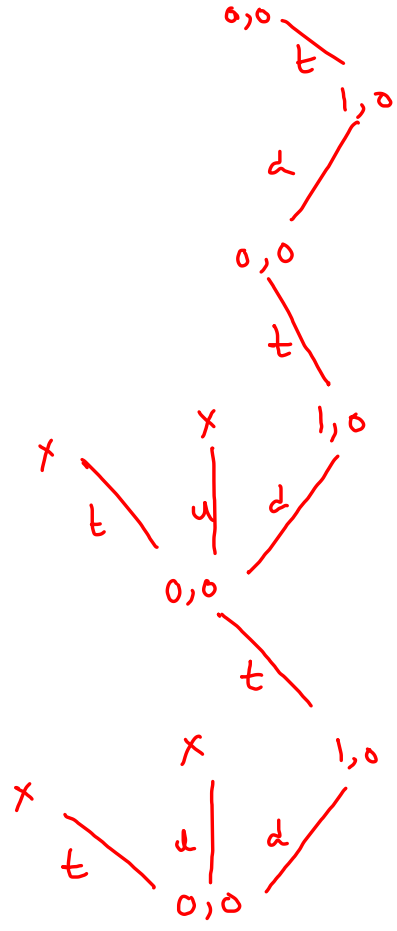
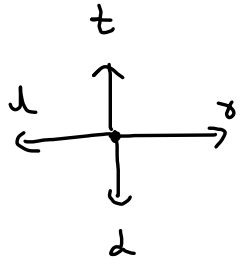
Floodfill

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	1	0	1	1	1	0
2	0	0	0	0	0	0	0
3	1	0	1	1	0	1	1
4	1	0	1	1	0	1	1
5	1	0	0	0	0	0	0

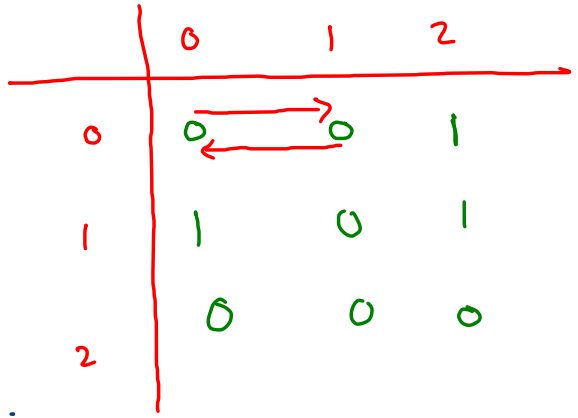


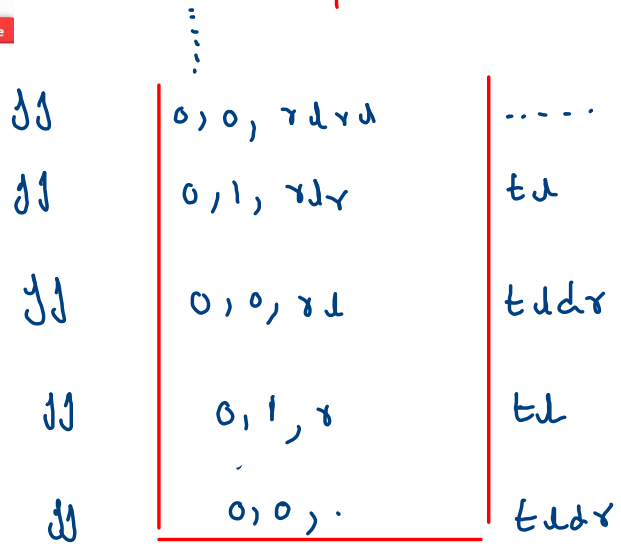
(i)

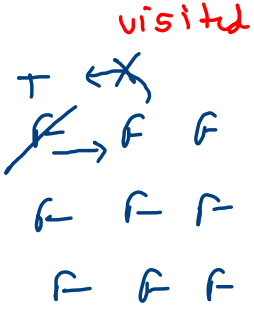


(ii) 1 obstacle

```java
public static void floodfill(int[][] maze, int sr, int sc, String asf) {
    if(sr < 0 || sc < 0 || sr >= maze.length || sc >= maze[0].length || maze[sr][sc] == 1) {
        return;
    }

    if(sr == maze.length - 1 && sc == maze[0].length-1) {
        System.out.println(asf);
    }

    //top nbr
    floodfill(maze,sr-1,sc,asf + "t");

    //left nbr
    floodfill(maze,sr,sc-1,asf + "l");

    //down nbr
    floodfill(maze,sr+1,sc,asf + "d");

    //right nbr
    floodfill(maze,sr,sc+1,asf + "r");
}
```



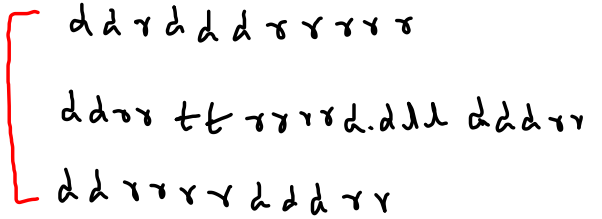


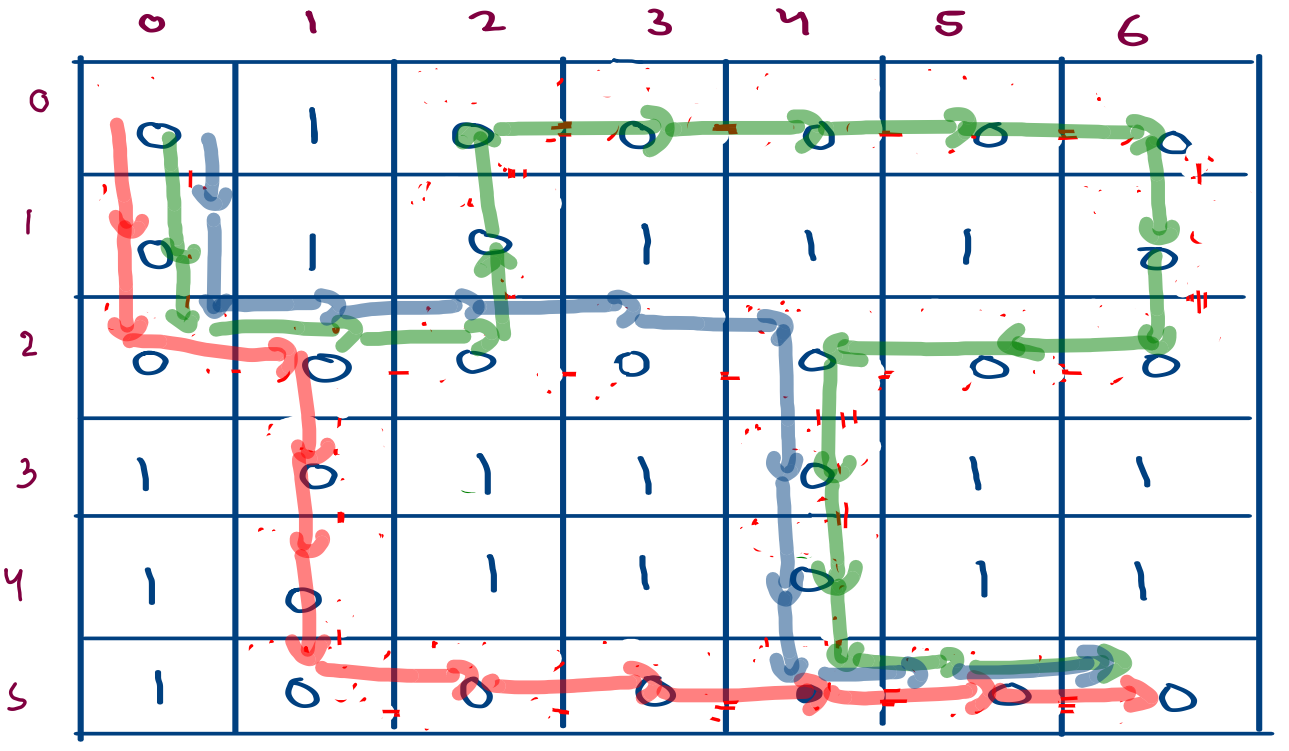
```java
if(sr < 0 || sc < 0 || sr >= maze.length || sc >= maze[0].length ||
maze[sr][sc] == 1 || vis[sr][sc] == true) {
    return;
}

if(sr == maze.length - 1 && sc == maze[0].length-1) {
    System.out.println(asf); return;
}

vis[sr][sc] = true;

//top nbr
floodfill(maze,sr-1,sc,asf + "t",vis);

//left nbr
floodfill(maze,sr,sc-1,asf + "l",vis);

//down nbr
floodfill(maze,sr+1,sc,asf + "d",vis);

//right nbr
floodfill(maze,sr,sc+1,asf + "r",vis);
```

ddrddrrrrr

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	1	0	1	1	1	0
2	0	0	0	0	0	0	0
3	1	0	1	1	0	1	1
4	1	0	1	1	0	1	1
5	1	0	0	0	0	0	0

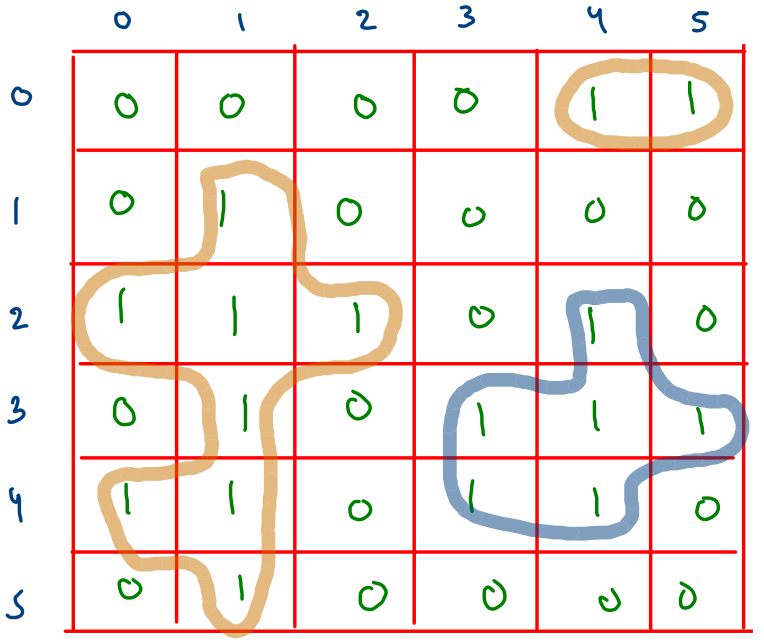
```java
if(sr < 0 || sc < 0 || sr >= maze.length || sc >= maze[0].length ||
maze[sr][sc] == 1 || vis[sr][sc] == true) {
    return;
}

if(sr == maze.length - 1 && sc == maze[0].length-1) {
    System.out.println(asf);
}

vis[sr][sc] = true;

//top nbr
floodfill(maze,sr-1,sc,asf + "t",vis);

//Left nbr
floodfill(maze,sr,sc-1,asf + "l",vis);

//down nbr
floodfill(maze,sr+1,sc,asf + "d",vis);

//right nbr
floodfill(maze,sr,sc+1,asf + "r",vis);

vis[sr][sc] = false;
```





Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return *the number of islands*.

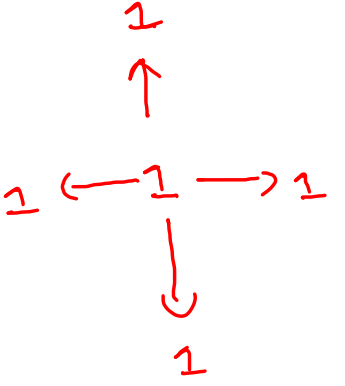
An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

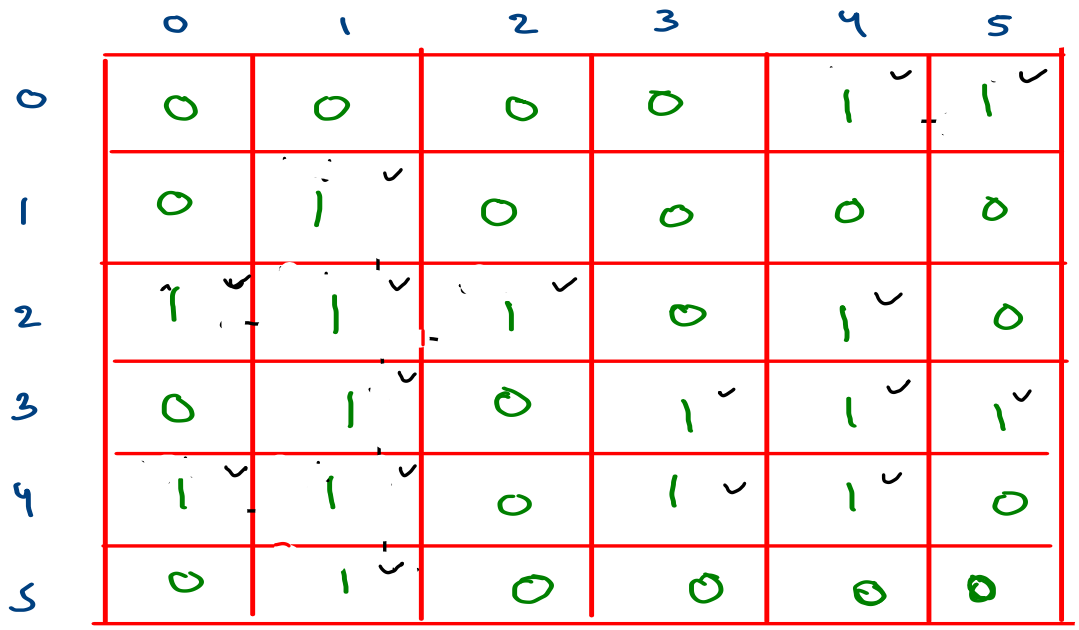


3 islands

1 -> land

0 -> water





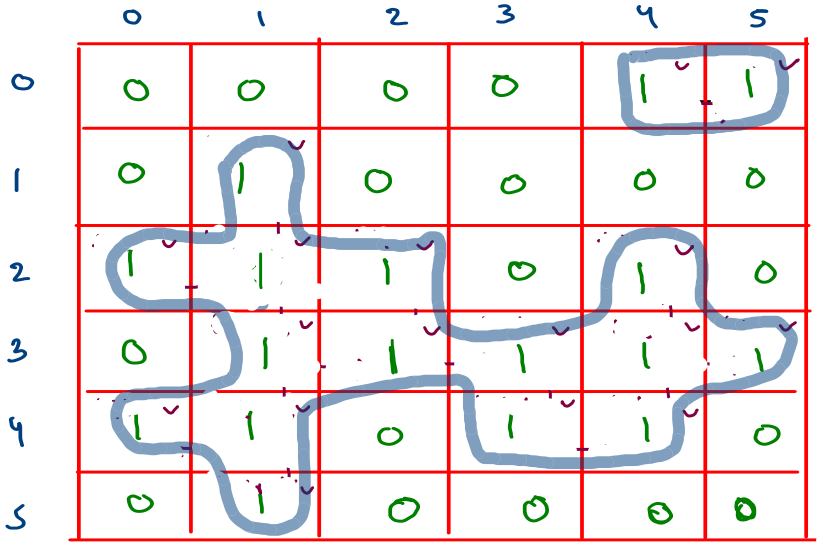
An unvisited '1'

3 island

(0,4) → go and travel your comp

(1,1)

(2,4)




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

    boolean[][]vis = new boolean[n][m];

    for(int i=0; i < n;i++) {
        for(int j=0; j < m;j++) {
            //an visited '1'
            if(grid[i][j] == '1' && vis[i][j] == false) {
                travel(i,j,grid,vis);
                count++;
            }
        }
    }

    return count;
}
```

```java
public void travel(int sr,int sc,char[][]grid,boolean[][]vis) {
    if(sr < 0 || sc < 0 || sr >= grid.length || sc >= grid[0].length
       || grid[sr][sc] == '0' || vis[sr][sc] == true) {
        return;
    }

    vis[sr][sc] = true;

    //top
    travel(sr-1,sc,grid,vis);

    //left
    travel(sr,sc-1,grid,vis);

    //down
    travel(sr+1,sc,grid,vis);

    //right
    travel(sr,sc+1,grid,vis);
}
```

```
5
10
20
30
40
50
60
```

n = 5

arr : 10 20 30 40 50

target : 60

10 20 30 → 60

10 50 → 60

20 40 → 60

arr : [2 3 1 0]
0 1 2 3

target = 3

idx — arr[idx]

3 — 0

2 — 1

2 — 1 — 0

2 - 1

1 — 3

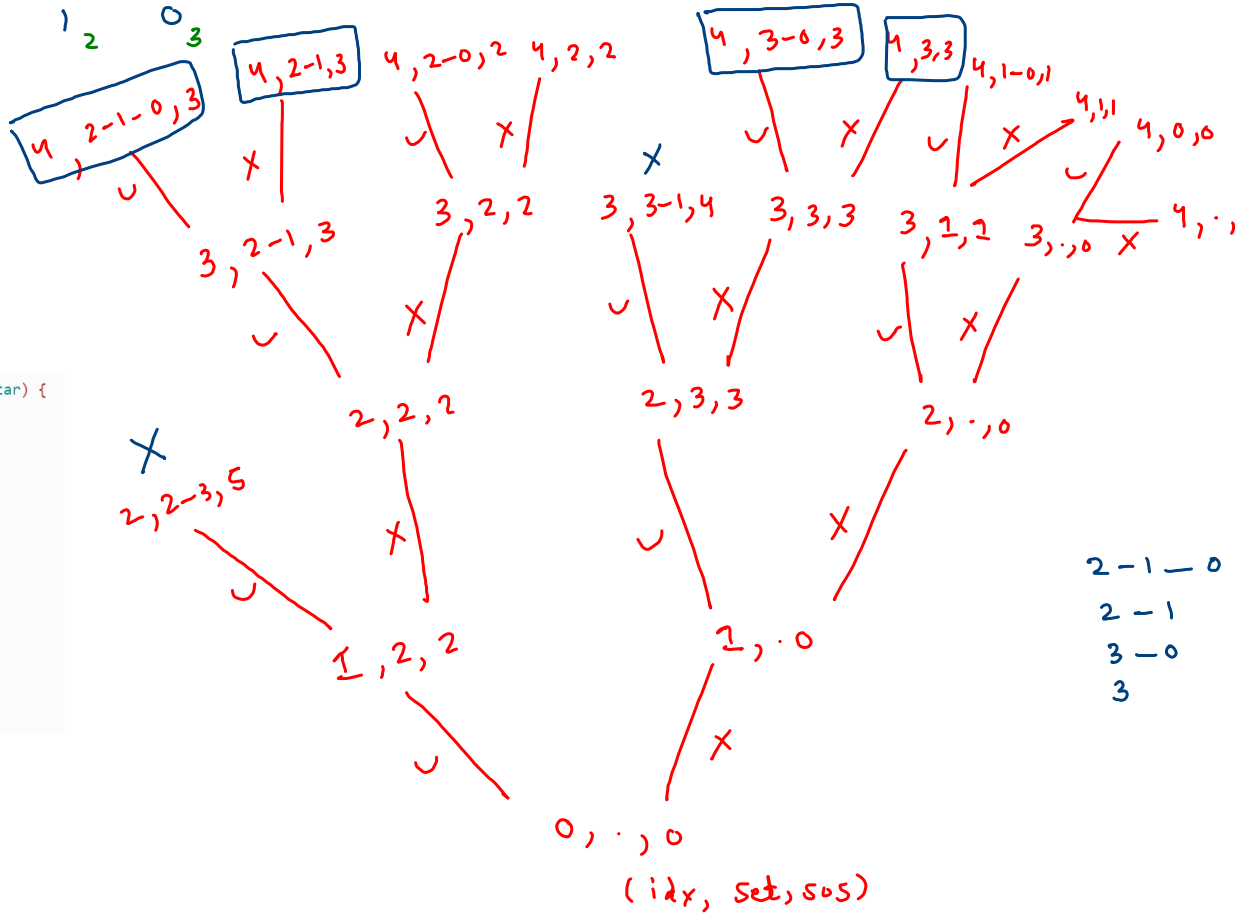
3 - 0

3

0 — 2

arr : 2 3 1 0

target = 3




```java
// tar is target
public static void printTargetSumSubsets(int[] arr, int idx, String set, int sos, int tar) {
    if(idx == arr.length) {
        if(sos == tar) {
            System.out.println(set + ".");
        }
        return;
    }

    if(sos > tar) {
        return;
    }


    //arr[idx] -> yes
    printTargetSumSubsets(arr,idx+1,set + arr[idx] + ", ",sos + arr[idx],tar);

    //arr[idx] -> no
    printTargetSumSubsets(arr,idx+1,set,sos,tar);
}
```