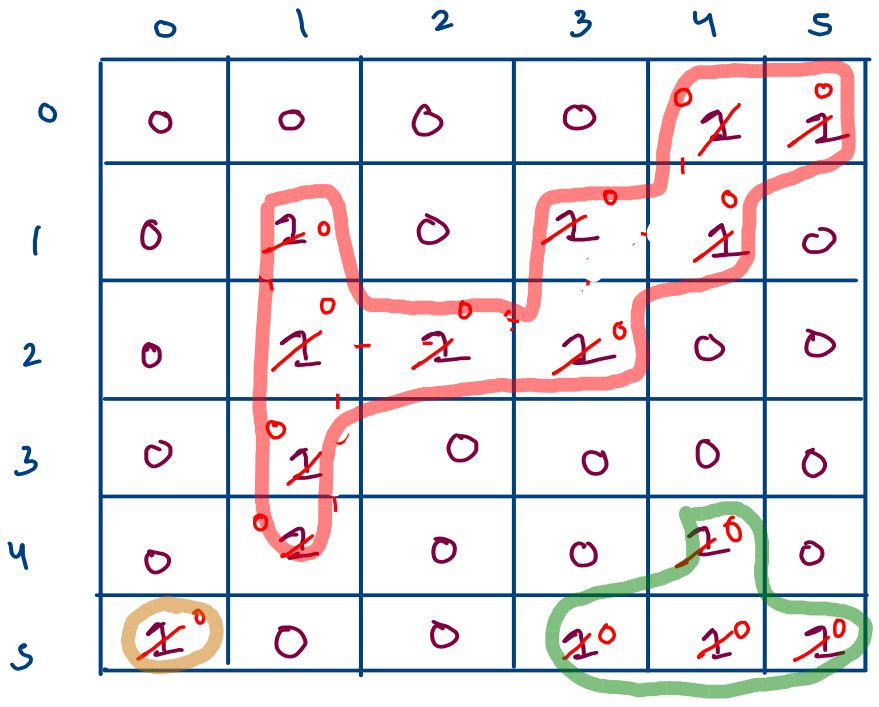
# no. of islands

0 -> water

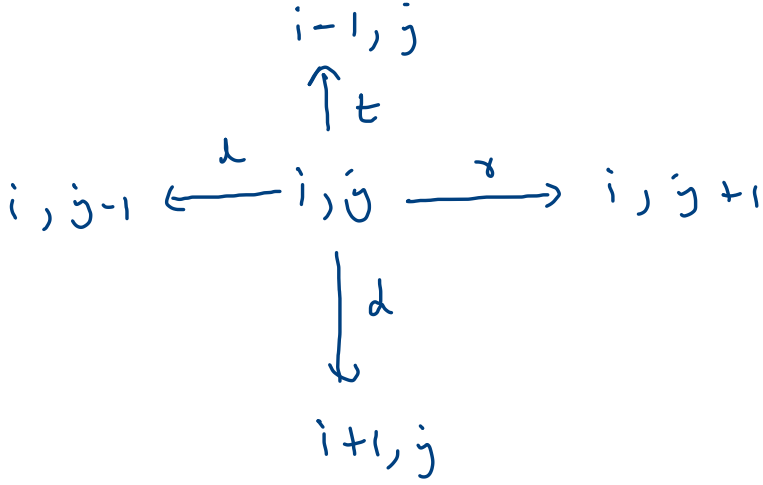
1 -> land

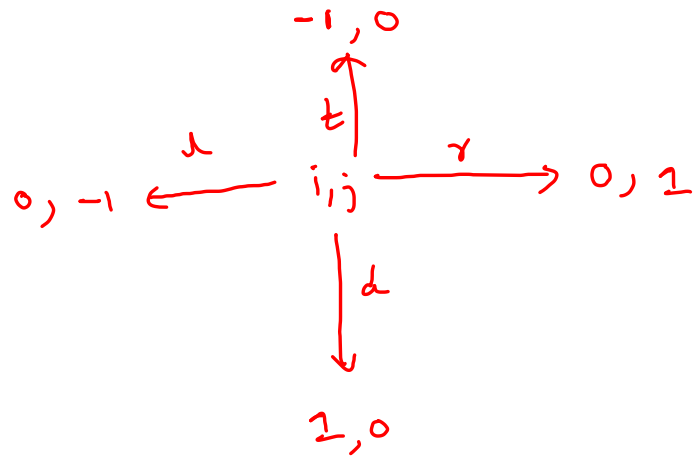



```java
//0 -> water, 1 -> land
public int numIslands(char[][] grid) {
    int count = 0;

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

    return count;
}
```

count = 0 → 1 → 2 → 3

```java
public void dfs(int i,int j,char[][] grid) {
    if(i < 0 || i >= grid.length || j < 0 || j >= grid[0].length || grid[i][j] == '0') {
        return;
    }

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

    //top
    dfs(i-1,j,grid);

    //left
    dfs(i,j-1,grid);

    //down
    dfs(i+1,j,grid);

    //right
    dfs(i,j+1,grid);
}
```

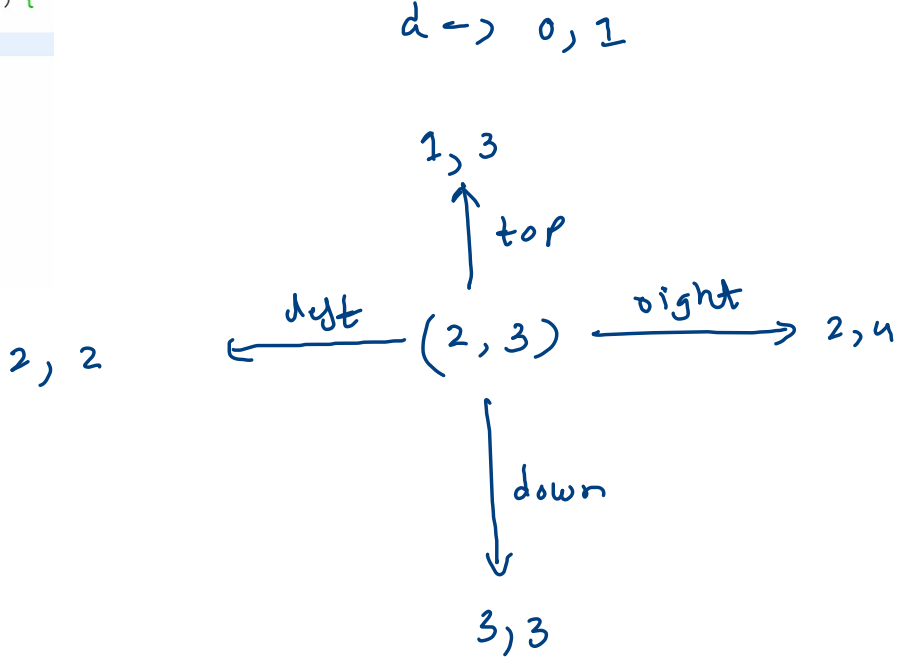



dir = [ [-1, 0], [0, -1], [1, 0], [0, 1] ]

top left down right

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

public void dfs(int i,int j,char[][] grid) {
    if(i < 0 || i >= grid.length || j < 0 || j >= grid[0].length || grid[i][j] == '0') {
        return;
    }

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

    for(int[]d : dir) {
        int ni = i + d[0];
        int nj = j + d[1];
        dfs(ni,nj,grid);
    }
}
```



# no. of distinct islands

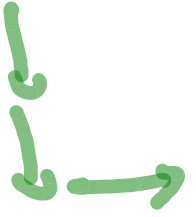
Str : ddrrrrr

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |

1 -> land

0 -> water

Str = ddrrrrr



String

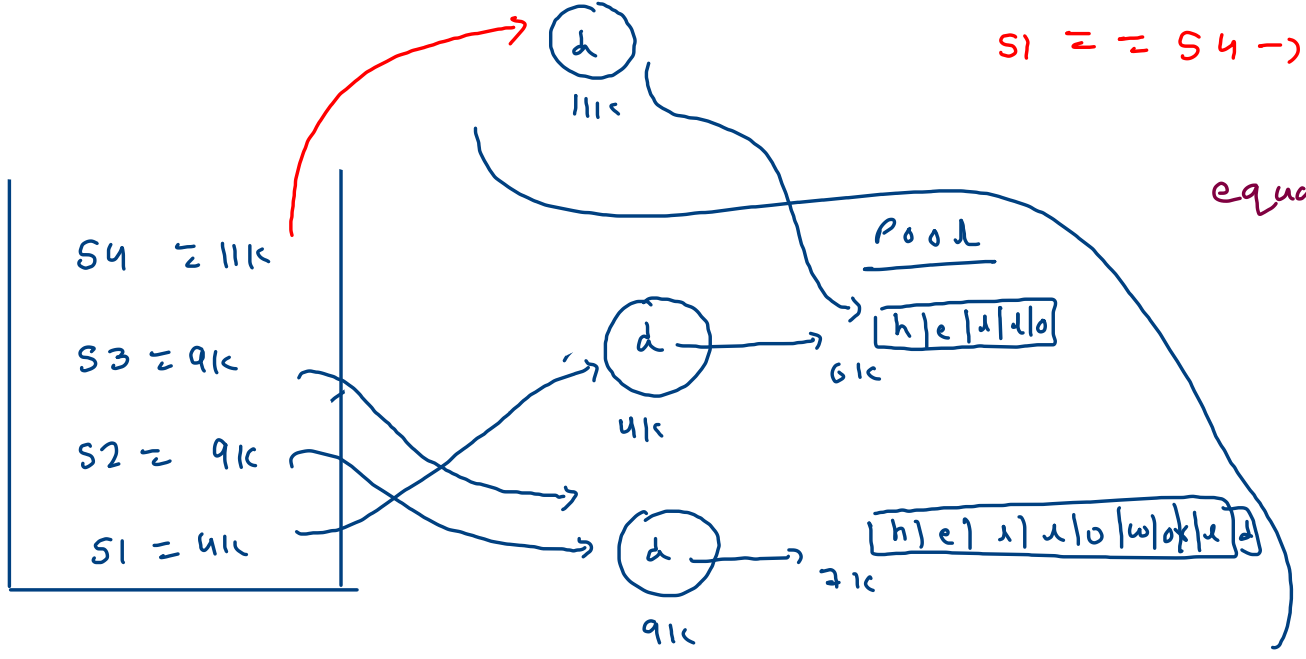
```
String s1 = "hello";
String s2 = "hello";
String s3 = s2;
String s4 = new String("hello");
```

s2 = s2 + "world";

String -> immutable

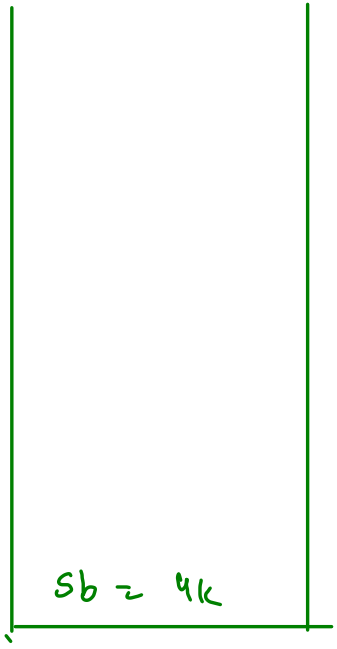
s1 == s4 -> false

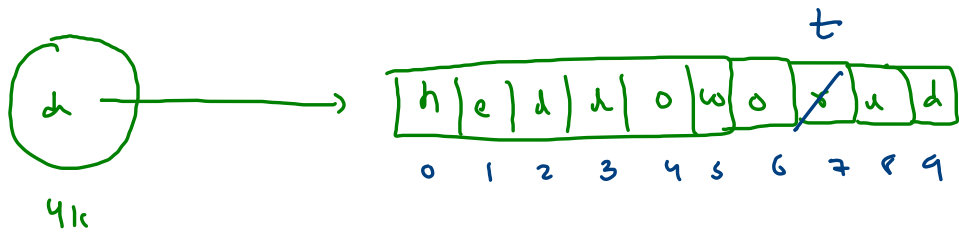
equals



StringBuilder
(ArrayList<char>)

StringBuilder sb = new StringBuilder("hello");

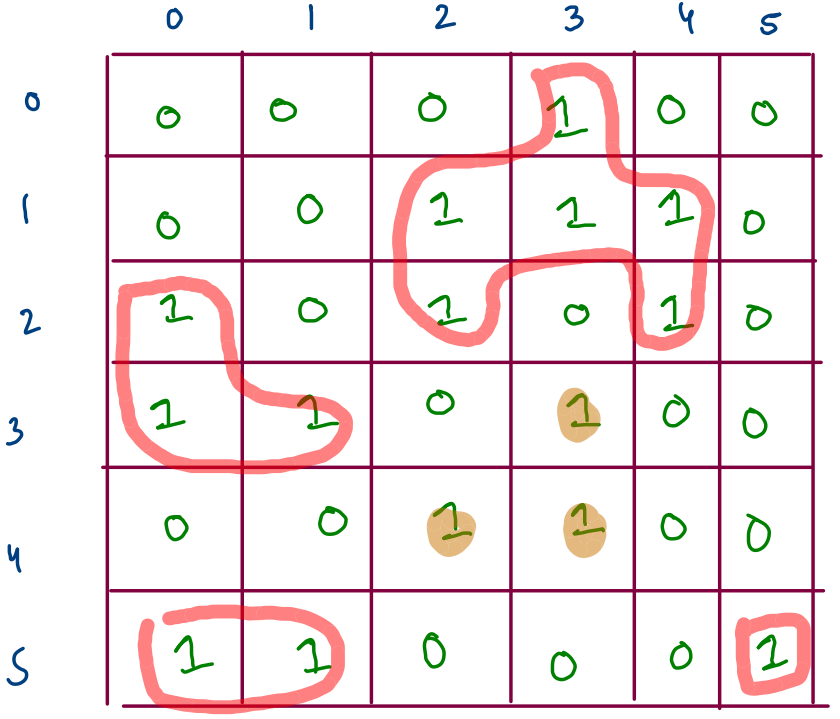


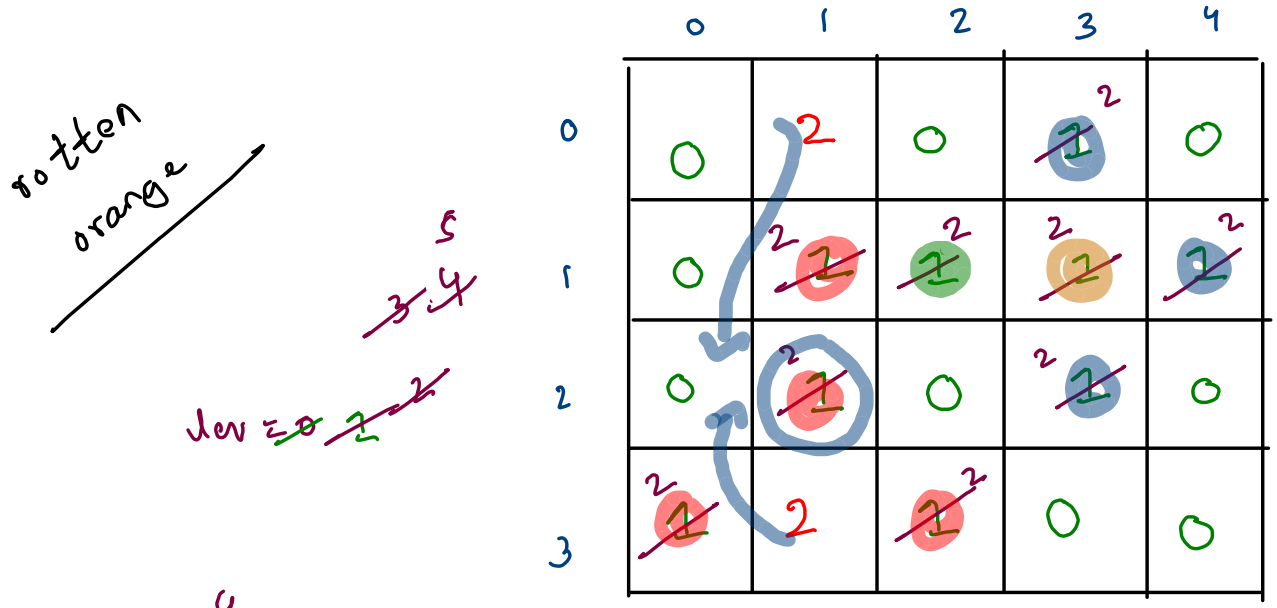




sb.append("world");

sb.setCharAt(7, 't')

no. of enclaves






multiple sources

0 -> empty

1 -> fresh orange

2 -> rotten orange


Count
- remove
- mark
- add nbr

```
while(q.size() > 0) {
    int count = q.size();

    if(fo == 0) {
        return lev-1;
    }

    while(count-- > 0) {
        //remove
        Pair rem = q.remove();

        //mark
        if(lev != 0 && grid[rem.i][rem.j] == 2) {
            continue;
        }

        if(grid[rem.i][rem.j] == 1) {
            fo--;
        }

        grid[rem.i][rem.j] = 2;


        //add nbr
        int ri = rem.i;
        int rj = rem.j;

        if(ri - 1 >= 0 && grid[ri-1][rj] == 1) {
            q.add(new Pair(ri-1,rj));
        }

        if(rj - 1 >= 0 && grid[ri][rj-1] == 1) {
            q.add(new Pair(ri,rj-1));
        }

        if(ri + 1 < grid.length && grid[ri+1][rj] == 1) {
            q.add(new Pair(ri+1,rj));
        }

        if(rj + 1 < grid[0].length && grid[ri][rj+1] == 1) {
            q.add(new Pair(ri,rj+1));
        }
    }

    lev++;
}
```

0

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

fo = ~~12 11 10 9 8 7 6 5 4 3 2 1 0~~

lev = ~~0 1 2~~ 3

return 2

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | ~~1~~ 2 | 0 | 0 | 0 |
| 1 | ~~1~~ 2 | 2 | ~~1~~ 2 | ~~1~~ 2 | 0 |
| 2 | ~~1~~ 2 | ~~1~~ 2 | ~~1~~ 2 | ~~1~~ 2 | 0 |
| 3 | 2 | 0 | 0 | ~~1~~ 2 | 0 |
| 4 | ~~1~~ 2 | ~~1~~ 2 | 0 | 2 | ~~1~~ 2 |

| ~~2,1~~ | ~~3,0~~ | ~~4,3~~ | ~~0,1~~ | ~~1,0~~ | ~~2,1~~ | ~~1,2~~ | ~~2,0~~ | ~~4,0~~ | ~~3,3~~ | ~~4,4~~ | ~~2,0~~ | ~~2,0~~ | ~~2,2~~ | ~~1,3~~ | ~~2,2~~ | ~~4,1~~ | ~~2,3~~ | 2,3 | 2,3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|