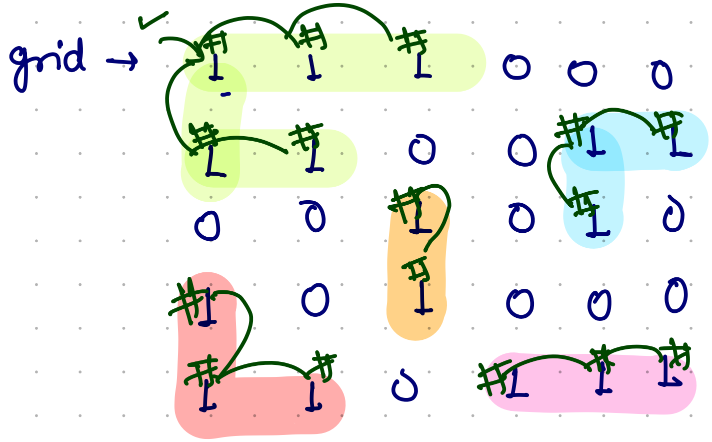
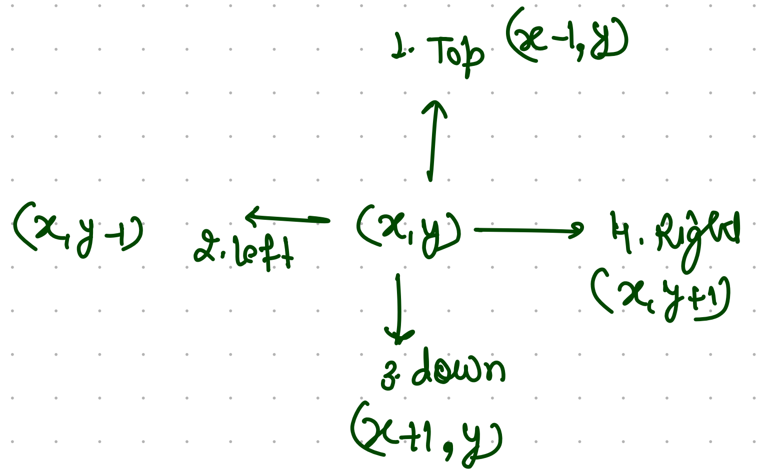
Number of Islands:



Reach → horizontal + vertical

dfs from a point ⟶ similar as flood fill



T L D R

NOTE: Do not unmark

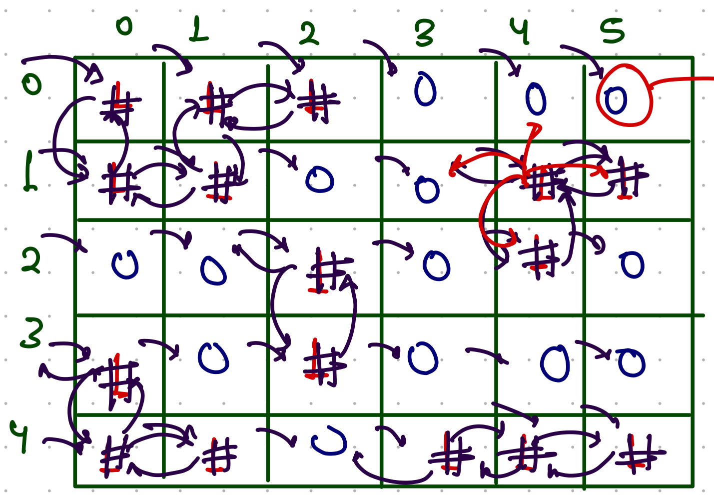
```java
static int[] xdir = {-1, 0, 1, 0};
static int[] ydir = {0, -1, 0, 1};

private void numIslandsComp(char[][] grid, int x, int y) {
    grid[x][y] = '#';
    // from x, y we can move top left down and right
    for(int d = 0; d < 4; d++) {
        int r = x + xdir[d];
        int c = y + ydir[d];

        if(r >= 0 && r < grid.length && c >= 0 &&
        c < grid[0].length && grid[r][c] == '1') {
            numIslandsComp(grid, r, c);
        }
    }
}

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++;
                numIslandsComp(grid, i, j);
            }
        }
    }
    return count;
}
```

out of Range check + invalid Reach

(x, y) using Loop

x dir → {-1, 0, 1, 0}

y dir → {0, -1, 0, 1}



count = 0 1 2 3 4 (5)

(0, 5)

(1, 4)

x = 1
y = 4

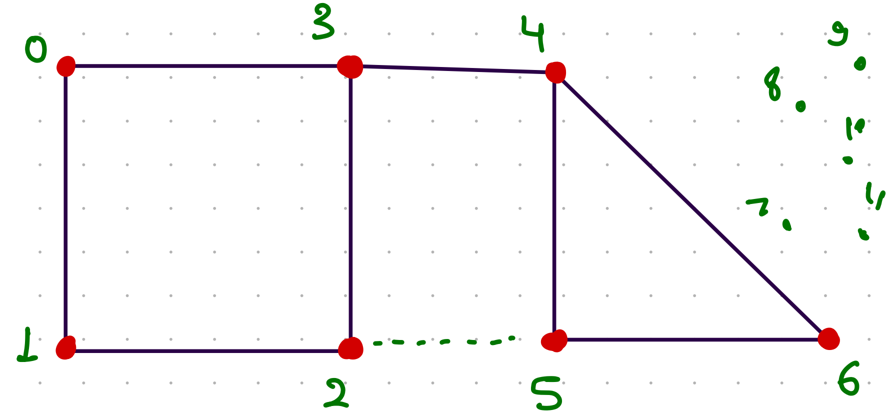
r = x + xdir[d], c = y + ydir[d]

T	d = 0	1 + (-1) = 0	4 + 0 = 4
L	= 1	1 + 0 = 1	4 + (-1) = 3
D	= 2	1 + 1 = 2	4 + 0 = 4
R	= 3	1 + 0 = 1	4 + 1 = 5

Hamiltonian path and cycle:-

Hamiltonian path: If in a graph we can visit all the vertices without visiting any vertex twice then that path is known as Hamiltonian path.

Hamiltonian cycle: In a hamiltonian path, if there is a back Edge from last point to src point then that path is hamiltonian cycle.



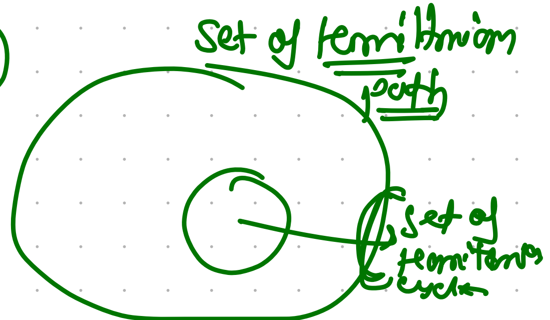
6→5→4→3→0→1→2
6→5→4→3→2→1→0
5→6→4→3→0→1→2
5→6→4→3→2→1→0
2→1→0→3→4→5→6
2→1→0→3→4→6→5
0→1→2→3→4→6→5
0→1→2→3→4→5→6

Hamiltonian path

5→6→4→3→0→1→2 (*) cycle,

"0123456" → length (7)

0 1 2 3 4 5 6 7 8 3 (10)

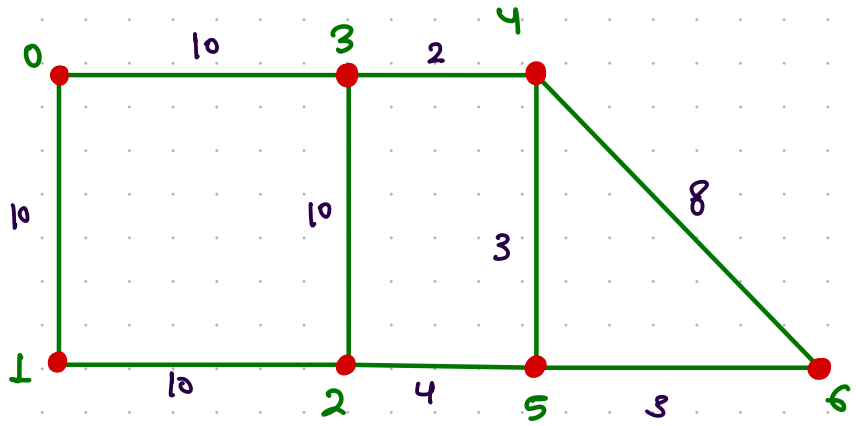


```
[0] -> [0-1@10], [0-3@40],
[1] -> [1-0@10], [1-2@10],
[2] -> [2-1@10], [2-3@10], [2-5@4],
[3] -> [3-0@40], [3-2@10], [3-4@2],
[4] -> [4-3@2], [4-5@3], [4-6@8],
[5] -> [5-4@3], [5-6@3], [5-2@4],
[6] -> [6-4@8], [6-5@3],
6430125*
6452103.
6452301.
6543012.
6543210.
6521034*
```

cyclic path

path

cyclic path



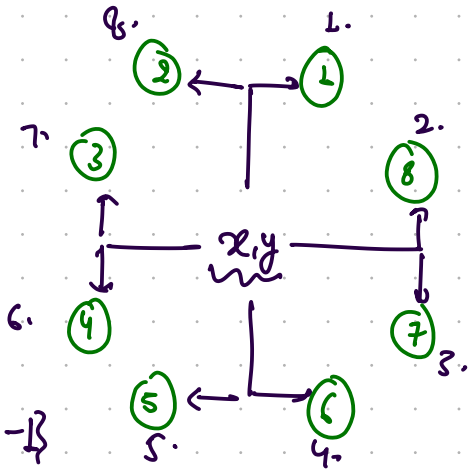
src = 6

Knights Tour:

n = 5.

	0	1	2	3	4
0		17	12	3	
1	13	2	5		
2	18	7	14	11	4
3	1	14	9	6	
4	8	19		15	10

initial position → (3,0)



	r	c
1.	x-2,	y+1 → ①
2.	x-2,	y-1 → ⑧
3.	x-1,	y-2 → ⑦
4.	x+1,	y-2 → ⑥
5.	x+2,	y-1 → ⑤
6.	x+2,	y+1 → ④
7.	x+1,	y+2 → ③
8.	x-1,	y+2 → ②

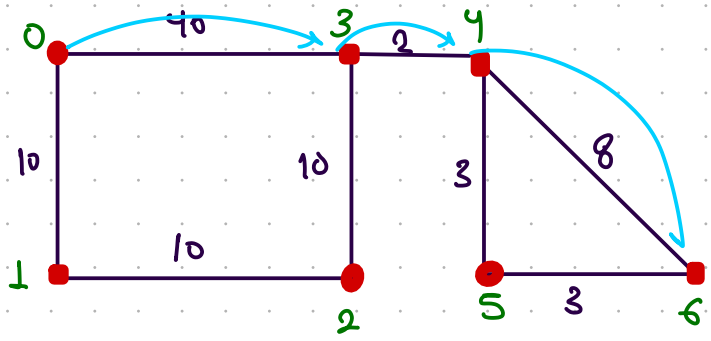
rdir → {-2, -2, -1, 1, 2, 2, 1, -1}

cdir → {1, -1, -2, -2, -1, 1, 2, 2}

rdir →	-2	-1	1	2	2	1	-1	-2
cdir →	1	2	2	1	-1	-2	-2	-1

BFS Algorithm :

↳ Breadth first search



Steps

1. Get + Remove
2. mark (*) → if marked → continue; otherwise marked & save
3. work → print
4. Add unvisited neighbours

vis →

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

In trees we can't move toward root

src.	psf.	wsf
0	" "	@ 0
1	"0"	@ 10
3	"0"	@ 40
2	"01"	@ 20
4	"03"	@ 42
5	"034"	@ 45
6	"034"	@ 50

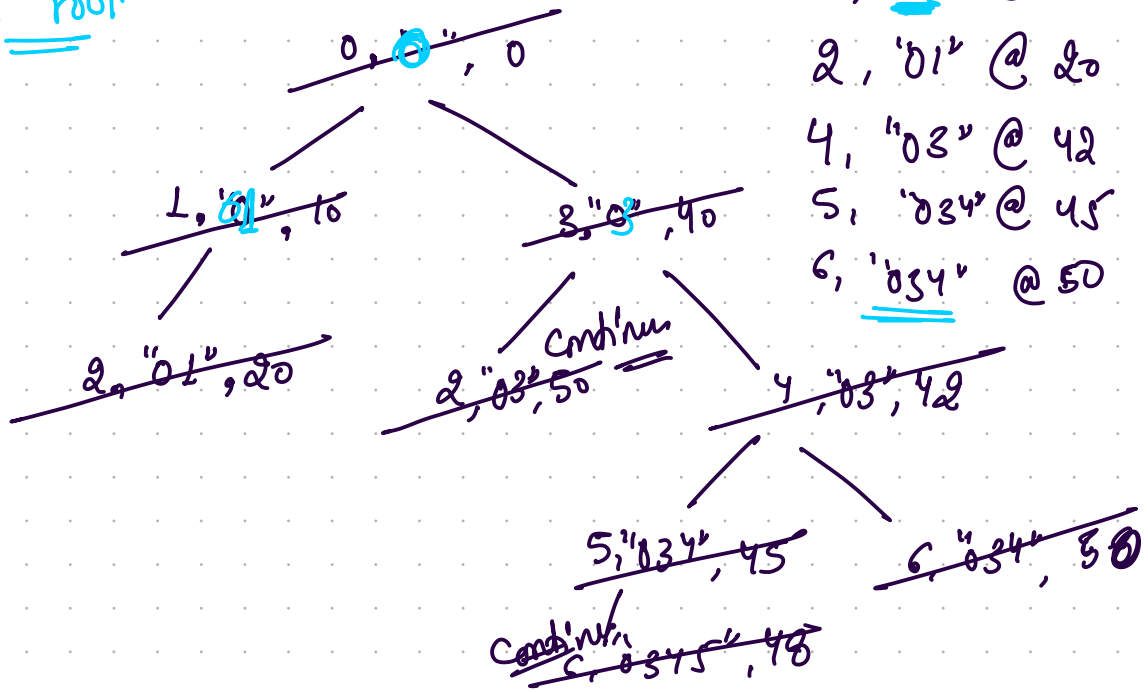
D.S used → Queue
data structure

Important → BFS moves Radially.

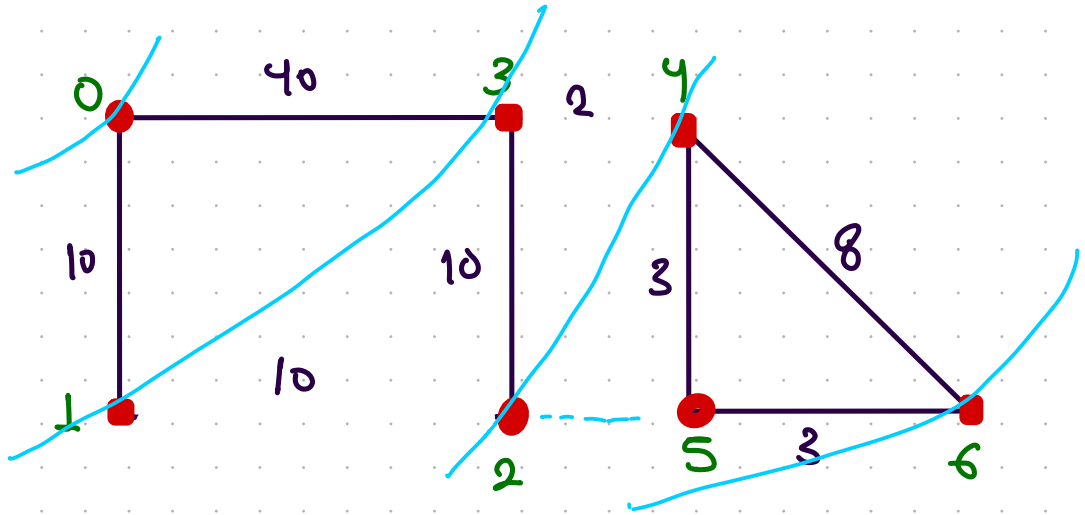
Shortest path in terms of Edge :=

0 1 3 2 4 5 6

src, psf, wsf



```
0@0-0
1@01-10
3@03-40
2@012-20
4@034-42
5@0125-24
6@0346-50
```



In BFS Algo.

we move Radially

Home work

{

① Number of island using BFS.

② Spread of Infection

↓

③ Rotten oranges