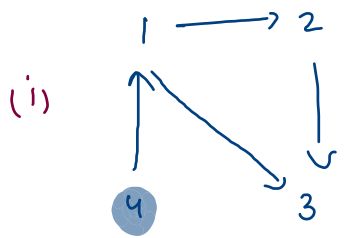
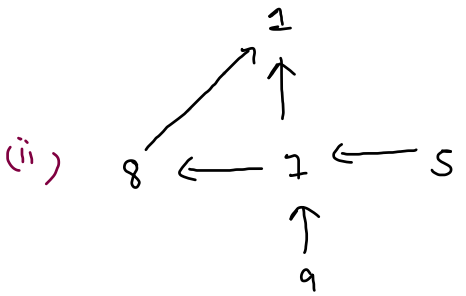
mother vertex

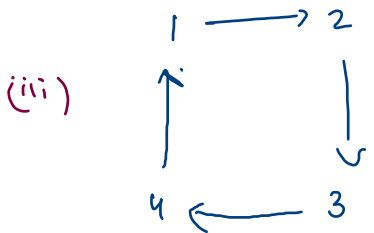
mother vertex : we can travel (direct or indirect) all the other vertices from a mother vertex



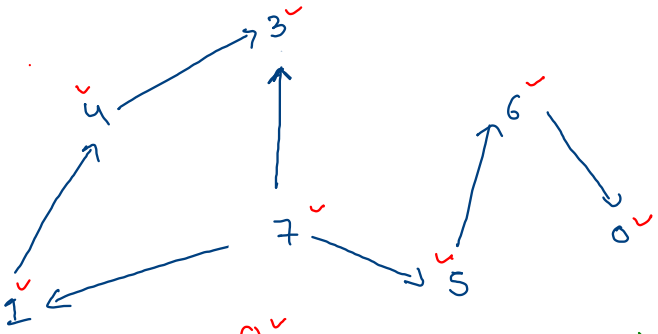
brute force: V^* (DFS)



[no mother vertex]



(multiple mother vertex)

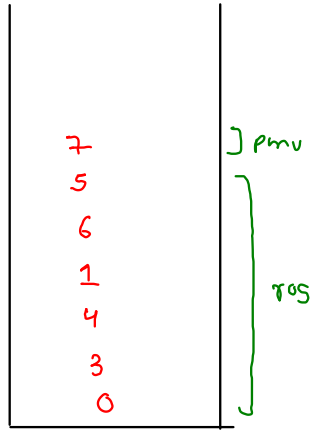


0 ✓
1 ✓
3 ✗
4 ✗
5 ✓
6 ✗
7 ✓

post → push vtx in stack

random DFS

DFS



pmv	ros	Algo
✓	✓	ans ✓
✗	✗	ans ✓
✓	✗	ans ✓
✗	✓	impossible

algo has 2 DFS : (i) to fill stack

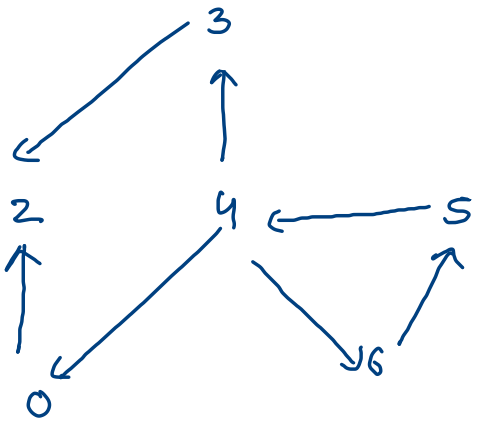
(ii) to check top most is matching vertex ;

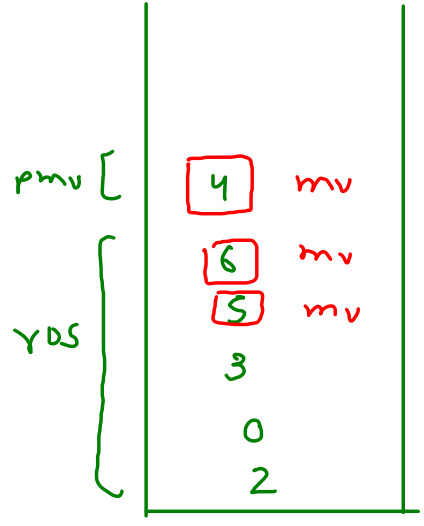
top most → mv ✓ → return ans ;

mv ✗ → return -1 ;

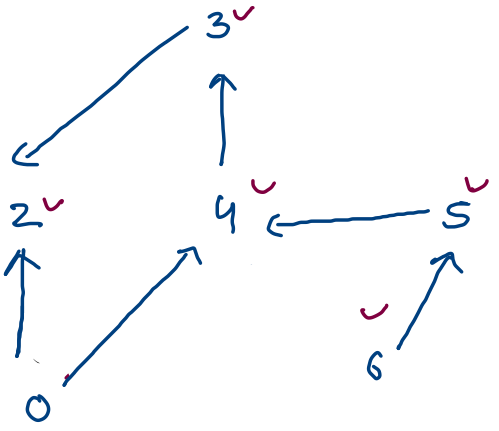
	pmv	rds	Algo
①	✓	✓	ans ✓
②	✗	✗	ans ✓
③	✓	✗	ans ✓
④	✗	✓	impossible

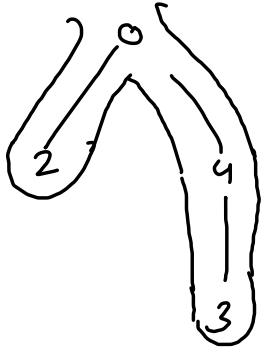
① .

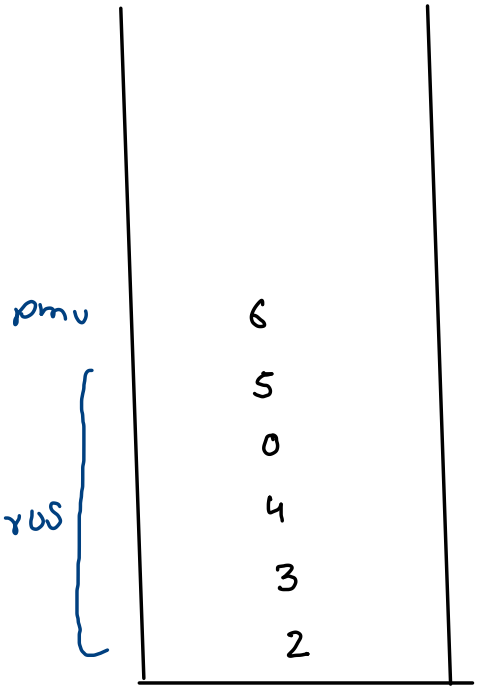




(1)



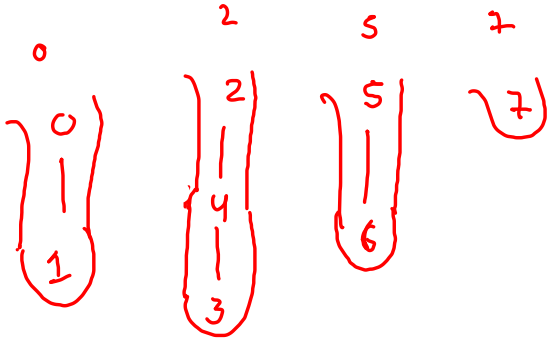


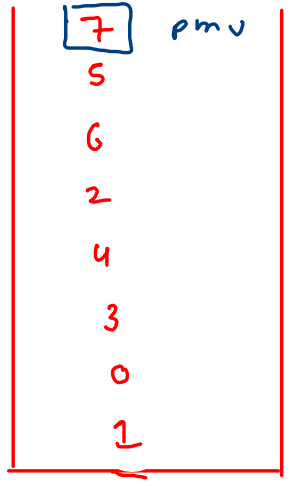


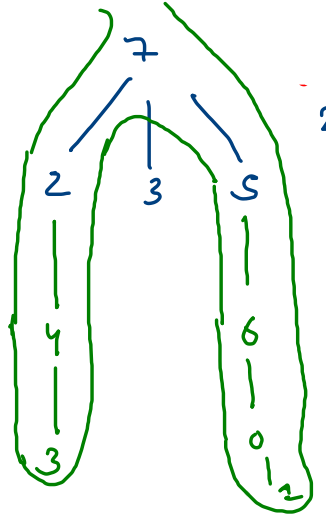
```java
//Function to find a mother vertex in the Graph.
public int findMotherVertex(int V, ArrayList<ArrayList<Integer>>adj)
{
    // Code here

    //1. to fill the stack
    int v = adj.size();
    Stack<Integer>st = new Stack<>();
    boolean[]vis = new boolean[v];

    for(int i=0; i < v;i++) {
        if(vis[i] == false) {
            dfs(adj,i,vis,st);
        }
    }


    //2. is pmv is actually a mother vertex or not
    int pmv = st.peek();
    vis = new boolean[v];
    dfs(adj,pmv,vis);

    for(int i=0; i < vis.length;i++) {
        if(vis[i] == false) {
            return -1;
        }
    }

    return pmv;
}
```

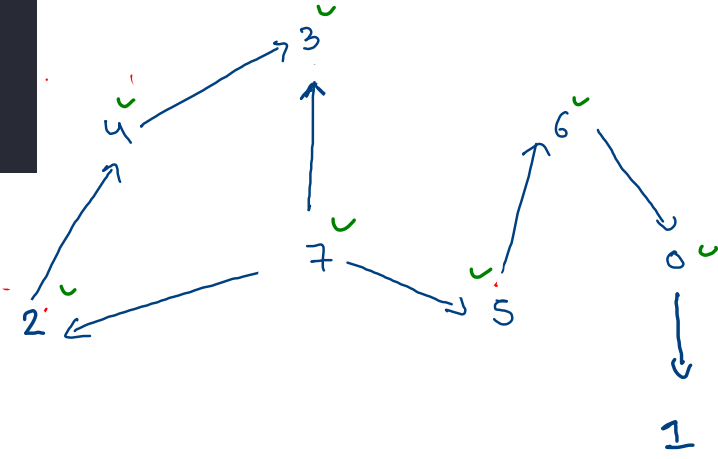
```java
public void dfs(ArrayList<ArrayList<Integer>>graph,int src,boolean[]vis,Stack<Integer>st) {

    vis[src] = true;

    for(int nbr : graph.get(src)) {
        if(vis[nbr] == false) {
            dfs(graph,nbr,vis,st);
        }
    }

    st.push(src);
}
```

```java
public void dfs(ArrayList<ArrayList<Integer>>graph,int src,boolean[]vis) {

    vis[src] = true;

    for(int nbr : graph.get(src)) {
        if(vis[nbr] == false) {
            dfs(graph,nbr,vis);
        }
    }


}
```



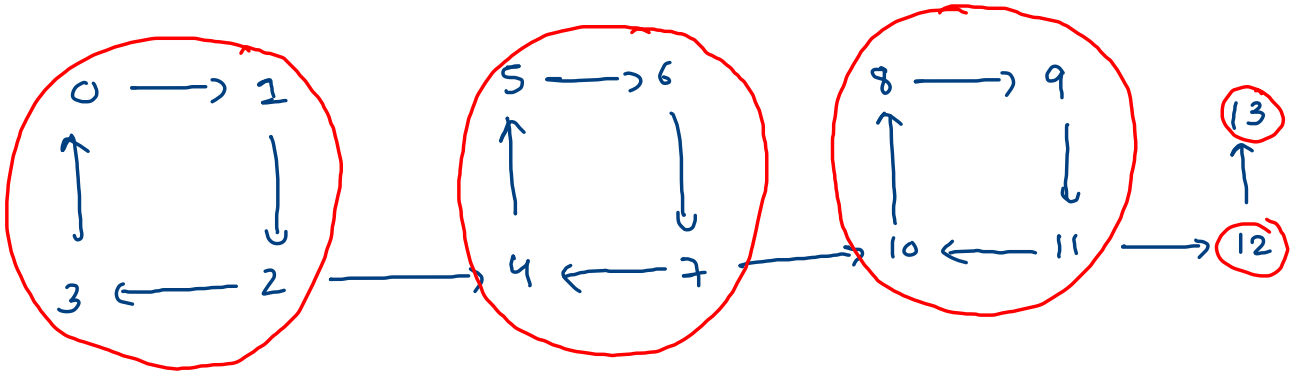


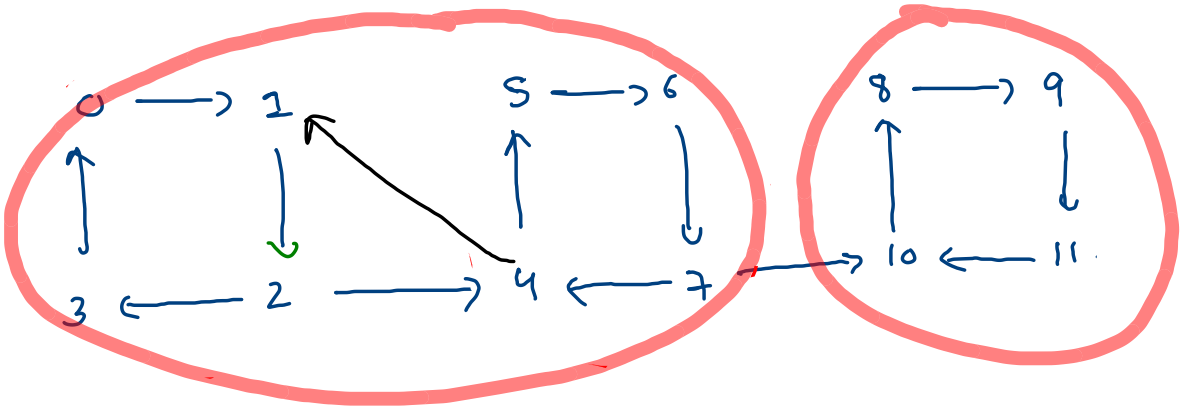




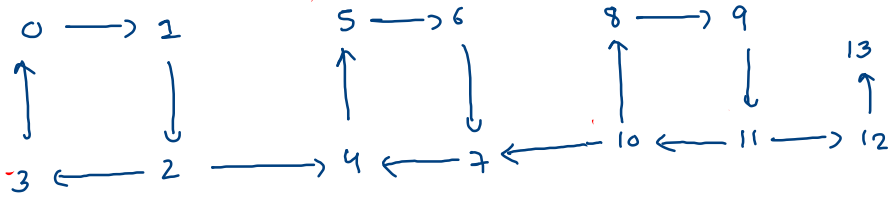
Kosaraju

no. of Strongly connected components



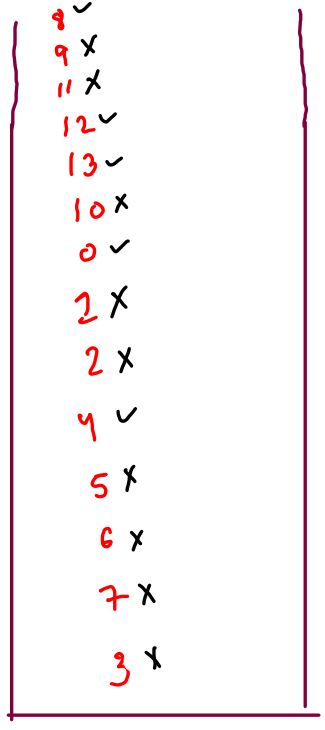


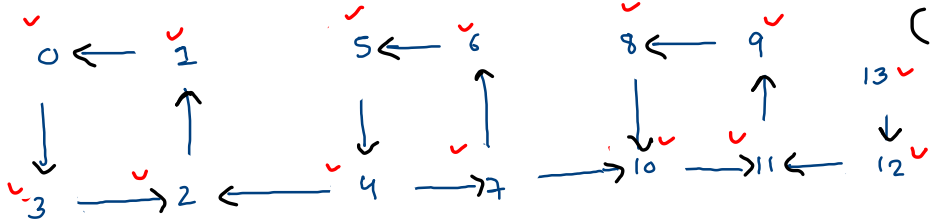
no. of cycle != no. of strongly connected components.

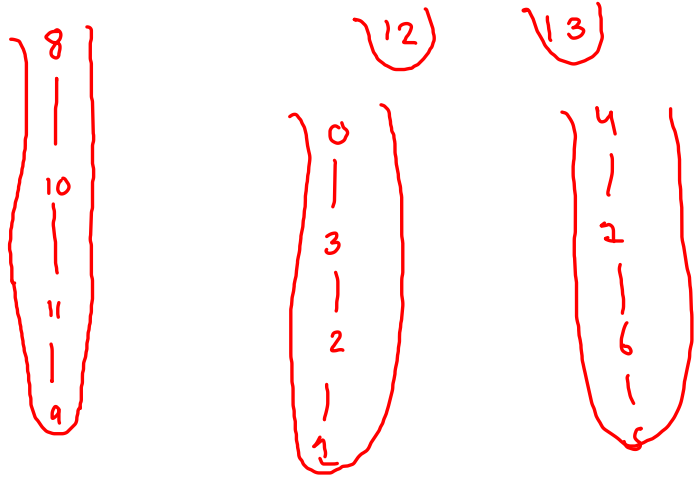


st.peek -> left vtx

scc = 1+1+1+1+1



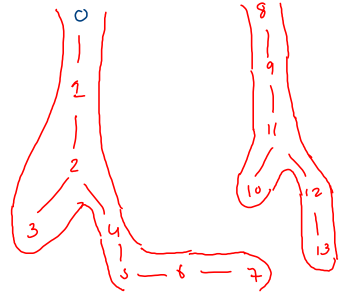




(i) fill stack, start dfs from left area's vtx.

(ii) reverse all edges

(iii) no. of dfs -> strongly connected comp.

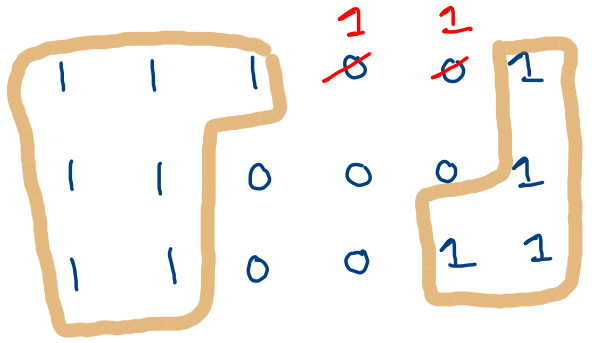


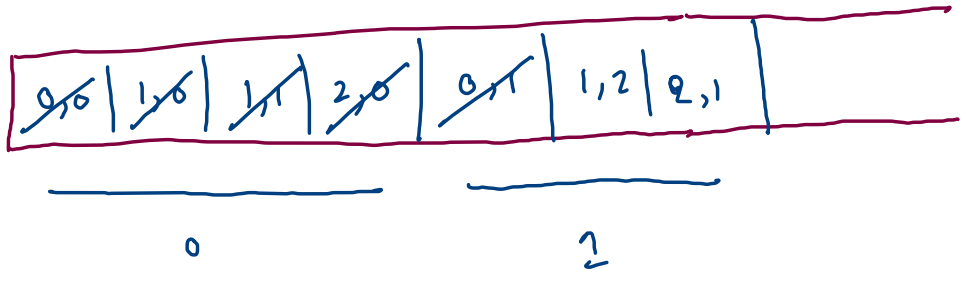
dfs order

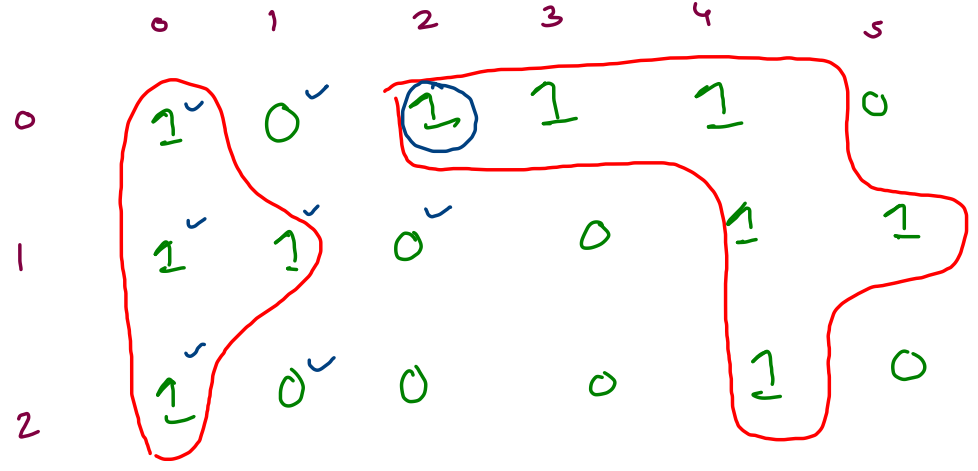
stack

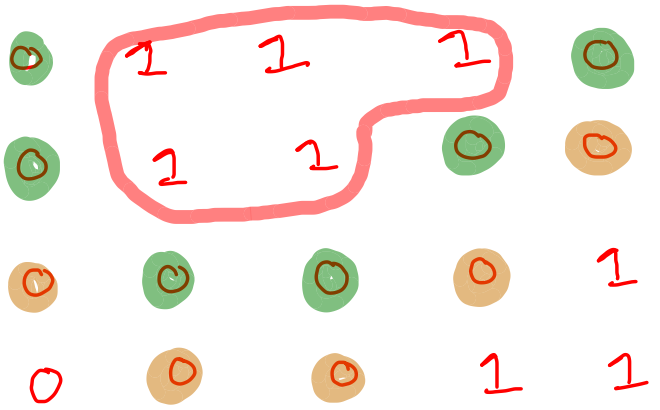
shortest bridge

(bfs application)









● -> deg 1

● -> deg 2

flag = ~~false~~ true

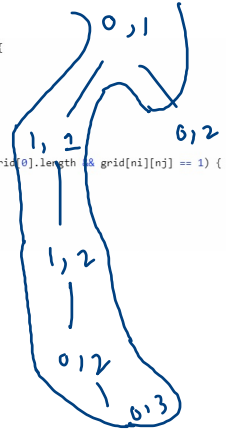
```
ArrayDeque<Pair>q = new ArrayDeque<>();
boolean flag = false;

for(int i=0; i < grid.length && flag == false ;i++) {
    for(int j=0; j < grid[0].length;j++) {
        if(grid[i][j] == 1) {
            dfs(grid,i,j,q);
            flag = true;
            break;
        }
    }
}
```

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

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

    while(count-- > 0) {
        //remove
        Pair rem = q.remove();
        int ri = rem.i;
        int rj = rem.j;

        //add nbr
        for(int i=0; i < 4;i++) {
            int ni = ri + dir[i][0];
            int nj = rj + dir[i][1];

            if(ni >= 0 && ni < grid.length && nj >= 0 && nj < grid[0].length && grid[ni][nj] != -1) {
                if(grid[ni][nj] == 1) {
                    return lev;
                }

                grid[ni][nj] = -1;
                q.add(new Pair(ni,nj));
            }
        }

    }

    lev++;
}
```

tldr

```
public void dfs(int[][]grid,int r,int c,ArrayDeque<Pair>q) {

    grid[r][c] = -1;
    q.add(new Pair(r,c));

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

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

}
```



new = ~~0~~ ~~1~~ 2

c = 5

~~0,1~~ ~~1,1~~ ~~1,2~~ ~~0,2~~ ~~0,3~~ | ~~0,0~~ | ~~1,0~~ | ~~2,1~~ | ~~1,5~~ | ~~2,2~~ | ~~0,4~~ | ~~2,0~~ | ~~3,1~~ | ~~1,4~~ | 2,3 | 3,2 | 3,0

2