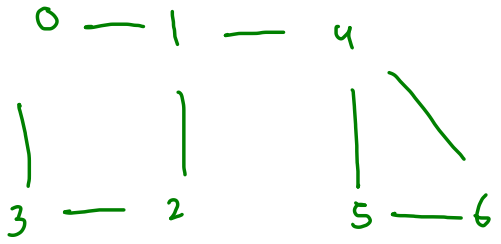
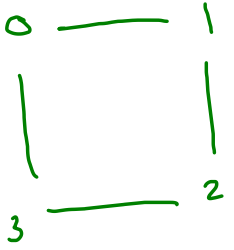
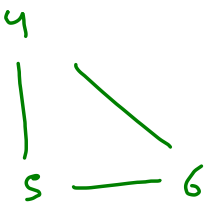
get connected comps



[[0,1,2,3,4,5,6]]



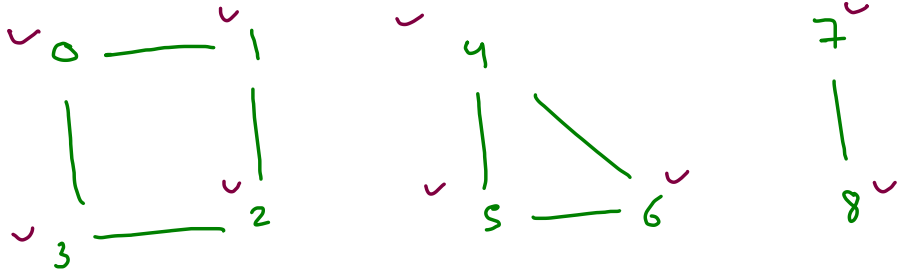
c1



c2

c3

[[0,1,2,3], [4,5,6], [7,8]]



0 → scc (0, vis, list)

1 X

2 X

3 X

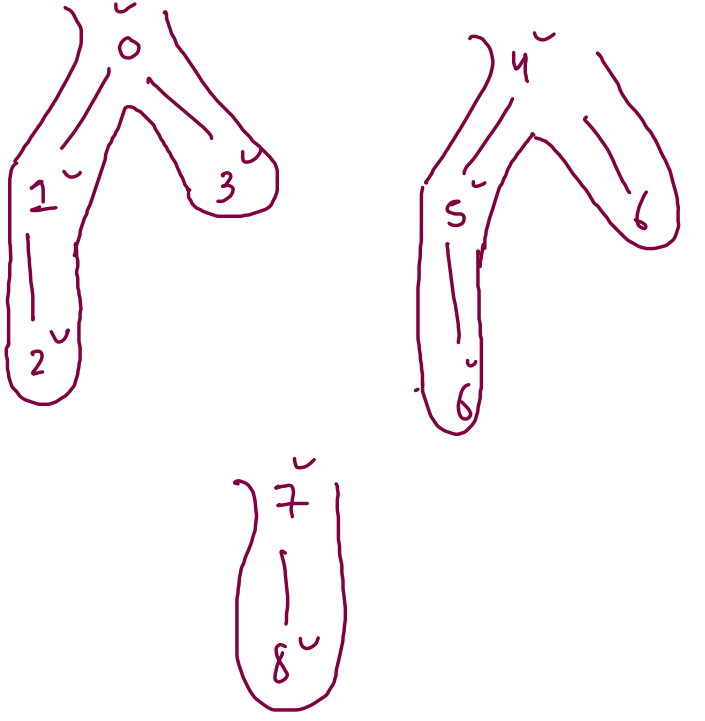
4 → scc (4, vis, list)

5 X

6 X

7 → scc (7, vis, list)

8 X



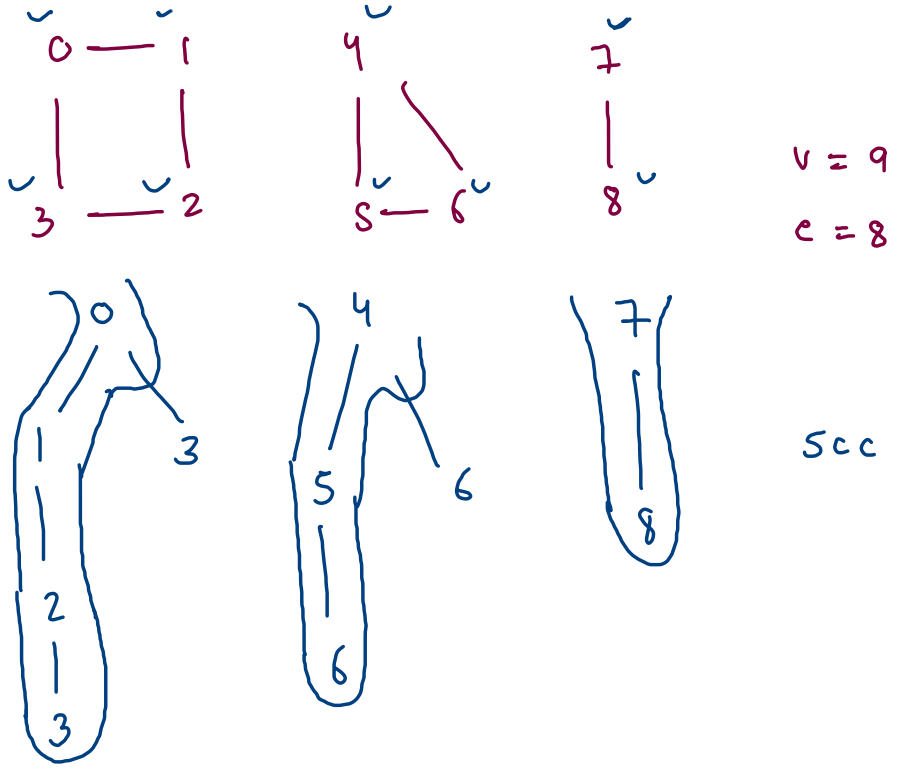
list

7, 8

ans [[0, 1, 2, 3], [4, 5, 6], [7, 8]]

```java
public static ArrayList<ArrayList<Integer>> getConnectedComp(ArrayList<Edge>[]graph) {
    ArrayList<ArrayList<Integer>>cc = new ArrayList<>();

    boolean[]vis = new boolean[graph.length];

    for(int i = 0; i < graph.length;i++) {
        if(vis[i] == false) {
            ArrayList<Integer>scc = new ArrayList<>();
            singleConnectedComp(i,graph,scc,vis);
            cc.add(scc);
        }
    }

    return cc;
}
```

```java
public static void singleConnectedComp(int src,ArrayList<Edge>[]graph, ArrayList<Integer>comp,boolean[]vis) {
    vis[src] = true;

    comp.add(src);

    for(Edge ne : graph[src]) {
        int nbr = ne.nbr;

        if(vis[nbr] == false) {
            singleConnectedComp(nbr,graph,comp,vis);
        }
    }
}
```



cc [4ik 9ik 3ik]

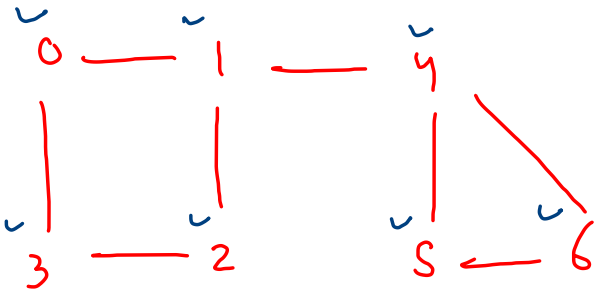
10k

0, 1, 2, 3

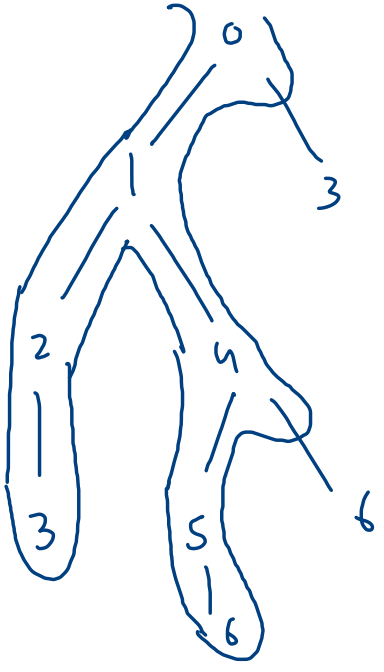
4, 5, 6

7, 8

```java
public static ArrayList<ArrayList<Integer>> getConnectedComp(ArrayList<Edge>[]graph) {
    ArrayList<ArrayList<Integer>>cc = new ArrayList<>();

    boolean[]vis = new boolean[graph.length];

    for(int i = 0; i < graph.length;i++) {
        if(vis[i] == false) {
            ArrayList<Integer>scc = new ArrayList<>();
            singleConnectedComp(i,graph,scc,vis);
            cc.add(scc);
        }
    }

    return cc;
}
```




```java
public static void singleConnectedComp(int src,ArrayList<Edge>[]graph, ArrayList<Integer>comp,boolean[]vis) {
    vis[src] = true;

    comp.add(src);

    for(Edge ne : graph[src]) {
        int nbr = ne.nbr;

        if(vis[nbr] == false) {
            singleConnectedComp(nbr,graph,comp,vis);
        }
    }
}
```

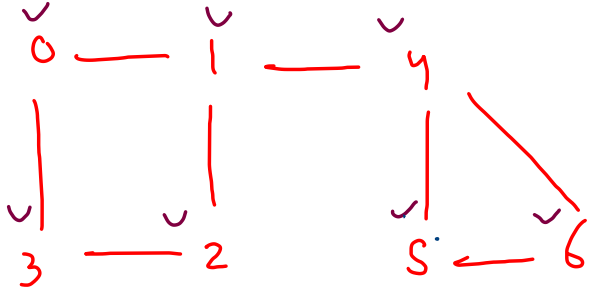


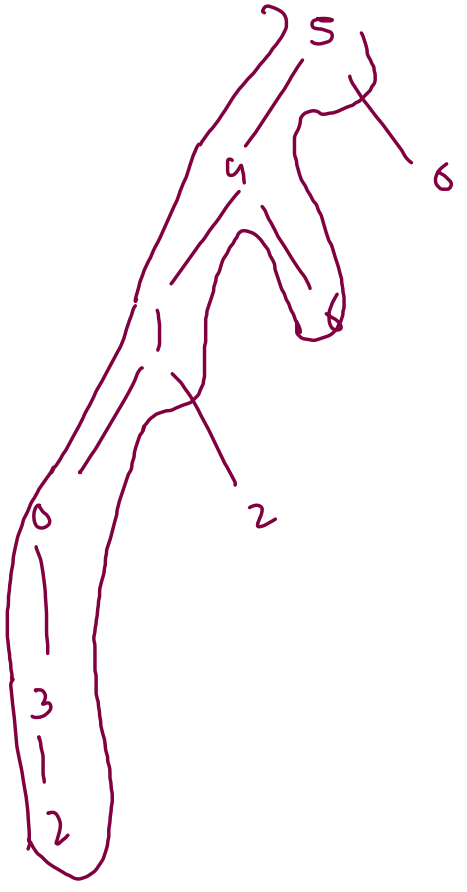
scc

V + E

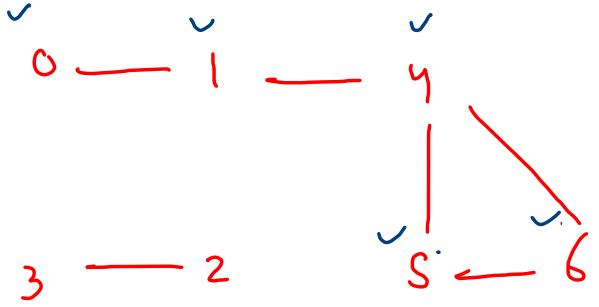
cc [[0, 2, 2, 3, 4, 5, 6]]

connected

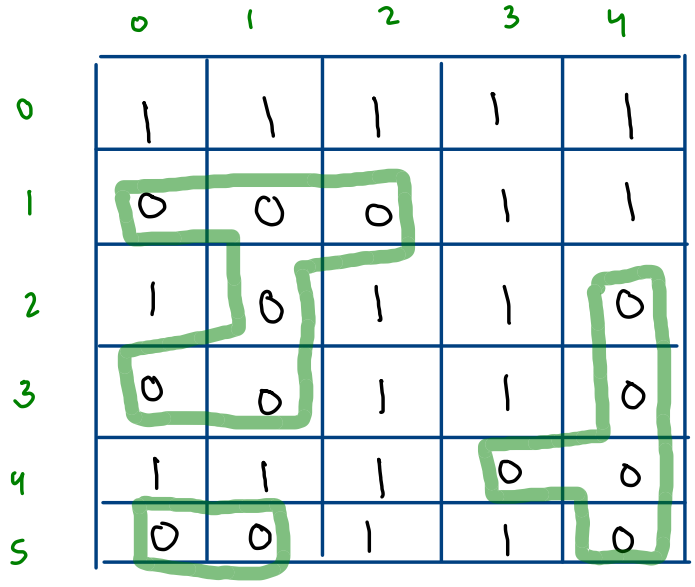




disconnected



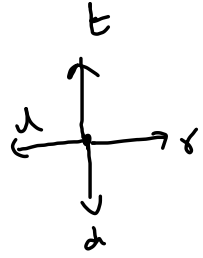
1. You are given a 2d array where 0's represent land and 1's represent water.
   Assume every cell is linked to it's north, east, west and south cell.
2. You are required to find and count the number of islands.



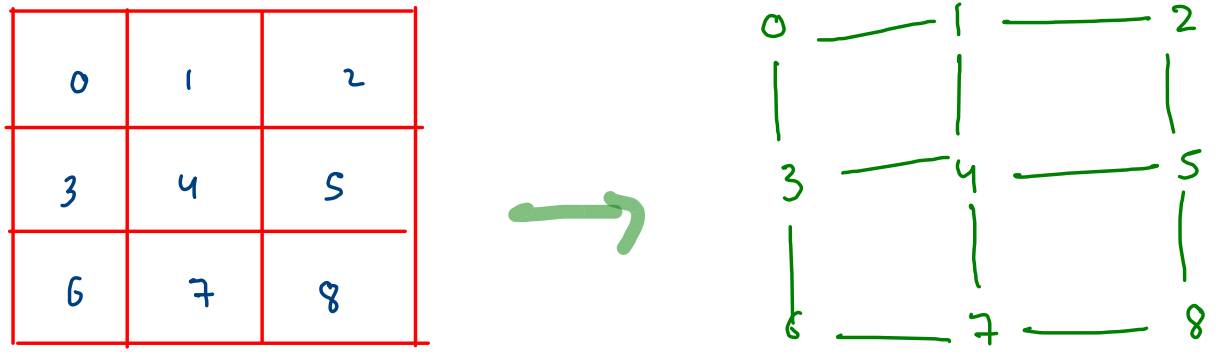
0 -> land

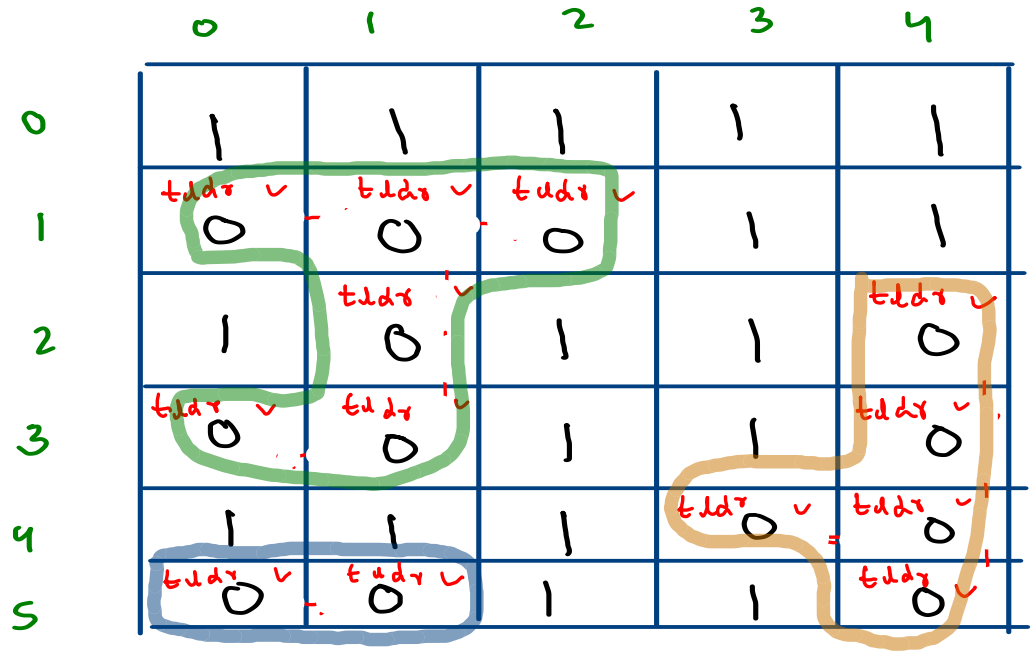
1 -> water

how many islands?



1. You are given a 2d array where 0's represent land and 1's represent water.
   Assume every cell is linked to it's north, east, west and south cell.
2. You are required to find and count the number of islands.





tldr

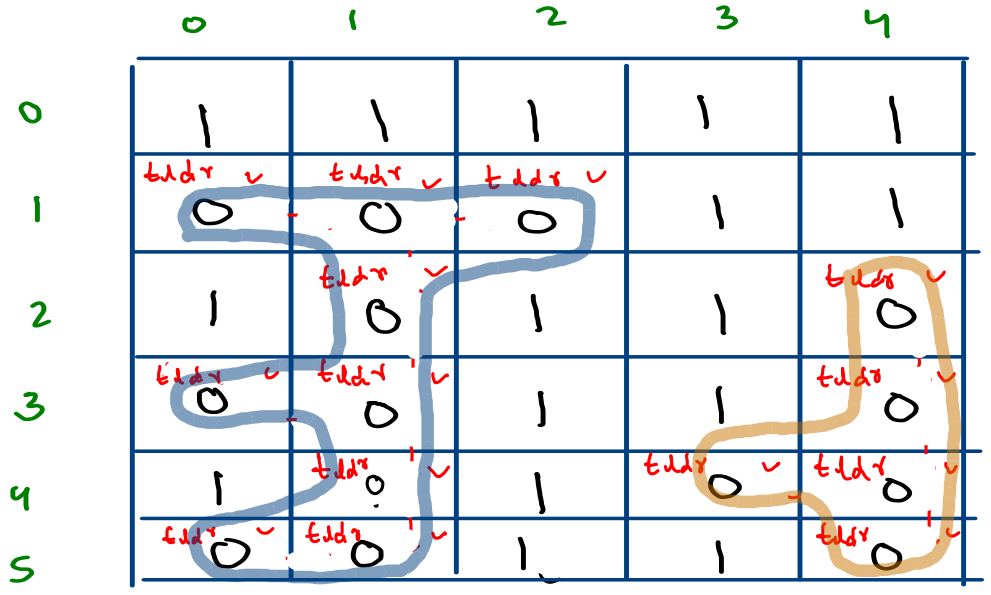
valid unvisited vertex
(an unvisited 0)

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

(1,0) → dfs

(2,4) → dfs

(3,0) → dfs




```
//0 -> land, 1 -> water
for(int i=0; i < n;i++) {
    for(int j=0; j < m;j++) {

        //valid unvisited vertex -> an unvisited 0
        if(mat[i][j] == 0 && vis[i][j] == false) {
            count++;
            dfs(i,j,mat,vis);
        }

    }
}
```

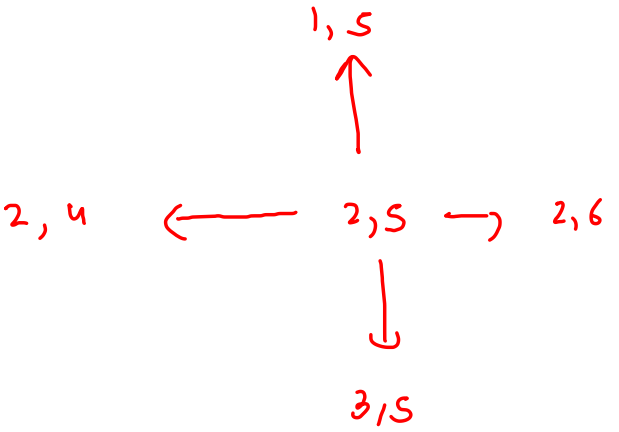
```
public static void dfs(int sr,int sc,int[][]mat,boolean[][]vis) {
    if(sr < 0 || sr >= mat.length || sc < 0 || sc >= mat[0].length || mat[sr][sc] == 1 || vis[sr][sc] == true) {
        return;
    }

    vis[sr][sc] = true;

    //top
    dfs(sr-1,sc,mat,vis);

    //left
    dfs(sr,sc-1,mat,vis);

    //down
    dfs(sr+1,sc,mat,vis);

    //right
    dfs(sr,sc+1,mat,vis);
}
```

count = ~~0~~ 2

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

t l d r

```
for(int i=0; i < dir.length;i++) {
    int nr = sr + dir[i][0];
    int nc = sc + dir[i][1];

    if(nr >= 0 && nr < mat.length && nc >= 0 && nc < mat[0].length && mat[nr][nc] == 0 && vis[nr][nc] == false) {
        dfs(nr,nc,mat,vis);
    }
}
```

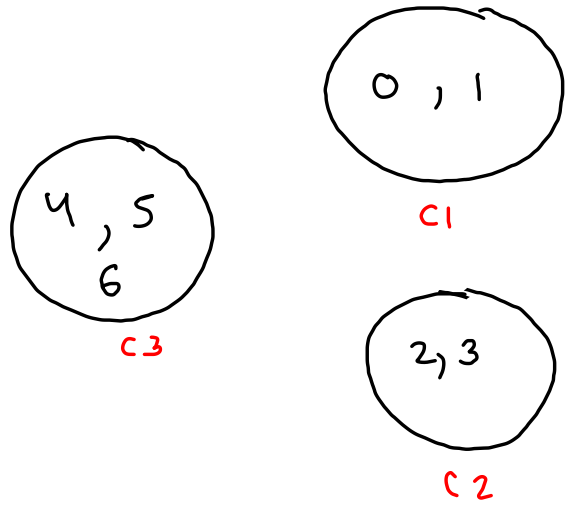


sr = 2

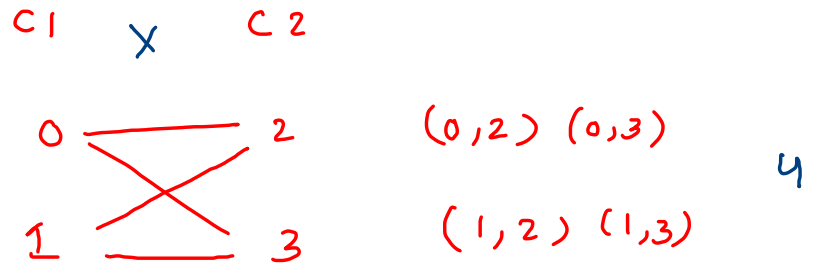
sc = 5

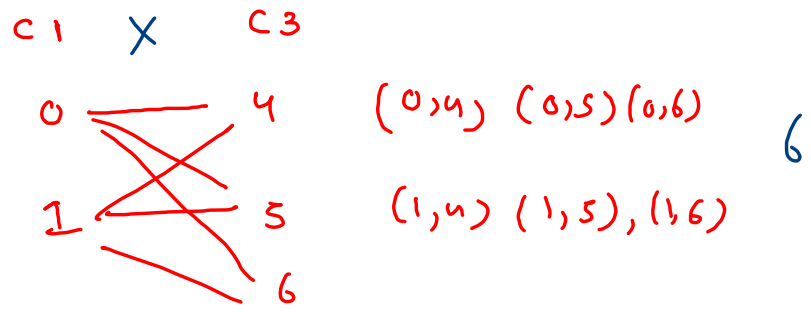
1. You are given a number n (representing the number of students). Each student will have an id from 0 to n - 1.
2. You are given a number k (representing the number of clubs)
3. In the next k lines, two numbers are given separated by a space. The numbers are ids of students belonging to same club.
4. You have to find in how many ways can we select a pair of students such that both students are from different clubs.

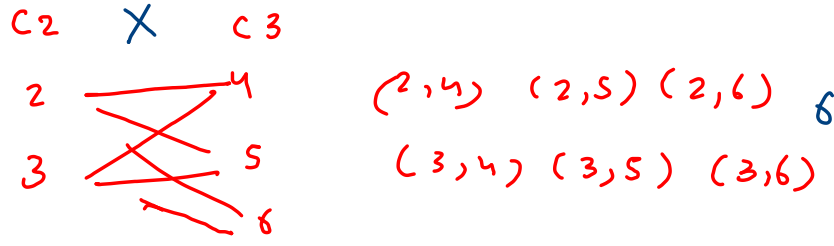
```
7
5
0 1
2 3
4 5
5 6
4 6
```

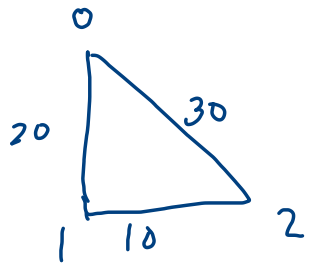


ways = 16









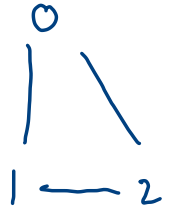
src

0 -> 1 @ 20, 2 @ 30

1 -> 0 @ 20, 2 @ 10

2 -> 1 @ 10, 0 @ 30

```
Edge {
    int nbr;
    int wt;
}
```



0 -> 1, 2

1 -> 0, 2

2 -> 1, 0

v = 7

e = 5

ArrayList<Int> [] graph;

```
7
5
0 1
2 3
4 5
5 6
4 6
```

0 -> 1

1 -> 0

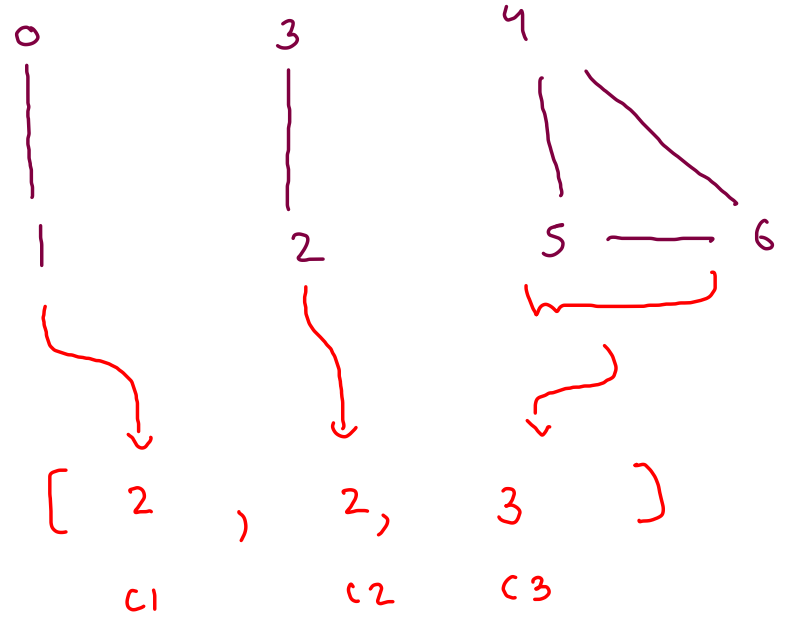
2 -> 3

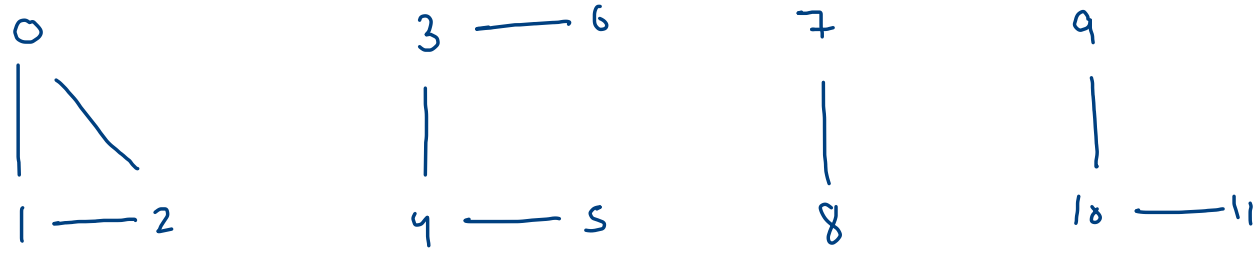
3 -> 2

4 -> 5, 6

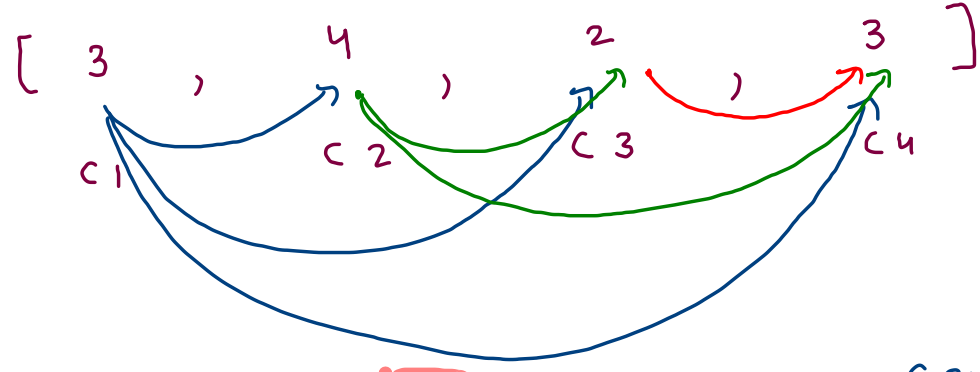
5 -> 4, 6

6 -> 5, 4





AL :



c1 x c2 = 3 x 4 = 12

c1 x c3 = 3 x 2 = 6

c1 x c4 = 3 x 3 = 9

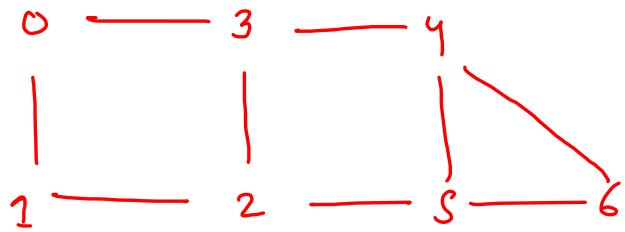
c2 x c3 = 4 x 2 = 8

c2 x c4 = 4 x 3 = 12

c3 x c4 = 2 x 3 = 6

12 + 6 + 9 + 8 + 12 + 6

Hamiltonian path and cycle.



src = 1

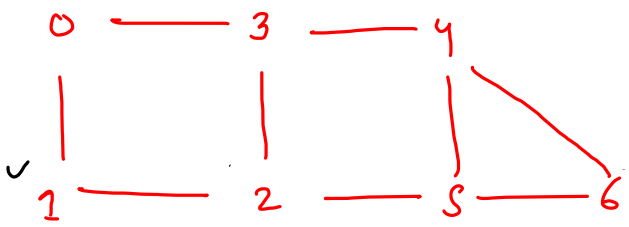
hamiltonian paths and cycle

1 0 3 2 5 4 6.

1 0 3 2 5 6 4.

1 0 3 4 6 5 2*

1 2 5 6 4 3 0*




```
for(Edge ne : graph[osrc]) {
    int nbr = ne.nbr;

    if(nbr == src) {
        ishc = true;
    }
}
```

1 0 3 2 5 4 6.

1 0 3 2 5 6 4.

1 0 3 4 6 5 2 *

1 2 5 6 4 3 0 *

0 src = 1

src = 1