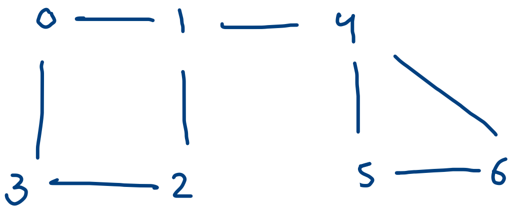
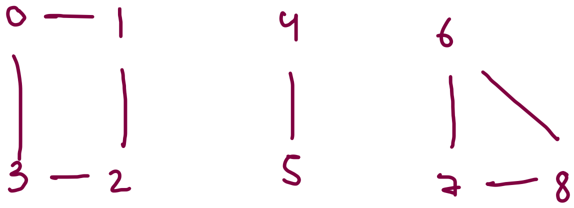
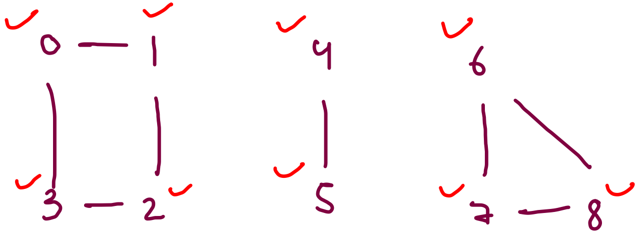
Get Connected Components Of A Graph



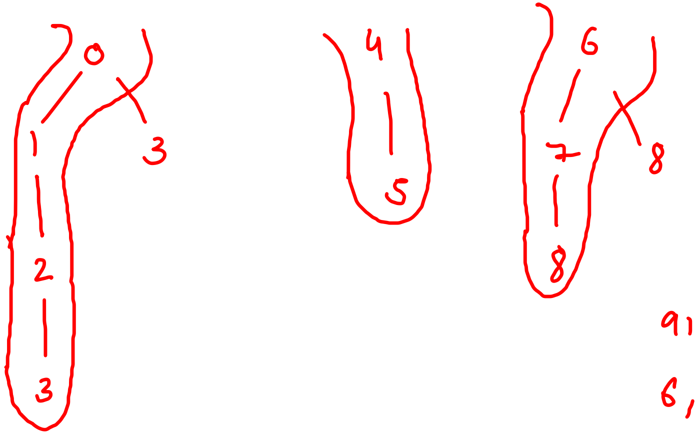


[[0, 1, 2, 3, 4, 5, 6]]

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



get single connected comp.



4k ←
0, 1, 2, 3

9k
6, 7, 8

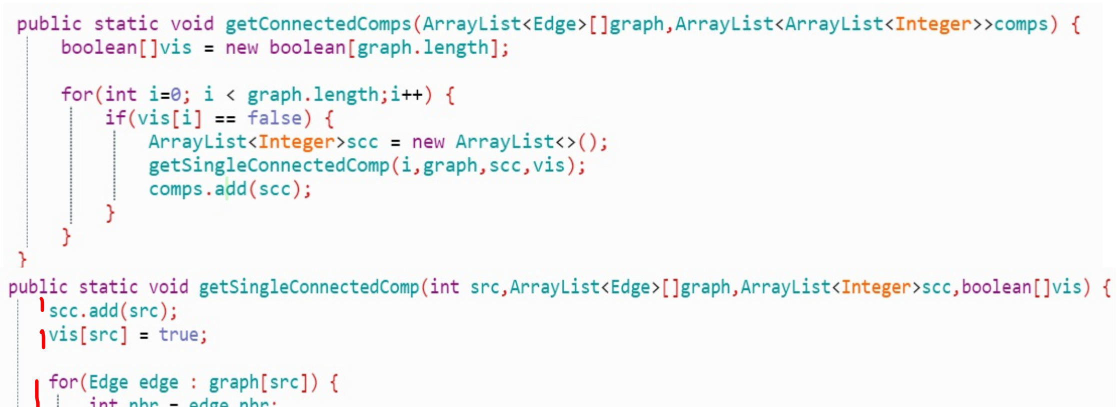
8k
4, 5

for loop

0 → gscc (0, vis, 4k)
1 → X
2 → X
3 → X
4 → gscc (4, vis, 8k)
5 → X
6 → gscc (6, vis, 9k)
7 → X
8 → X

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

DFS → v + E
(each vertex is vis only once)



```
public static void getConnectedComps(ArrayList<Edge>[]graph,ArrayList<ArrayList<Integer>>comps) {
    boolean[]vis = new boolean[graph.length];

    for(int i=0; i < graph.length;i++) {
        if(vis[i] == false) {
            ArrayList<Integer>scc = new ArrayList<>();
            getSingleConnectedComp(i,graph,scc,vis);
            comps.add(scc);
        }
    }
}
public static void getSingleConnectedComp(int src,ArrayList<Edge>[]graph,ArrayList<Integer>scc,boolean[]vis) {
    scc.add(src);
    vis[src] = true;

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

        if(vis[nbr] == false) {
            getSingleConnectedComp(nbr,graph,scc,vis);
        }
    }
}
```

4k

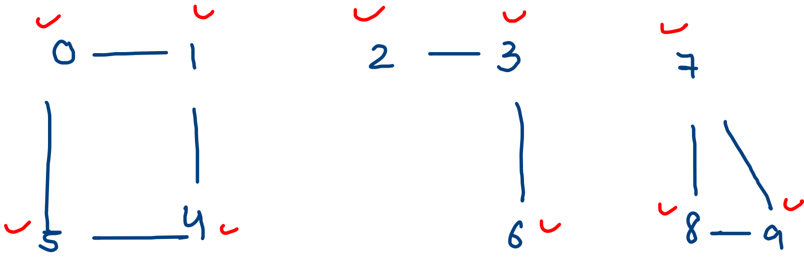
0, 1, 4, 5

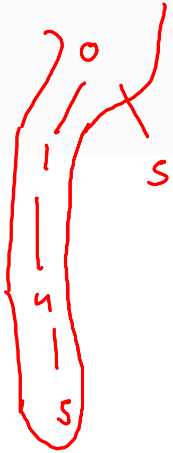
8k

2, 3, 6

9k

7, 8, 9





0 -> gscc(0, 4k, vis)

1 -> X

2 -> gscc(2, 8k, vis)

3 -> X

4 -> X

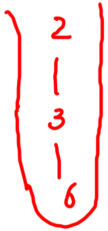
5 -> X

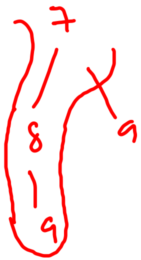
6 -> X

7 -> gscc(7, 9k, vis)

8 -> X

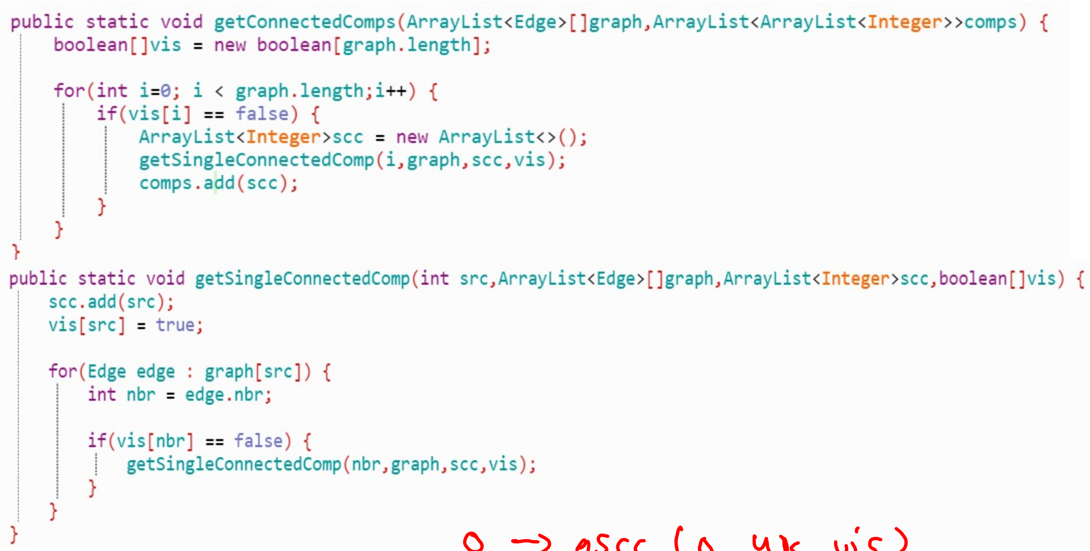
9 -> X



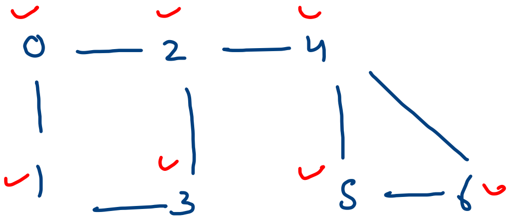


comps [4k, 8k, 9k]

[[0,1,4,5], [2,3,6], [7,8,9]]



```
public static void getConnectedComps(ArrayList<Edge>[]graph,ArrayList<ArrayList<Integer>>comps) {
    boolean[]vis = new boolean[graph.length];

    for(int i=0; i < graph.length;i++) {
        if(vis[i] == false) {
            ArrayList<Integer>scc = new ArrayList<>();
            getSingleConnectedComp(i,graph,scc,vis);
            comps.add(scc);
        }
    }
}
public static void getSingleConnectedComp(int src,ArrayList<Edge>[]graph,ArrayList<Integer>scc,boolean[]vis) {
    scc.add(src);
    vis[src] = true;

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

        if(vis[nbr] == false) {
            getSingleConnectedComp(nbr,graph,scc,vis);
        }
    }
}
```



0 -> gscc (0, 4k, vis)

1 -> X

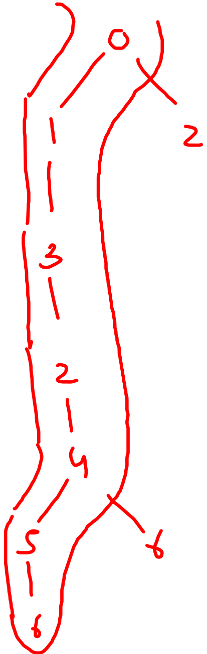
2 -> X

3 -> X

4 -> X

5 -> X

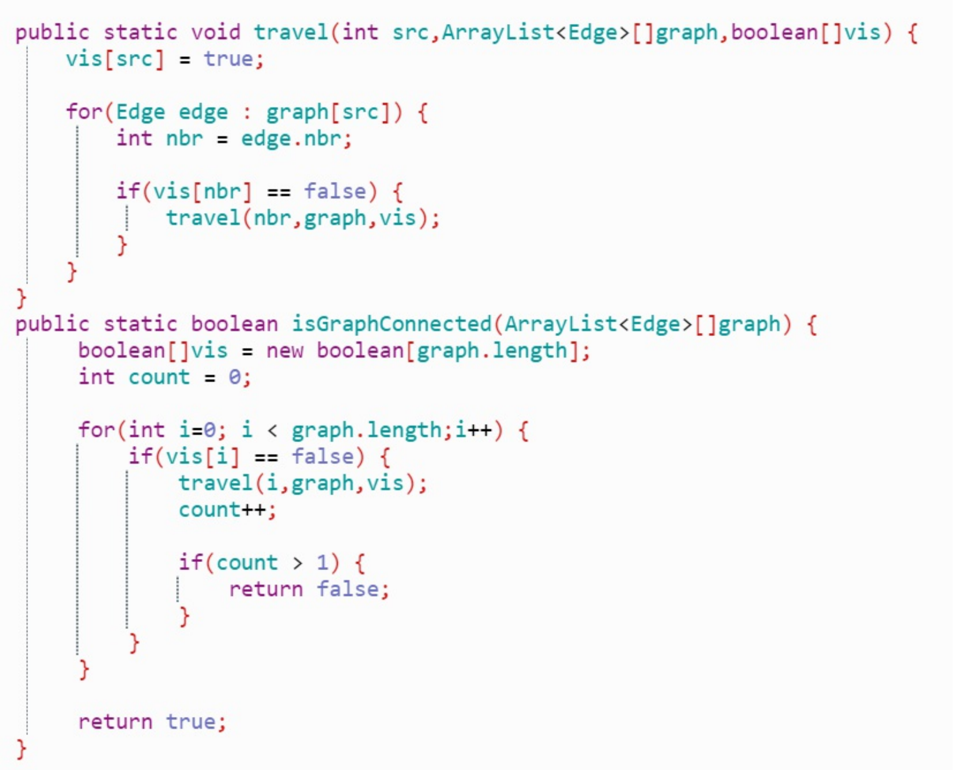
6 -> X



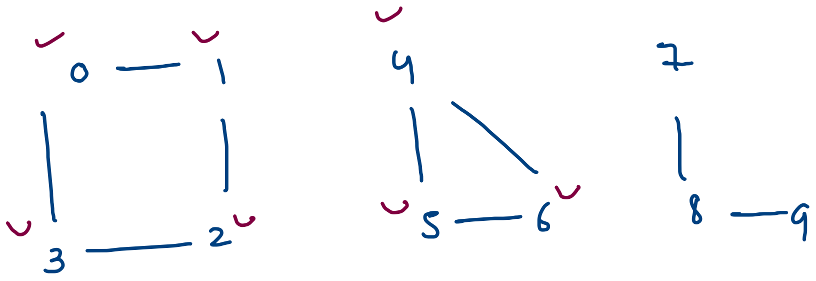
4k

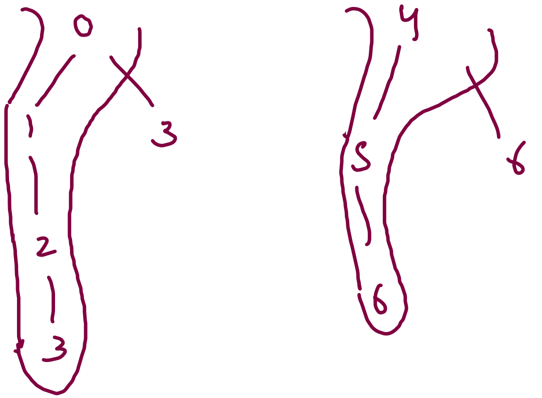
[[0, 1, 3, 2, 4, 5, 6]]

Is Graph Connected



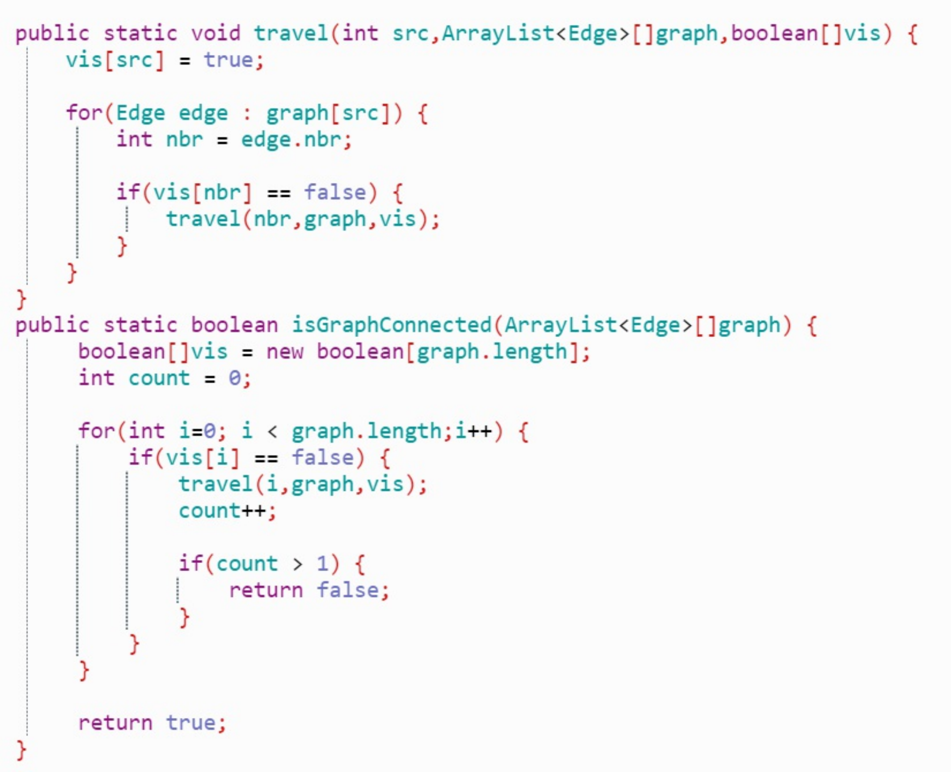
```
public static void travel(int src,ArrayList<Edge>[]graph,boolean[]vis) {
    vis[src] = true;

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

        if(vis[nbr] == false) {
            travel(nbr,graph,vis);
        }
    }
}
public static boolean isGraphConnected(ArrayList<Edge>[]graph) {
     boolean[]vis = new boolean[graph.length];
     int count = 0;

     for(int i=0; i < graph.length;i++) {
         if(vis[i] == false) {
             travel(i,graph,vis);
             count++;

             if(count > 1) {
                 return false;
             }
         }
     }

     return true;
}
```



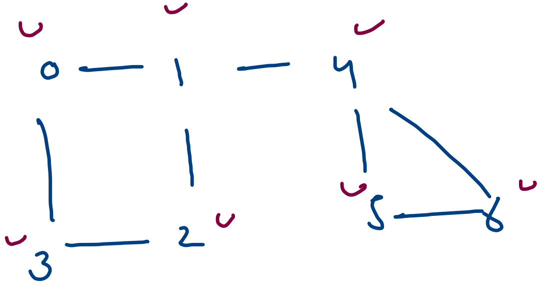


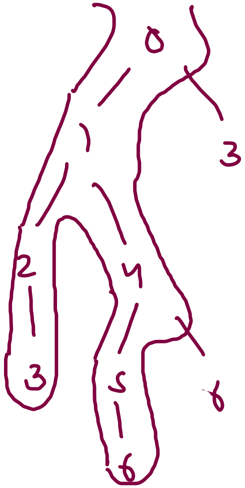
c = $\cancel{0}$ $\cancel{1}$

2



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

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

        if(vis[nbr] == false) {
            travel(nbr,graph,vis);
        }
    }
}
public static boolean isGraphConnected(ArrayList<Edge>[]graph) {
     boolean[]vis = new boolean[graph.length];
     int count = 0;

     for(int i=0; i < graph.length;i++) {
         if(vis[i] == false) {
             travel(i,graph,vis);
             count++;

             if(count > 1) {
                 return false;
             }
         }
     }

     return true;
}
```





c = ~~0~~ 1

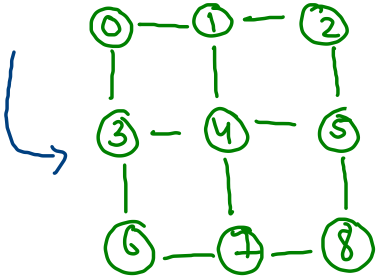
Number Of Islands

connected components

tddr

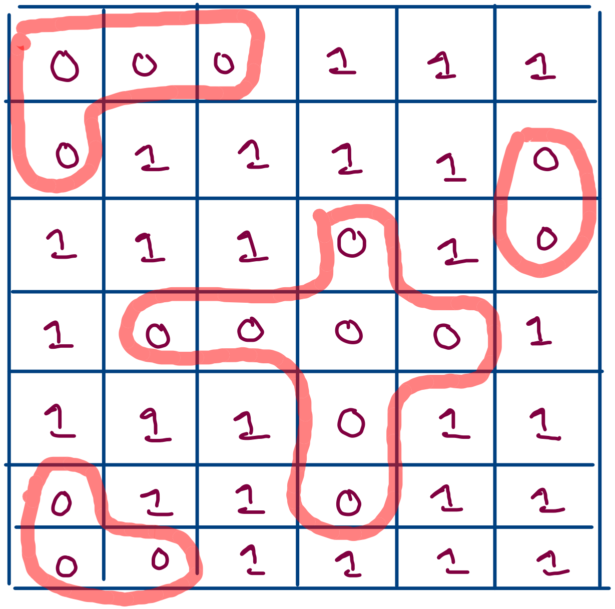
1. You are given a 2d array where 0's represent land and 1's represent water.
   Assume every cell is linked to it's north, east, west and south cell.
2. You are required to find and count the number of islands.

0	1	2
3	4	5
6	7	8



0 -> land

1 -> water



0	0	0	1	1	1
0	1	1	1	1	0
1	1	1	0	1	0
1	0	0	0	0	1
1	1	1	0	1	1
0	1	1	0	1	1
0	0	1	1	1	1

cells : vertices

connection : edges

	0	1	2	3	4	5
0	✓ 0	✓ 0	✓ 0	1	1	1
1	✓ 0	1	1	1	1	✓ 0
2	1	1	1	✓ tldr 0	1	✓ 0
3	1	✓ tldr 0	✓ tldr 0	✓ tldr 0	✓ tldr 0	1
4	1	1	1	✓ tldr 0	1	1
5	✓ 0	1	1	✓ tldr 0	1	1
6	✓ 0	✓ 0	1	1	1	1

count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

call -> 0, 0

call -> 1, 5

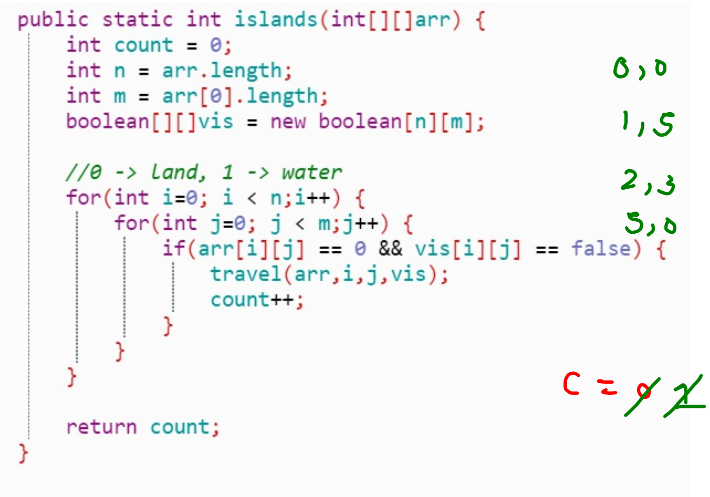
call -> 2, 3

tldr

call -> 5, 0

	0	1	2	3	4	5
0	✓ tddr 0	✓ tddr 0	✓ tddr 0	1	1	1
1	✓ tddr 0	1	1	1	1	✓ tddr 0
2	1	1	1	✓ tddr 0	1	✓ tddr 0
3	1	✓ tddr 0	✓ tddr 0	✓ tddr 0	✓ tddr 0	1
4	1	1	1	✓ tddr 0	1	1
5	✓ tddr 0	1	1	✓ tddr 0	1	1
6	✓ tddr 0	✓ tddr 0	1	1	1	1

n*m



```
public static int islands(int[][]arr) {
    int count = 0;
    int n = arr.length;
    int m = arr[0].length;
    boolean[][]vis = new boolean[n][m];

    //0 -> land, 1 -> water
    for(int i=0; i < n;i++) {
        for(int j=0; j < m;j++) {
            if(arr[i][j] == 0 && vis[i][j] == false) {
                travel(arr,i,j,vis);
                count++;
            }
        }
    }

    return count;
}
```

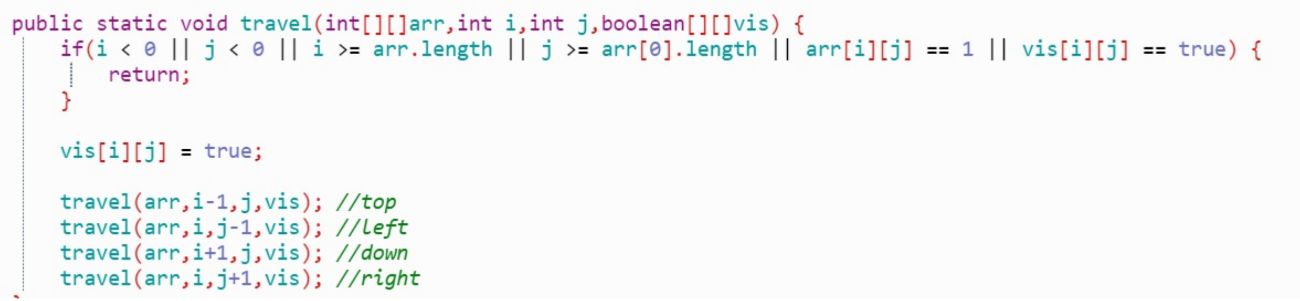
0,0

1,5

2,3

5,0

$C = \cancel{0}\ \cancel{1}\ \cancel{2}\ \cancel{3}\ 4$



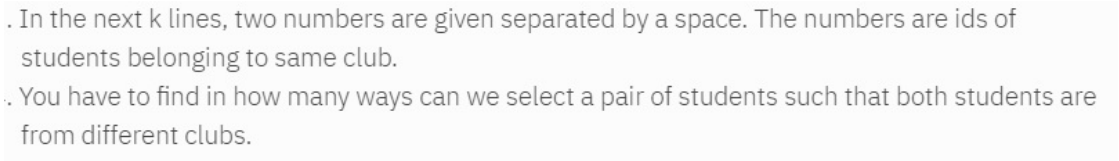
```
public static void travel(int[][]arr,int i,int j,boolean[][]vis) {
    if(i < 0 || j < 0 || i >= arr.length || j >= arr[0].length || arr[i][j] == 1 || vis[i][j] == true) {
        return;
    }

    vis[i][j] = true;

    travel(arr,i-1,j,vis); //top
    travel(arr,i,j-1,vis); //left
    travel(arr,i+1,j,vis); //down
    travel(arr,i,j+1,vis); //right
```

invalidity -> out of matrix,

water cell,

vis cell.

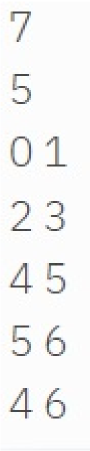
Perfect Friends



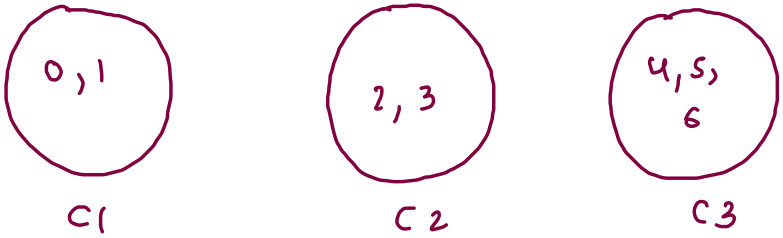
. In the next k lines, two numbers are given separated by a space. The numbers are ids of students belonging to same club.

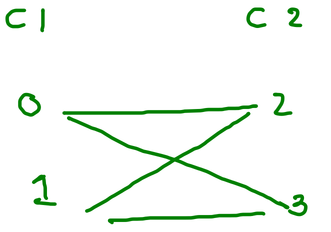
. You have to find in how many ways can we select a pair of students such that both students are from different clubs.

gcc

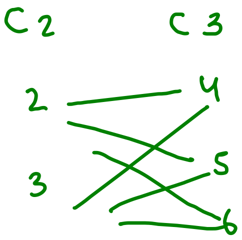


```
7
5
0 1
2 3
4 5
5 6
4 6
```

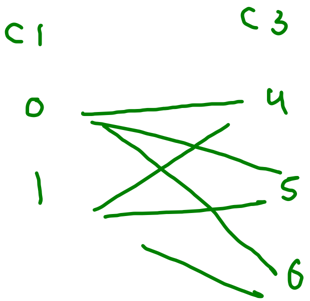




4

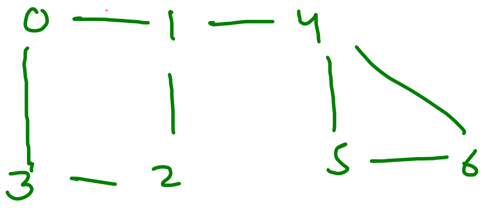


6



6

$4+6+6 = \boxed{16}$



```
ArrayList <Integer> [] graph;
```

0 → 1, 3

1 → 0, 2, 4

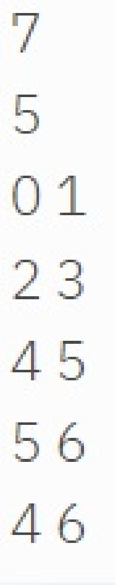
2 → 1, 3

3 → 0, 2

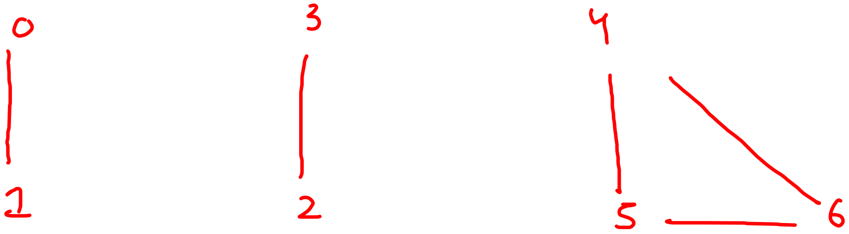
4 → 1, 5, 6

5 → 4, 6

6 → 4, 5



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



[2 , 2 , 3]

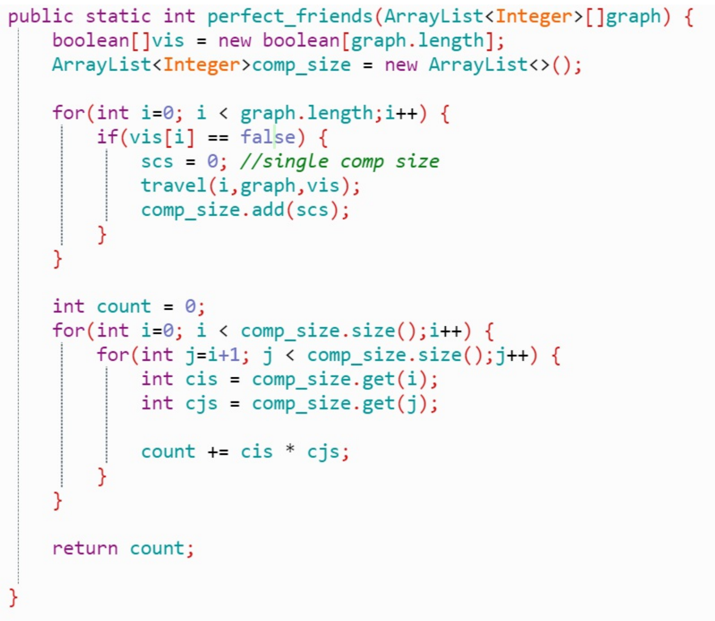
c1 c2 c3

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

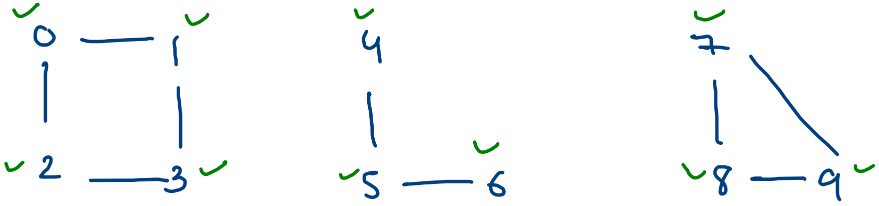
↓ ↓ ↓

2 , 2 , 3

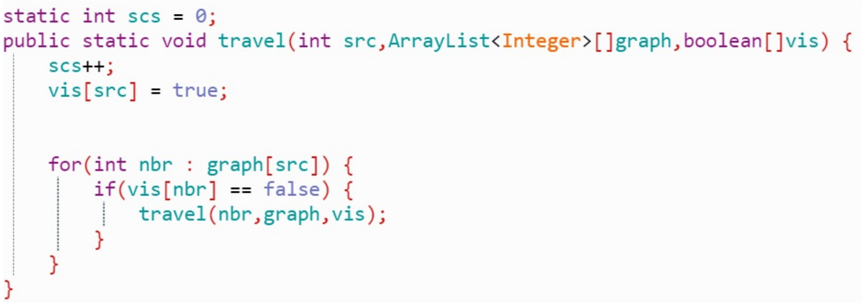
c1 c2 c3



```java
public static int perfect_friends(ArrayList<Integer>[]graph) {
    boolean[]vis = new boolean[graph.length];
    ArrayList<Integer>comp_size = new ArrayList<>();

    for(int i=0; i < graph.length;i++) {
        if(vis[i] == false) {
            scs = 0; //single comp size
            travel(i,graph,vis);
            comp_size.add(scs);
        }
    }

    int count = 0;
    for(int i=0; i < comp_size.size();i++) {
        for(int j=i+1; j < comp_size.size();j++) {
            int cis = comp_size.get(i);
            int cjs = comp_size.get(j);

            count += cis * cjs;
        }
    }

    return count;

}
```



scs = ~~0~~ ~~1~~ ~~2~~ 3



```java
static int scs = 0;
public static void travel(int src,ArrayList<Integer>[]graph,boolean[]vis) {
    scs++;
    vis[src] = true;


    for(int nbr : graph[src]) {
        if(vis[nbr] == false) {
            travel(nbr,graph,vis);
        }
    }
}
```

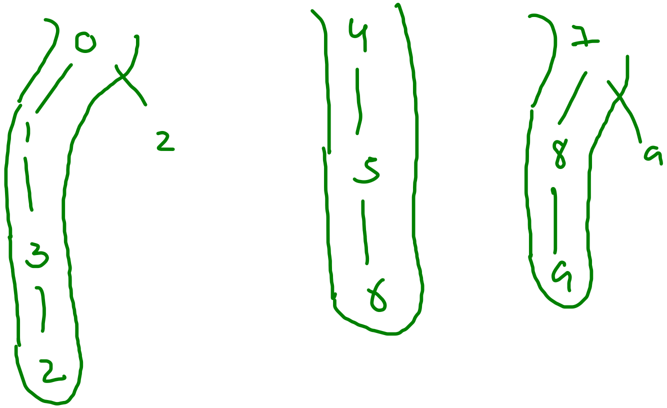
comp_ size

[4,3,3]

c1 c2 c3

c1 x c2, c1 x c3

c2 x c3



[4, 3, 3]

c1 c2 c3

12+12+9

33