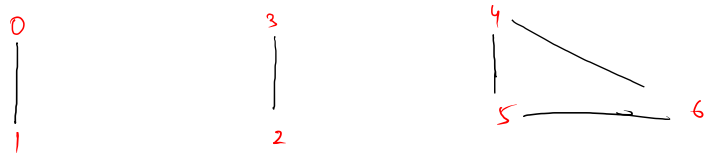
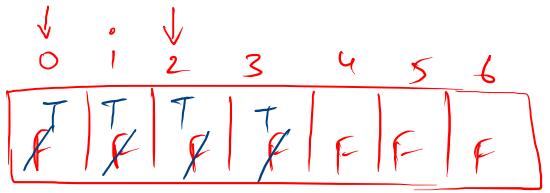
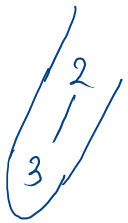
- Get Connected Components Of A Graph
- Is Graph Connected
- Number Of Islands
- Perfect Friends
- Hamiltonian Path And Cycle
- Knights Tour





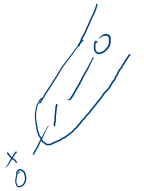

```java
public static ArrayList<ArrayList<Integer>> gcc(ArrayList<Edge>[] graph){
    boolean visited[] = new boolean[graph.length];
    ArrayList<ArrayList<Integer>> allComps = new ArrayList<>();

    for(int vtx = 0 ; vtx < graph.length ; vtx++){
        if(!visited[vtx]){
            ArrayList<Integer> comp = new ArrayList<>();
            gccHelper(graph,vtx,comp,visited);

            allComps.add(comp);
        }
    }

    return allComps;
}
```

allComps = [0|1] [2|3]



[2|3]

```java
public static void gccHelper(ArrayList<Edge>[] graph,int vtx,ArrayList<Integer> comp,boolean[] visited){
    visited[vtx] = true;
    comp.add(vtx);

    for(Edge e : graph[vtx]){
        if(!visited[e.nbr]){
            gccHelper(graph,e.nbr,comp,visited);
        }
    }
}
```

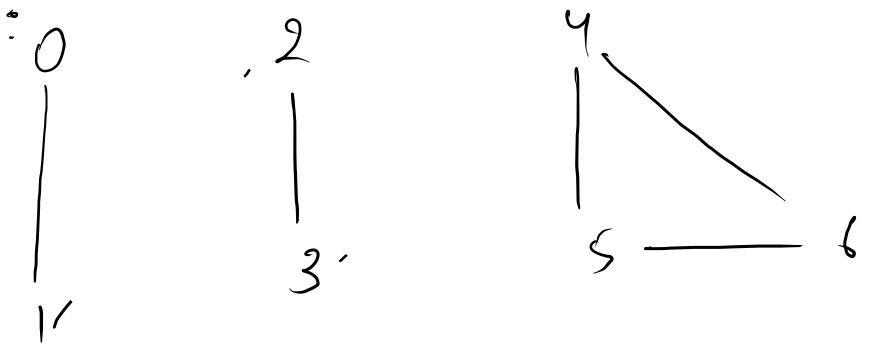


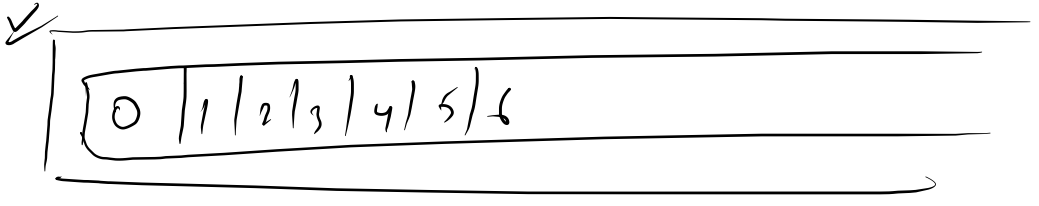
comp = [0|1]

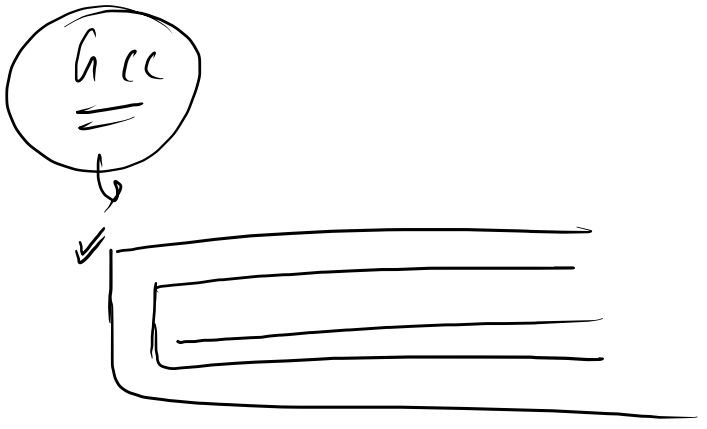
ArrayList<ArrayList<Integer>> res = [0|1] [2|3] [4|5|6]

7
5
0 1 10
2 3 10
4 5 10
5 6 10
4 6 10

Connected → Every vertex can be visited from each vertex.







Count = 0 1 2 3 = 3

(0,0) Double

(2,7)

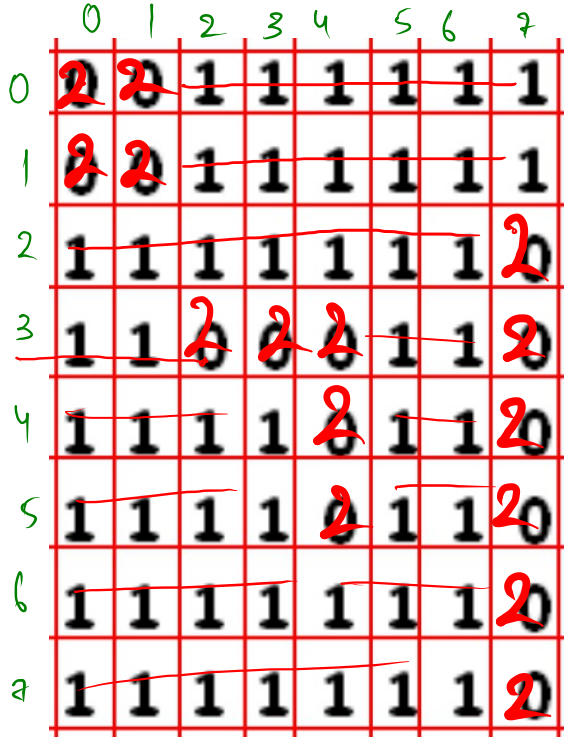
(3,2)

NEWS

0 → land

1 → water

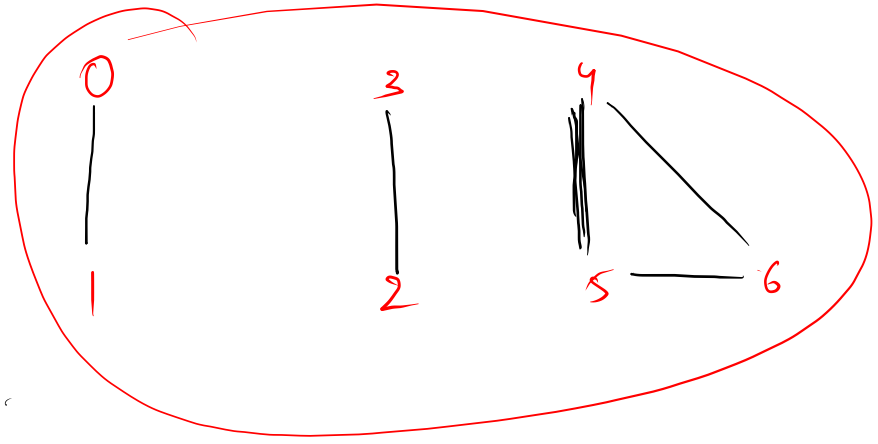
2 → visited



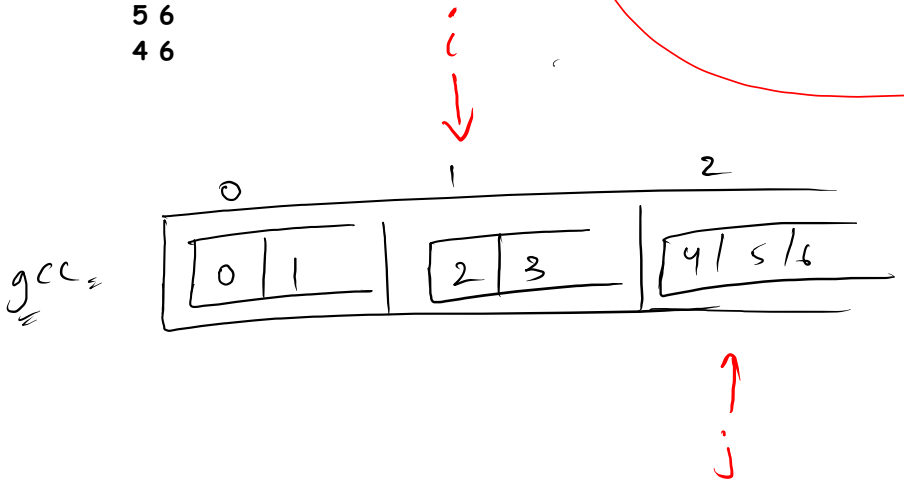

```java
public static int numOfIslands(int board[][]) {
  int count = 0;
  for (int i = 0 ; i < board.length ; i++) {
    for (int j = 0 ; j < board[0].length ; j++) {
      if (board[i][j] == 0) {
        count++;
        numOfIslandsHelper(board, i, j); // mark whole island
      }
    }
  }
  return count;
}

public static void numOfIslandsHelper(int[][] board, int r, int c) {
  if (r < 0 || c < 0 || r >= board.length || c >= board[0].length || board[r][c] == 1 || board[r][c] == 2) {
    return;
  }


  board[r][c] = 2;
  numOfIslandsHelper(board, r - 1, c); // north
  numOfIslandsHelper(board, r, c - 1); // east
  numOfIslandsHelper(board, r, c + 1); // west
  numOfIslandsHelper(board, r + 1, c); // south
}
```

5 min

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



No. of ways to select two students such that they don't belong to same club.

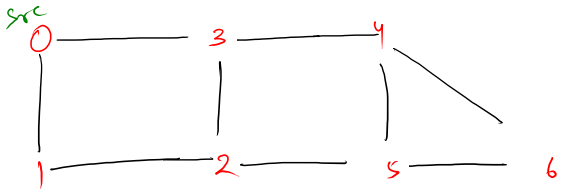


(2.2) + (2.3) + (2.3) => 16

```java
public static int perfectFriends(ArrayList<Edge>[] graph){
    ArrayList<ArrayList<Integer>> allComps = gcc(graph);
    int ways = 0;
    for(int i = 0 ; i < allComps.size() ; i++){
        for(int j = i+1 ; j < allComps.size() ; j++){
            ways += (allComps.get(i).size() * allComps.get(j).size());
        }
    }
    return ways;
}
```

Hamiltonian Cycle

HP + src → direction, e.vtx



visited = | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

0 1 2 3 4 5 6 ✓ :→ HP

0 1 2 3 4 6 5 ✓ → HP

0 1 2 5 4 3 ×

0 1 2 5 4 6 ×

0 1 2 5 6 4 3 ✓ → H.C.

0 3 2 1 ×

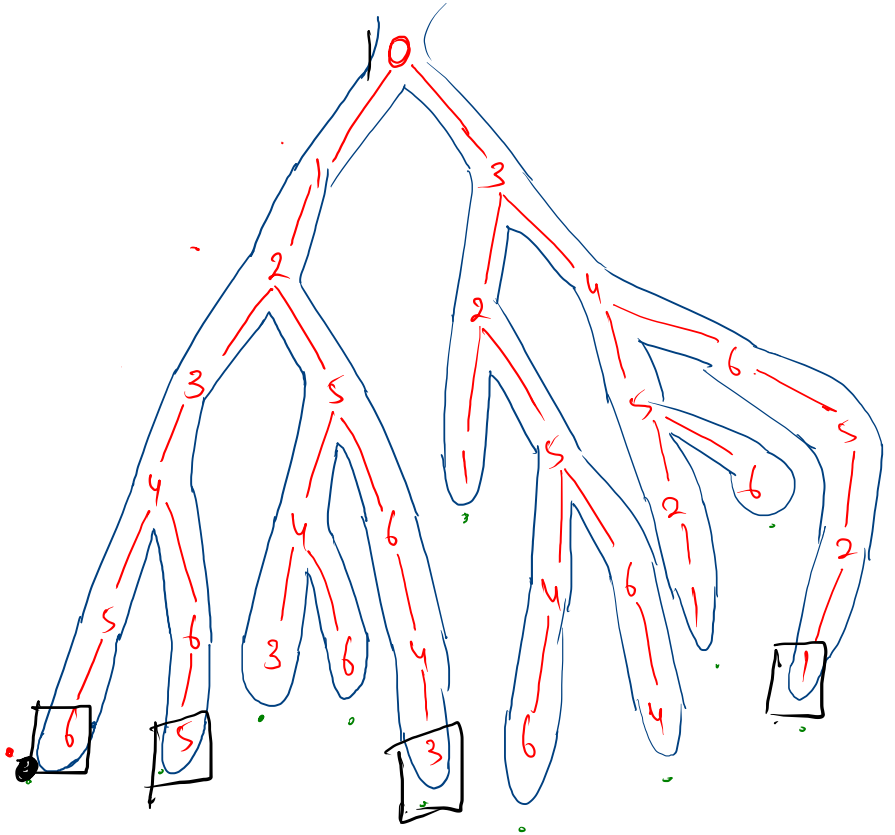
0 3 2 5 4 6 ×

0 3 2 5 6 4 ×

0 3 4 5 2 1 ×

0 3 4 5 6 ×

0 3 4 6 5 2 1 ✓ → H.C.



</>	Get Connected Components Of A Graph	Easy	10	✓Auth	0	✓Public	✓Sol	5
</>	Is Graph Connected	Easy	10	✓Auth	0	✓Public	✓Sol	6
</>	Number Of Islands	Easy	10	✓Auth	0	✓Public	✓Sol	7
</>	Perfect Friends	Easy	10	✓Auth	0	✓Public	✓Sol	8
</>	Knights Tour	Easy	10	✓Auth	0	✓Public	✓Sol	9
</>	Hamiltonian Path And Cycle	Easy	10	✓Auth	0	✓Public	✓Sol	10