```
Sending email
import java.io.*;
 import java.util.*;
 public class Main {
        static Vertex[] G;
        static int N, M;
        static final int INF = Integer.MAX_VALUE;
        public static void main(String[] args) throws Exception {
               LECTURA TXT

               StringBuilder sb = new StringBuilder();
               StringTokenizer st;
               String ent;
               int casos = Integer.parseInt(br.readLine());
               for (int k = 0; k < casos; k++) {
                      ent = br.readLine();
                      st = new StringTokenizer(ent);
                      N = Integer.parseInt(st.nextToken());
                      M = Integer.parseInt(st.nextToken());
                      int s = Integer.parseInt(st.nextToken());
                      int t = Integer.parseInt(st.nextToken());
                      G = new Vertex[N];
                      for (int i = 0; i < M; i++) {
                             st = new StringTokenizer(br.readLine());
                             int f = Integer.parseInt(st.nextToken());
                             int to = Integer.parseInt(st.nextToken());
                             int w = Integer.parseInt(st.nextToken());
                             if(G[f] == null){
                                    G[f] = new Vertex(f, INF);
                             }
                             if(G[to] == null){
                                    G[to] = new Vertex(to, INF);
                             }
                             G[f].la.add(new Edge(G[to], w));
                             G[to].la.add(new Edge(G[f], w));
                      }
                      int res = -1;
                      if(M>0) res = Dijkstra(s, t);
                      if (res == -1)
                             sb.append("Case #").append(k+1).append(": unreachable\n");
                    else
                     sb.append("Case #").append(k+1).append(": ").append(res).append("\n");
                     }
                     System.out.print(sb);
        }

        static int Dijkstra(int s, int t) {
                      boolean[] visited = new boolean[N];
                      PriorityQueue<Vertex> Q = new PriorityQueue<Vertex>();
                      G[s].dist = 0;
                      Q.offer(G[s]);
                      while(!Q.isEmpty()){
                             Vertex u = Q.poll();
                             if(u.id == t) return u.dist;
                             if(!visited[u.id]){
```

```
                                visited[u.id]=true;
                                for(Edge edge : G[u.id].la){
                                        Vertex w = edge.to;
                                        int peso = edge.peso;
                                        if(w.dist > u.dist+peso){
                                                w.dist = u.dist + peso;
                                                Q.offer(w);
                                        }
                                }
                        }
                }
                return -1;
        }
}
class Vertex implements Comparable<Vertex>{
        int id;
        int dist;
        List<Edge> la;
        public Vertex(int id, int d){
                this.id = id;
                this.la = new ArrayList<Edge>();
                this.dist = d;
        }
        @Override
        public int compareTo(Vertex v) {
                return dist - v.dist;
        }
}
class Edge{
        Vertex to;
        int peso;
        public Edge(Vertex v, int p){
                this.to = v;
                this.peso = p;
        }
}
```

Bombs! NO they are Mines!!

```
import java.io.*;
import java.lang.reflect.Array;
import java.text.DecimalFormat;
import java.util.*;

public class Main {
        static int[][] M;
        static int salida, n,m;
        static int[] dx = {0, 0, 1, -1};
        static int[] dy = {1, -1, 0, 0};
        public static void main(String[] args) throws Exception {
                LECTURA
                StringBuilder sb = new StringBuilder();
                StringTokenizer st;
                String ent;
                while((ent=br.readLine()) !=null) {
                        st = new StringTokenizer(ent);
```

```
			n = Integer.parseInt(st.nextToken());
			m = Integer.parseInt(st.nextToken());
			if(n==0) break;
			M = new int[n][m];
			int r = Integer.parseInt(br.readLine());
			for (int i = 0; i < r; i++) {
				st = new StringTokenizer(br.readLine());
				Integer k = Integer.parseInt(st.nextToken());
				st.nextToken();
				while(st.hasMoreElements()){
					int l = Integer.parseInt(st.nextToken());
					M[k][l] = 1;
				}
			}
			st = new StringTokenizer(br.readLine());
			int a = Integer.parseInt(st.nextToken());
			int b = Integer.parseInt(st.nextToken());
			st = new StringTokenizer(br.readLine());
			int c = Integer.parseInt(st.nextToken());
			int d = Integer.parseInt(st.nextToken());
			BFS(a,b,c,d);
			sb.append(salida).append("\n");

		}
		System.out.print(sb);
	}

	static void BFS(int a, int b, int c, int d) {
		Queue<Integer> q = new ArrayDeque<Integer>();
		Queue<Integer> cantidad = new ArrayDeque<Integer>();
		salida = 0;
		q.add(a);
		q.add(b);
		cantidad.add(0);
		while (!q.isEmpty()) {
			int u = q.poll(), v = q.poll(), cant = cantidad.poll();
			if(u == c && v == d){
				salida = cant;
				return;
			}else{
				for(int i = 0; i< 4; i++){
					int x = u + dx[i];
					int y = v + dy[i];
					if(isvalid(x, y) && M[x][y] == 0){
						q.add(x);
						q.add(y);
						cantidad.add(cant+1);
						M[x][y] = 1;
					}
				}
			}
		}

	}
	static boolean isvalid(int x, int y) {
```

```
            return (x < n && y < m && x >= 0 && y >= 0);
        }
}
```

All Roads Lead Where

```
import java.io.*;
import java.lang.reflect.Array;
import java.text.DecimalFormat;
import java.util.*;

public class Main {
        static HashMap<String, ArrayList<String>> hm;
        static HashMap<String, Boolean> hv;
        static String salida;

        public static void main(String[] args) throws Exception {
                LECTURA
                StringBuilder sb = new StringBuilder();
                StringTokenizer st;

                int casos = Integer.parseInt(br.readLine());
                while(casos-- > 0) {
                        br.readLine();
                        st = new StringTokenizer(br.readLine());
                        int m = Integer.parseInt(st.nextToken());
                        int n = Integer.parseInt(st.nextToken());
                        hm = new HashMap<String, ArrayList<String>>();
                        hv = new HashMap<String, Boolean>();
                        for (int i = 0; i < m; i++) {
                                st = new StringTokenizer(br.readLine());
                                String t1 = st.nextToken();
                                String t2 = st.nextToken();
                                if(hm.containsKey(t1)){
                                        ArrayList<String> tt= hm.get(t1);
                                        tt.add(t2);
                                        hm.put(t1, tt);
                                }else{
                                        ArrayList<String> tt = new ArrayList<String>();
                                        tt.add(t2);
                                        hm.put(t1, tt);
                                        hv.put(t1, false);
                                }
                                if(hm.containsKey(t2)){
                                        ArrayList<String> tt= hm.get(t2);
                                        tt.add(t1);
                                        hm.put(t2, tt);
                                }else{
                                        ArrayList<String> tt = new ArrayList<String>();
                                        tt.add(t1);
                                        hm.put(t2, tt);
                                        hv.put(t2, false);
                                }

                        }
                        for (int i = 0; i < n; i++) {
```

```
                        st = new StringTokenizer(br.readLine());
                        String t1 = st.nextToken();
                        String t2 = st.nextToken();
                        BFS(t1, t2);
                        sb.append(salida).append("\n");
                        exploredfalse();
                  }
                  //if(casos != 0) sb.append("\n");
            }
            System.out.print(sb);
      }

      static void BFS(String nd, String d) {
            Queue<String> q = new ArrayDeque<String>();
            Queue<String> camino = new ArrayDeque<String>();
            salida = "";
            hv.put(nd, true);
            q.add(nd);
            camino.add(""+nd.charAt(0));
            while (!q.isEmpty()) {
                  String u = q.poll(), recorrido = camino.poll();
                  if(u.equals(d)){
                        salida = recorrido;
                        return;
                  }else{
                        ArrayList<String> alc = hm.get(u);
                        for(int i = 0; i< alc.size(); i++){
                              String key = alc.get(i);
                              if(!hv.get(key)){
                                    q.add(key);
                                    hv.put(key, true);
                                    camino.add(recorrido+key.charAt(0));
                              }
                        }
                  }
            }

      }
      static void exploredfalse(){
            for(Entry<String, Boolean> dt:hv.entrySet()){
                  hv.put(dt.getKey(), false);
            }
      }
}
```

Counting Cells in a Blob

```
import java.io.*;
import java.util.*;
public class Main {
      static char M[][];
      static int dx[] = { 0, 1, 1, 1, 0, -1, -1, -1 }, dy[] = { -1, -1, 0, 1, 1,
                  1, 0, -1 }, m, n, sum;

      static void DFS(int index1, int index2) {
            M[index1][index2] = 0;
```

```
		int x, y;
		for (int k = 0; k < 8; k++) {
			x = index1 + dx[k];
			y = index2 + dy[k];
			if (isvalid(x, y) && M[x][y] == '1') {
				sum++;
				DFS(x, y);
			}
		}
	}

	static boolean isvalid(int x, int y) {
		return (x < n && y < n && x >= 0 && y >= 0);
	}

	public static void main(String[] args) throws Exception {
		LECTURA TXT
		StringBuilder sb = new StringBuilder();
		int casos = Integer.parseInt(br.readLine());
		for (int i = 0; i < casos; i++) {
			br.readLine();
			String ent = br.readLine();
			n = ent.length();
			M = new char[n][n];
			M[0] = ent.toCharArray();
			for (int j = 1; j < n; j++) M[j] = br.readLine().toCharArray();

			int max = Integer.MIN_VALUE;
			for (int j = 0; j < n; j++) {
				for (int k = 0; k < n; k++) {
					if (M[j][k] == '1') {
						sum = 1;
						DFS(j, k);
						max = Math.max(max, sum);
					}
				}
			}
			if(i != 0) sb.append("\n");
			if(max >0) sb.append(max).append("\n");
			else sb.append(0).append("\n");
		}
		System.out.print(sb);
	}
}
```