

public class A {

static final int dx[] = {-1, -1, -1, 0, 1, 1, 1, 0}; static final int dy[] = {-1, 0, 1, 1, 1, 0, -1, -1};

static int n, m; static int dinh[][]; static char ao[][];

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

n = in.nextInt(); m = in.nextInt(); in.nextLine();

dinh = new int[n + 1][m + 1];

ao = new char[n + 1][m + 1];

for (int i = 1; i <= n; i++) {

String s = in.nextLine();

for (int j = 1; j <= s.length(); j++) {

ao[i][j] = s.charAt(j - 1);

} } int dem = 0;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= m; j++) {

if (ao[i][j] == 'W' && dinh[i][j] == 0) {

dinh[i][j] = 1; dem++;

DFS(i, j);

}}}System.out.println(dem);

}

private static void DFS(int i, int j) {

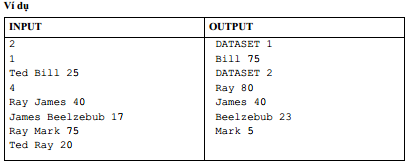
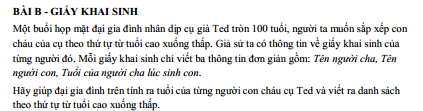
for (int k = 0; k < 8; k++) {

int x = j + dx[k]; int y = i + dy[k];

if (x >= 1 && x <= m && y >= 1 && y <= n && ao[y][x] == 'W' && dinh[y][x] == 0) {

dinh[y][x] = 1; DFS(y, x);

}}}}



public class B {

static String son[], father[];

static int ageHaveSon[]; static int age[];

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int testcase = in.nextInt();

for (int tc = 1; tc <= testcase; tc++) {

System.out.println("DATASET " + tc);

int k = in.nextInt(); son = new String[k];

father = new String[k]; age = new int[k]; ageHaveSon = new int[k];

for (int i = 0; i < k; i++) {

father[i] = in.next(); son[i] = in.next(); ageHaveSon[i] = in.nextInt();

}

Stack<String> name = new Stack<>(); Stack<Integer> index = new Stack<>();

name.push("Ted"); index.push(0);

while (!name.isEmpty()) {

String currentName = name.pop(); int currentIndex = index.pop();

for (int i = 0; i < k; i++) {

if (father[i].equals(currentName)) {

index.push(i);

name.push(son[i]);

if (currentName.equals("Ted")) {

age[i] = 100 - ageHaveSon[i];

} else {

age[i] = age[currentIndex] - ageHaveSon[i];

}}}}

for (int i = 0; i < k; i++) {

for (int j = i + 1; j < k; j++) {

if (age[i] < age[j] || (age[i] == age[j] && son[j].compareTo(son[i]) < 0)) {

int t = age[i]; age[i] = age[j]; age[j] = t;

String temp = son[i];

son[i] = son[j];

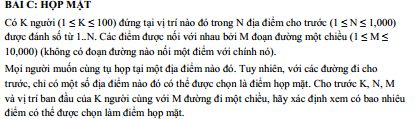
son[j] = temp;

}}}

for (int i = 0; i < k; i++) {

System.out.println(son[i] + " " + age[i]);

}}}}



public class C {

static boolean road[] = new boolean [1005];

static Vector<Integer> pos = null;

static ArrayList<Vector<Integer>> a = null;

static int n, k, m;

static void DFS(int i){

boolean flag[] = new boolean [1005];

Arrays.fill(flag, true);

flag[i] = false;

Stack<Integer> st = new Stack<Integer>();

st.push(i);

while(!st.empty()){

int u = st.peek();

st.pop();

for(int k = 0; k < a.get(u).size(); k++){

int v = a.get(u).get(k);

if(flag[v]){

flag[v] = false; st.push(v);

}}}

for(int j = 1; j <= n; j++){

if(flag[j]) road[j] = false;

}

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

Arrays.fill(road, true); pos = new Vector<>();

a = new ArrayList<Vector<Integer>>();

int t, t1, t2;

k = in.nextInt(); n = in.nextInt(); m = in.nextInt();

for (int i = 0; i<=n+2; i++) a.add( new Vector<Integer>());

for(int i = 1; i <= k; i++){

t = in.nextInt();

pos.add(t);

}

for(int i = 1; i <= m; i++){

t1 = in.nextInt(); t2 = in.nextInt();

a.get(t1).add(t2);

}

for(int i = 0; i < k; i++){

DFS(pos.get(i));

}

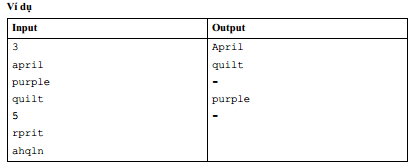
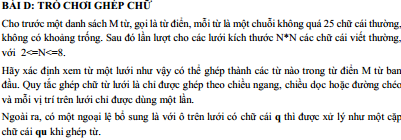
int count = 0;

for(int i = 1; i <= n; i++){

if(road[i]) count++; }

System.out.println(count);

}}



import java.io.IOException;

import java.io.InputStream; import java.util.ArrayList;

import java.util.Arrays; import java.util.Collections;

import java.util.InputMismatchException; import java.util.List;

import java.util.Scanner; import java.util.Vector;

public class D {

static InputStream is;

public static void main(String[] args) {

is = System.in;

int M = ni();

char[][] words = new char[M][];

for (int i = 0; i < M; i++) {

words[i] = ns().toCharArray();

}

StringBuilder outBf = new StringBuilder();

int N = 0;

while ((N = ni()) > 0) {

char[][] matrix = nm(N, N);

Node[][] nodes = new Node[N][N];

Node root = new Node('^');

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

root.nodes.add(nodes[i][j] = new Node(matrix[i][j]));

}}

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (i > 0 && j > 0) {

nodes[i][j].nodes.add(nodes[i - 1][j - 1]);

}

if (i > 0) {

nodes[i][j].nodes.add(nodes[i - 1][j]);

}

if (j > 0) {

nodes[i][j].nodes.add(nodes[i][j - 1]);

}

if (i > 0 && j < N - 1) {

nodes[i][j].nodes.add(nodes[i - 1][j + 1]);

}

if (i < N - 1 && j > 0) {

nodes[i][j].nodes.add(nodes[i + 1][j - 1]);

}

if (i < N - 1 && j < N - 1) {

nodes[i][j].nodes.add(nodes[i + 1][j + 1]);

}

if (i < N - 1) {

nodes[i][j].nodes.add(nodes[i + 1][j]);

}

if (j < N - 1) {

nodes[i][j].nodes.add(nodes[i][j + 1]);

}}}

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (nodes[i][j].letter == 'q') {

Node uNode = new Node('u');

uNode.nodes = nodes[i][j].nodes;

nodes[i][j].nodes = new ArrayList<Node>();

nodes[i][j].nodes.add(uNode);

}}}

List<String> result = new ArrayList<String>();

for (char[] word : words) {

if (find(root, word, 0)) {

result.add(new String(word));

}}

result.sort((s1, s2) -> s1.compareTo(s2));

for (String word : result) {

outBf.append(word + "\n");

} outBf.append("-\n");

}

System.out.print(outBf);

}

static public boolean find(Node node, char[] word, int index) {

if (index == word.length) {

return word[index - 1] != 'q';

}

node.visiting = true;

for (Node child : node.nodes) {

if (child.letter == word[index] && !child.visiting) {

boolean subResult = find(child, word, index + 1);

if (subResult) {

node.visiting = false;

return true;

}}}

node.visiting = false;

return false;

}

static class Node {

public char letter;

public boolean visiting;

public List<Node> nodes = new ArrayList<Node>();

public Node(char letter) {

this.letter = letter;

}}

static byte[] inbuf = new byte[4096];

static int lenbuf = 0, ptrbuf = 0;

static int readByte() {

if (lenbuf == -1) {

throw new InputMismatchException();

}

if (ptrbuf >= lenbuf) {

ptrbuf = 0;

try {

lenbuf = is.read(inbuf);

} catch (IOException e) {

throw new InputMismatchException();

}

if (lenbuf <= 0) {

return -1;

}}

return inbuf[ptrbuf++];

}

static boolean isSpaceChar(int c) {

return !(c >= 33 && c <= 126);

}

static int skip() {

int b;

while ((b = readByte()) != -1 && isSpaceChar(b));

return b;

}

static double nd() {

return Double.parseDouble(ns());

}

static char nc() {

return (char) skip();

}

static String ns() {

int b = skip();

StringBuilder sb = new StringBuilder();

while (!(isSpaceChar(b))) {

sb.appendCodePoint(b);

b = readByte();

}

return sb.toString();

}

static char[] ns(int n) {

char[] buf = new char[n];

int b = skip(), p = 0;

while (p < n && !(isSpaceChar(b))) {

buf[p++] = (char) b;

b = readByte();

} return n == p ? buf : Arrays.copyOf(buf, p);

}

static char[][] nm(int n, int m) {

char[][] map = new char[n][];

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

map[i] = ns(m);

}return map;

}

static int[] na(int n) {

int[] a = new int[n];

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

a[i] = ni();

} return a;

}

static int ni() {

int num = 0, b; boolean minus = false;

while ((b = readByte()) != -1 && !((b >= '0' && b <= '9') || b == '-'));

if (b == '-') {

minus = true; b = readByte();

}

while (true) {

if (b >= '0' && b <= '9') {

num = num \* 10 + (b - '0');

} else {

return minus ? -num : num;

}b = readByte();

}}

static long nl() {

long num = 0;

int b;

boolean minus = false;

while ((b = readByte()) != -1 && !((b >= '0' && b <= '9') || b == '-'));

if (b == '-') {

minus = true; b = readByte();

}

while (true) {

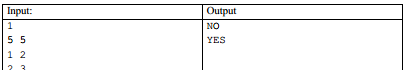
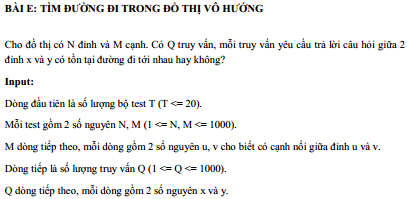
if (b >= '0' && b <= '9') {

num = num \* 10 + (b - '0');

} else { return minus ? -num : num;

} b = readByte();

}}}



public class E {

public static int m,n,q, x, y, F[] = new int[1001], A[][] = new int[1001][1001];

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int t = in.nextInt();

while (t-- > 0) {

n = in.nextInt(); m = in.nextInt();

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

A[i][j] = 0;

}}

for (int i = 1; i <= m; i++) {

int u = in.nextInt(), v = in.nextInt();

A[u][v] = 1;

A[v][u] = 1;

} q = in.nextInt();

while (q-- > 0) {

for (int i = 1; i <= n; i++) {

F[i] = 1;

}

x = in.nextInt(); y = in.nextInt();

DFS(x);

if (F[y] == 0) { System.out.println("YES");

} else { System.out.println("NO");

}}}}

public static void DFS(int u) {

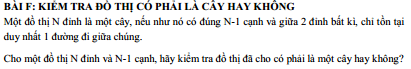
F[u] = 0;

for (int v = 1; v <= n; v++) {

if (A[u][v] == 1 && F[v] == 1) {

DFS(v);

}}}}



import java.util.Arrays;

import java.util.Queue;

import java.util.LinkedList;

import java.util.Scanner;

public class F {

static int A[][] = new int[1010][1010];

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int t = in.nextInt();

while (t-- > 0) {

int n, u, v;

n = in.nextInt();

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

for (int j = 0; j <= n; j++) {

A[i][j] = 0;

}}

boolean isTree = true;

for (int i = 1; i <= n - 1; i++) {

u = in.nextInt();

v = in.nextInt();

A[u][v]++;

A[v][u]++;

}

if (isTree == true && lt(n) == false) {

isTree = false;

}

if (isTree) { System.out.println("YES");

} else { System.out.println("NO");

}} }

static boolean lt(int n) {

boolean considered[] = new boolean[n + 1];

Arrays.fill(considered, false);

Queue<Integer> q = new LinkedList<Integer>();

q.add(1);

while (!q.isEmpty()) {

int u = q.peek(); considered[u] = true; q.remove();

for (int i = 1; i <= n; i++) {

int v = i;

if (A[u][v] >= 1 && considered[v] == false) {

q.add(v);

}}}

for (int i = 1; i <= n; i++) {

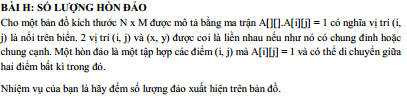
if (considered[i] == false) {

return false;

}} return true;

}

}



private static int h, c, a[][], b[][], dem;

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int test = in.nextInt();

while (test-- > 0) {

h = in.nextInt(); c = in.nextInt();

dem = 0; a = new int[503][503]; b = new int[503][503];

for (int i = 1; i <= h; i++) {

for (int j = 1; j <= c; j++) {

a[i][j] = in.nextInt();

b[i][j] = 1;

}}

for (int i = 1; i <= h; i++) {

for (int j = 1; j <= c; j++) {

if (a[i][j] == 1 && b[i][j] == 1) {

dem++;

dequy(i, j);

b[i][j] = 0;

}} } System.out.println(dem);

}}

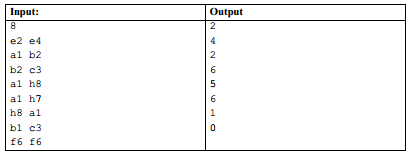
private static void dequy(int i, int j) {

if (a[i][j] == 1 && b[i][j] == 1) {

b[i][j] = 0; dequy(i, j - 1); dequy(i + 1, j - 1); dequy(i - 1, j - 1); dequy(i, j + 1);

dequy(i + 1, j + 1); dequy(i - 1, j + 1); dequy(i - 1, j); dequy(i + 1, j);

}}



import java.io.IOException; import java.util.HashMap;

import java.util.Queue; import java.util.LinkedList;

import java.util.Map; import java.util.Scanner;

import java.util.StringTokenizer;

class ChessBoard {

private int Grid[][];

public ChessBoard(int row, int col) {

Grid = new int[row][col];

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

Grid[i][j] = 0;

}}}

void setCellValue(int x, int y, int v) {

Grid[x][y] = v;

}

int getCellValue(int x, int y) {

return Grid[x][y];

}}

class Pair<F, L> {

private final F first; private final L last;

public Pair(F f, L l) {

first = f; last = l;

} F getFirst() { return first; }

L getLast() {

return last;

}}

public class I {

private final int ROW = 8; private final int COL = 8;

private final char FROM = 'a'; private final char TO = 'h';

private final int fx[] = {1, 1, -1, -1, 2, 2, -2, -2};

private final int fy[] = {2, -2, 2, -2, 1, -1, 1, -1};

int sx, sy; int dx, dy;

public static void main(String args[]) {

I obj = new I();

obj.beginProcess();

}

void beginProcess() {

Scanner in = new Scanner(System.in);

int test = in.nextInt();

Map<Character, Integer> col\_map = new HashMap<Character, Integer>();

int i = 0;

for (char ch = FROM; ch <= TO; ch++) {

col\_map.put(new Character(ch), new Integer(i));

i++;

}

while (test-- > -1) {

StringTokenizer token = new StringTokenizer(in.nextLine())

if (token.countTokens() == 2) {

String source = token.nextToken();

String destination = token.nextToken();

if (source.length() == 2 && destination.length() == 2) {

sx = Integer.parseInt(Character.toString(source.charAt(1)));

char c1 = source.charAt(0);

dx = Integer.parseInt(Character.toString(destination.charAt(1)));

char c2 = destination.charAt(0);

if (c1 >= FROM && c1 <= TO && c2 >= FROM && c1 <= TO && sx > 0 && sx <= ROW && dx > 0 && dx <= COL) {

sx = sx - 1;

sy = (int) col\_map.get(c1);

dx = dx - 1;

dy = (int) col\_map.get(c2);

ChessBoard board = new ChessBoard(ROW, COL);

System.out.println(bfs(board));

}}}} }

int bfs(ChessBoard board) {

int distance[][] = new int[ROW][COL];

distance[sx][sy] = 0;

board.setCellValue(sx, sy, 1);

Queue<Pair> queue = new LinkedList<>();

queue.add(new Pair<Integer, Integer>(sx, sy));

while (!queue.isEmpty()) {

Pair top = queue.poll();

for (int i = 0; i < ROW; i++) {

int tx = ((int) top.getFirst()) + fx[i];

int ty = ((int) top.getLast()) + fy[i];

if (tx >= 0 && tx < ROW && ty >= 0 && ty < COL && board.getCellValue(tx, ty) == 0) {

distance[tx][ty] = distance[(int) top.getFirst()][(int) top.getLast()] + 1;

board.setCellValue(tx, ty, 1);

queue.add(new Pair<Integer, Integer>(tx, ty));

} else {

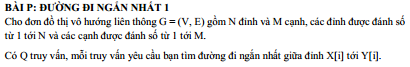
if (tx == dx && ty == dy) {

break;

}}}}

return distance[dx][dy];

}}



public class P {

static int n, m, u, v, c, b[][];

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

b = new int[1001][1001];

n = in.nextInt();

m = in.nextInt();

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if(i == j) {

b[i][j] = 0;

}

else {

b[i][j] = 999999;

}

}

}

for (int i = 1; i <= m; i++) {

int u = in.nextInt();

int v = in.nextInt();

int c = in.nextInt();

b[u][v] = c;

b[v][u] = c;

}

for (int k = 1; k <= n; k++) {

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if(b[i][j] > b[i][k] + b[k][j]) {

b[i][j] = b[i][k] + b[k][j];

}

}

}

}

int q = in.nextInt();

while(q--> 0) {

int u = in.nextInt();

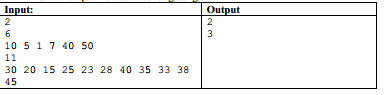
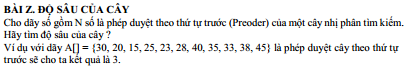
int v = in.nextInt();

System.out.println(b[u][v]);

}

}

}



import java.util.Scanner;

class Node {

int data;

public Node left, right;

Node(int d) {

data = d;

left = right = null;

}

}

class Index {

int index = 0;

}

class Z {

static int dem, pre[] = new int[10001];

Index index = new Index();

Node constructTreeUtil(int pre[], Index preIndex, int key,

int min, int max, int size) {

if (preIndex.index >= size) {

return null;

}

Node root = null;

if (key > min && key < max) {

root = new Node(key);

preIndex.index = preIndex.index + 1;

if (preIndex.index < size) {

root.left = constructTreeUtil(pre, preIndex, pre[preIndex.index],

min, key, size);

root.right = constructTreeUtil(pre, preIndex, pre[preIndex.index],

key, max, size);

}

}

return root;

}

Node constructTree(int pre[], int size) {

int preIndex = 0;

return constructTreeUtil(pre, index, pre[0], Integer.MIN\_VALUE,

Integer.MAX\_VALUE, size);

}

static void preorder(Node r) {

if (r != null) {

dem++;

System.out.print(r.data +" ");

preorder(r.left);

preorder(r.right);

}

}

void printInorder(Node node) {

if (node == null) {

return;

}

printInorder(node.left);

System.out.print(node.data + " ");

printInorder(node.right);

}

static int maxDept(Node root) {

if(root == null) return 0;

int left = maxDept(root.left);

int right = maxDept(root.right);

return Math.max(left, right) + 1;

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int t = in.nextInt();

while(t-->0) {

int n = in.nextInt();

Z tree = new Z();

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

pre[i] = in.nextInt();

}

Node root = tree.constructTree(pre, pre.length);

System.out.println(maxDept(root) - 1);

}

//int pre[] = new int[]{10, 5, 1, 7, 40, 50};

//tree.printInorder(root);

}

}