import java.util.Arrays;

import java.util.LinkedList;

import java.util.List;

import java.util.Scanner;

public class MainClass {

public static void main(String[] args) {

Pathfinding\_1();

}

private static List<List<Integer>> queue = new LinkedList<List<Integer>>();

private static int mods = 0;

private static void Pathfinding\_1() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter end coordinates: ");

queue.add(Arrays.asList(sc.nextInt(), sc.nextInt(), 0));

sc.close();

int value = -1;

for (int i = 0; i < queue.size(); i++) {

//System.out.println(queue.toString() + "\n");

boolean proceed = true;

for (int j = 0; j < queue.size(); j++)

if (queue.get(j).get(0) == 0 && queue.get(j).get(1) == 0) {

value = queue.get(j).get(2);

proceed = false;

break;

}

if (!proceed) break;

AddAdjacent(queue.get(i));

}

//System.out.println(queue.toString() + "\n");

List<Integer> pos = new LinkedList<Integer>();

pos.add(0);

pos.add(0);

pos.add(value);

//System.out.println(pos);

FindPath(pos);

System.out.println("Mods: " + mods);

}

private static void FindPath(List<Integer> position) {

for (List<Integer> next : queue) {

if (position.equals(queue.get(0))) {

//System.out.println("END " + position + " res: " + mods);

mods++;

break;

}

else if ((next.get(2) == position.get(2) - 1) && ((Math.abs(next.get(0) - position.get(0)) == 1) || (Math.abs(next.get(0) - position.get(0)) == 0))

&& ((Math.abs(next.get(1) - position.get(1)) == 1) || (Math.abs(next.get(1) - position.get(1)) == 0)) )

{

//System.out.println("new node: " + next);

FindPath(next);

}

}

}

private static void AddAdjacent(List<Integer> next) {

List<List<Integer>> adjacent = new LinkedList<List<Integer>>();

adjacent.add(Arrays.asList(next.get(0) - 1, next.get(1), next.get(2) + 1)); //stanga

adjacent.add(Arrays.asList(next.get(0), next.get(1) + 1, next.get(2) + 1)); //sus

adjacent.add(Arrays.asList(next.get(0) + 1, next.get(1), next.get(2) + 1)); //dreapta

adjacent.add(Arrays.asList(next.get(0), next.get(1) - 1, next.get(2) + 1)); //jos

for (int i = 0; i < queue.size(); i++) {

for (int k = 0; k < adjacent.size(); k++) {

List<Integer> l1 = queue.get(i);

List<Integer> l2 = adjacent.get(k);

if (l1.get(0) == l2.get(0) && l1.get(1) == l2.get(1) && l1.get(2) <= l2.get(2)) {

adjacent.remove(k);

k--;

}

}

}

for (int i = 0; i < adjacent.size(); i++)

queue.add(adjacent.get(i));

}

}