UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

INTELIGENCIA ARTIFICIAL

CONTENIDO:

LAB-1. EL PROBLEMA DEL LABERINTO BÁSICO.

PORCENTAJE TERMINADO: 100%

INTEGRANTES	DT	HG	HI	EVAL
Garcia Taboada Brayan Albaro	1	1	1	100

Fecha de presentación : Jueves, 2 de Mayo de 2024

Fecha Presentada : : Jueves, 2 de Mayo de 2024

Días de Atraso : 0

EL PROBLEMA DEL LABERINTO.

TRABAJO INDIVIDUAL.

1. Dado una matriz de n x m, inicialmente todas las posiciones con valores de cero, avanzar las casillas en sentido horario con movimientos de izquierda, arriba, derecha y abajo. Hacer Algoritmos para los siguientes:

a) Algoritmo para mostrar todos los caminos posibles desde una posición inicial a una posición final. Además, mostrar la cantidad de soluciones posibles (Cantidad de caminos posibles de la posición inicial a la posición final).

b) Algoritmo para mostrar todos los caminos posibles desde una posición inicial a una posición final tal que se visiten todas las casillas de la matriz. Además, mostrar la cantidad de soluciones posibles.

c) Algoritmo para mostrar todos los caminos posibles desde una posición inicial a una posición final tal que NO se visiten todas las casillas de la matriz. Además, mostrar la cantidad de soluciones posibles.

```
public class LABERINTO {
public static int soluciones=0;
   public static void main(String[] args) {
      System.out.println("Hello World!");
      int a=5,b=5;
      int m[][]=new int [a][b];
      mostrar(m);
      laberintoLleno(m,0,0,a-1,b-1,1);
      System.out.println(soluciones);
   }
   public static void mostrar(int m[][]){
      for (int i=0;i<m.length;i++){
         for (int j = 0; j < m[i].length; j++) {
            System.out.print(m[i][j]+"-");
         }
         System.out.println();
      }
```

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

public static void laberinto(int m[][],int i1,int j1, int i2,int j2,int paso){
   if(!posValida(m,i1,j1)) return;
      m[i1][j1]=paso;
      if(i1==i2 && j1==j2) {
         mostrar (m);
         soluciones++;
      }
      laberinto(m,i1,j1-1,i2,j2,paso+1);
      laberinto(m,i1-1,j1,i2,j2,paso+1);
      laberinto(m,i1,j1+1,i2,j2,paso+1);
      laberinto(m,i1+1,j1,i2,j2,paso+1);
   m[i1][j1]=0;
}
public static boolean posValida(int m[][],int i,int j){
   return i>=0 && i<m.length &&
         j>=0 && j<m[i].length && m[i][j]==0;
}
   public static void laberintoLleno(int m[][],int i1,int j1, int i2,int j2,int paso){
   if(!posValida(m,i1,j1)) return;
      m[i1][j1]=paso;
      if(i1==i2 && j1==j2 && todosMarcados(m)) {
         mostrar (m);
         soluciones++;
      }
      laberintoLleno(m,i1,j1-1,i2,j2,paso+1);
      laberintoLleno(m,i1-1,j1,i2,j2,paso+1);
      laberintoLleno(m,i1,j1+1,i2,j2,paso+1);
      laberintoLleno(m,i1+1,j1,i2,j2,paso+1);
```

```
        m[i1][j1]=0;
    }
        public static void laberintoNoLleno(int m[][],int i1,int j1, int i2,int j2,int paso){
    if(!posValida(m,i1,j1)) return;
        m[i1][j1]=paso;
        if(i1==i2 && j1==j2 && !todosMarcados(m)) {
            mostrar (m);
            soluciones++;
        }
        laberintoNoLleno(m,i1,j1-1,i2,j2,paso+1);
        laberintoNoLleno(m,i1-1,j1,i2,j2,paso+1);
        laberintoNoLleno(m,i1,j1+1,i2,j2,paso+1);
        laberintoNoLleno(m,i1+1,j1,i2,j2,paso+1);
        m[i1][j1]=0;
    }
    public static boolean todosMarcados(int m[][]){
        for (int i=0;i<m.length;i++){
        for (int j = 0; j < m[i].length; j++) {
            if(m[i][j]==0) return false;
        }
    }
        return true;
    }
}
```