UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGON

TAREA: INVESTIGACION

P R E S E N T A

Alexis Hernández Zamudio

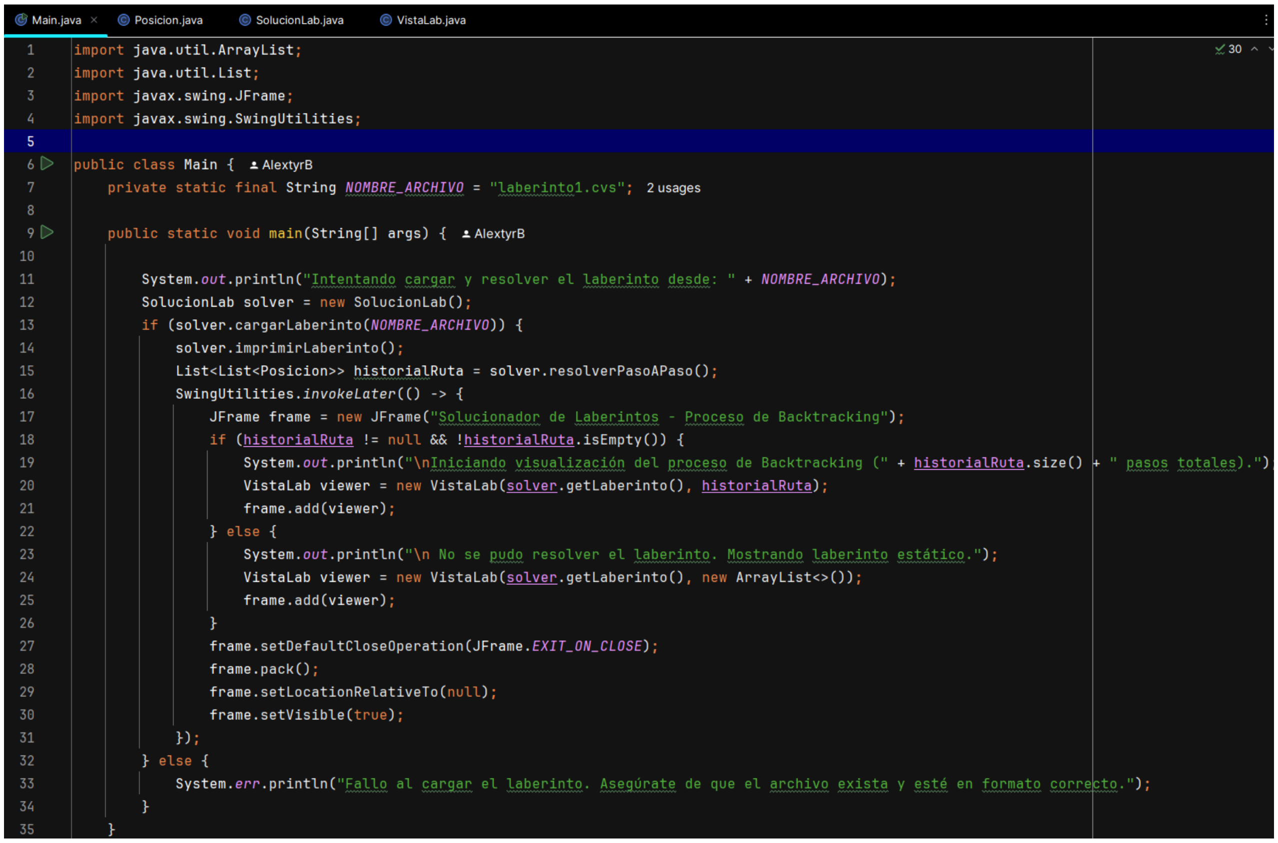
APROFESOR

Jesús Hernández Cabrera

Gpo:1558

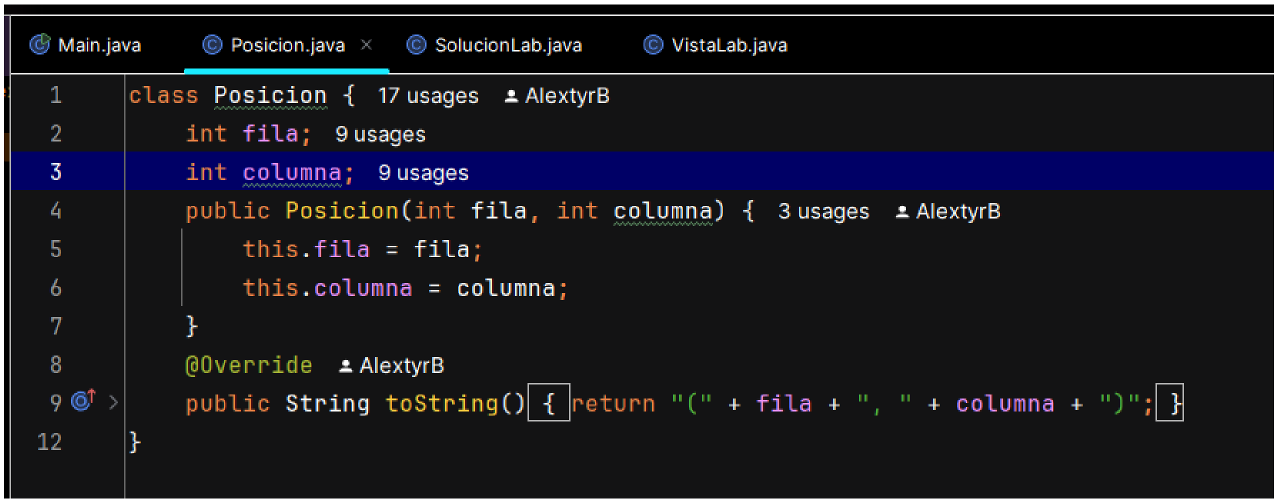
Ciudad Nezahualcóyotl, EDOMEX. 11 DE noviembre del 2025

Clase Main



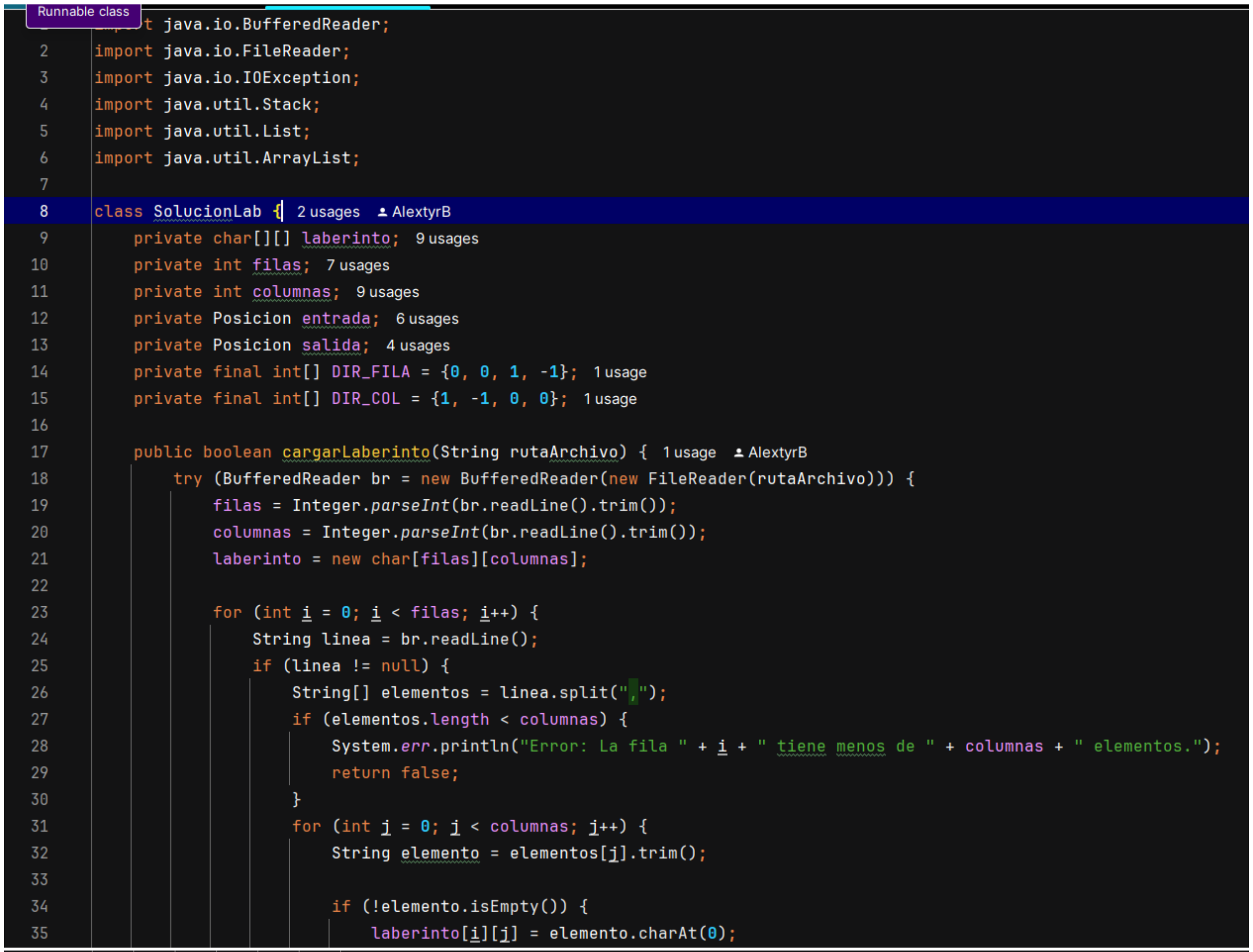
```java
import java.util.ArrayList;
import java.util.List;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;

public class Main {
    private static final String NOMBRE_ARCHIVO = "laberinto1.cvs";

    public static void main(String[] args) {

        System.out.println("Intentando cargar y resolver el laberinto desde: " + NOMBRE_ARCHIVO);
        SolucionLab solver = new SolucionLab();
        if (solver.cargarLaberinto(NOMBRE_ARCHIVO)) {
            solver.imprimirLaberinto();
            List<List<Posicion>> historialRuta = solver.resolverPasoAPaso();
            SwingUtilities.invokeLater(() -> {
                JFrame frame = new JFrame("Solucionador de Laberintos - Proceso de Backtracking");
                if (historialRuta != null && !historialRuta.isEmpty()) {
                    System.out.println("\nIniciando visualización del proceso de Backtracking (" + historialRuta.size() + " pasos totales).")
                    VistaLab viewer = new VistaLab(solver.getLaberinto(), historialRuta);
                    frame.add(viewer);
                } else {
                    System.out.println("\n No se pudo resolver el laberinto. Mostrando laberinto estático.");
                    VistaLab viewer = new VistaLab(solver.getLaberinto(), new ArrayList<>());
                    frame.add(viewer);
                }
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.pack();
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
            });
        } else {
            System.err.println("Fallo al cargar el laberinto. Asegúrate de que el archivo exista y esté en formato correcto.");
        }
    }
```

Clase posición

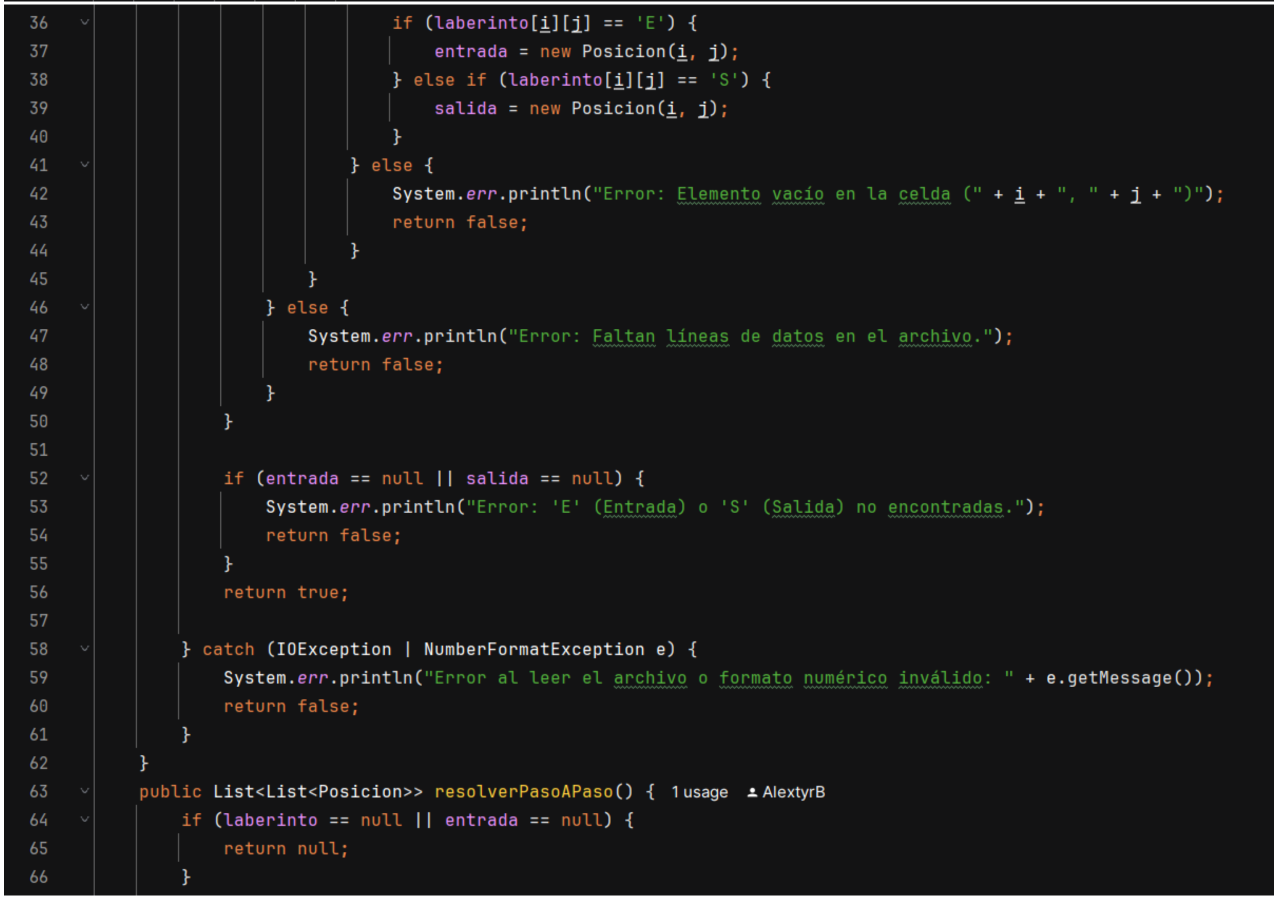


```java
class Posicion {
    int fila;
    int columna;
    public Posicion(int fila, int columna) {
        this.fila = fila;
        this.columna = columna;
    }
    @Override
    public String toString() { return "(" + fila + ", " + columna + ")"; }
}
```

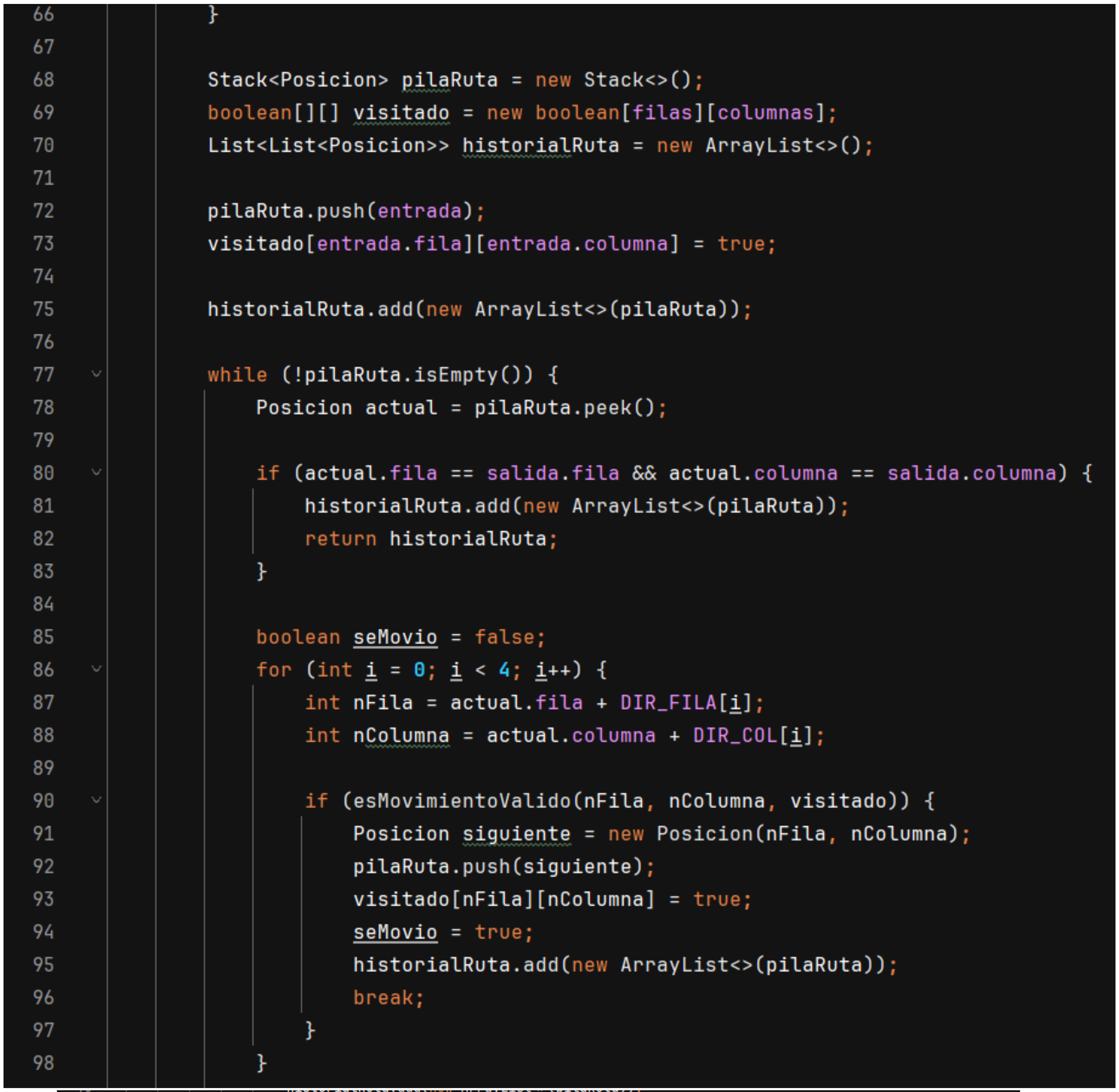
Solucion de laberinto



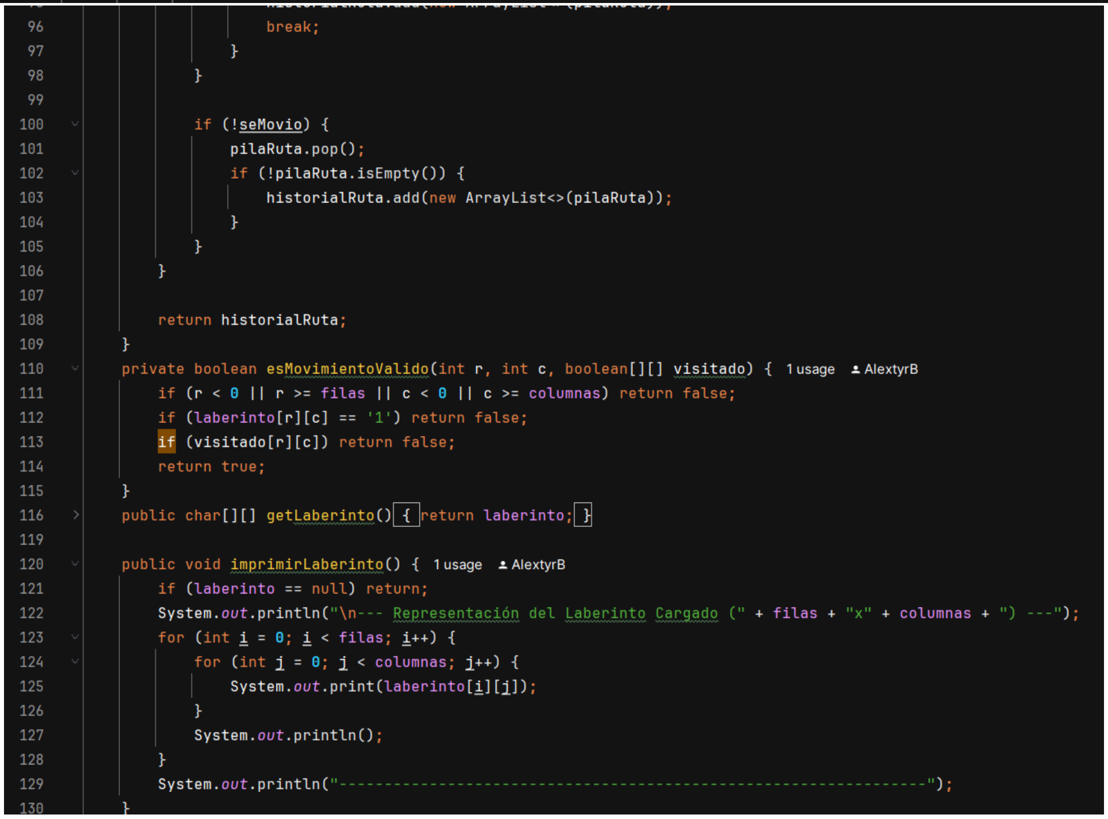
```java
t java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Stack;
import java.util.List;
import java.util.ArrayList;

class SolucionLab {  2 usages  AlexytrB
    private char[][] laberinto;  9 usages
    private int filas;  7 usages
    private int columnas;  9 usages
    private Posicion entrada;  6 usages
    private Posicion salida;  4 usages
    private final int[] DIR_FILA = {0, 0, 1, -1};  1 usage
    private final int[] DIR_COL = {1, -1, 0, 0};  1 usage

    public boolean cargarLaberinto(String rutaArchivo) {  1 usage  AlexytrB
        try (BufferedReader br = new BufferedReader(new FileReader(rutaArchivo))) {
            filas = Integer.parseInt(br.readLine().trim());
            columnas = Integer.parseInt(br.readLine().trim());
            laberinto = new char[filas][columnas];

            for (int i = 0; i < filas; i++) {
                String linea = br.readLine();
                if (linea != null) {
                    String[] elementos = linea.split(",");
                    if (elementos.length < columnas) {
                        System.err.println("Error: La fila " + i + " tiene menos de " + columnas + " elementos.");
                        return false;
                    }
                    for (int j = 0; j < columnas; j++) {
                        String elemento = elementos[j].trim();

                        if (!elemento.isEmpty()) {
                            laberinto[i][j] = elemento.charAt(0);
```



```java
                            if (laberinto[i][j] == 'E') {
                                entrada = new Posicion(i, j);
                            } else if (laberinto[i][j] == 'S') {
                                salida = new Posicion(i, j);
                            }
                        } else {
                            System.err.println("Error: Elemento vacío en la celda (" + i + ", " + j + ")");
                            return false;
                        }
                    }
                } else {
                    System.err.println("Error: Faltan líneas de datos en el archivo.");
                    return false;
                }
            }

            if (entrada == null || salida == null) {
                System.err.println("Error: 'E' (Entrada) o 'S' (Salida) no encontradas.");
                return false;
            }
            return true;

        } catch (IOException | NumberFormatException e) {
            System.err.println("Error al leer el archivo o formato numérico inválido: " + e.getMessage());
            return false;
        }
    }
    public List<List<Posicion>> resolverPasoAPaso() {  1 usage  AlexytrB
        if (laberinto == null || entrada == null) {
            return null;
        }
```

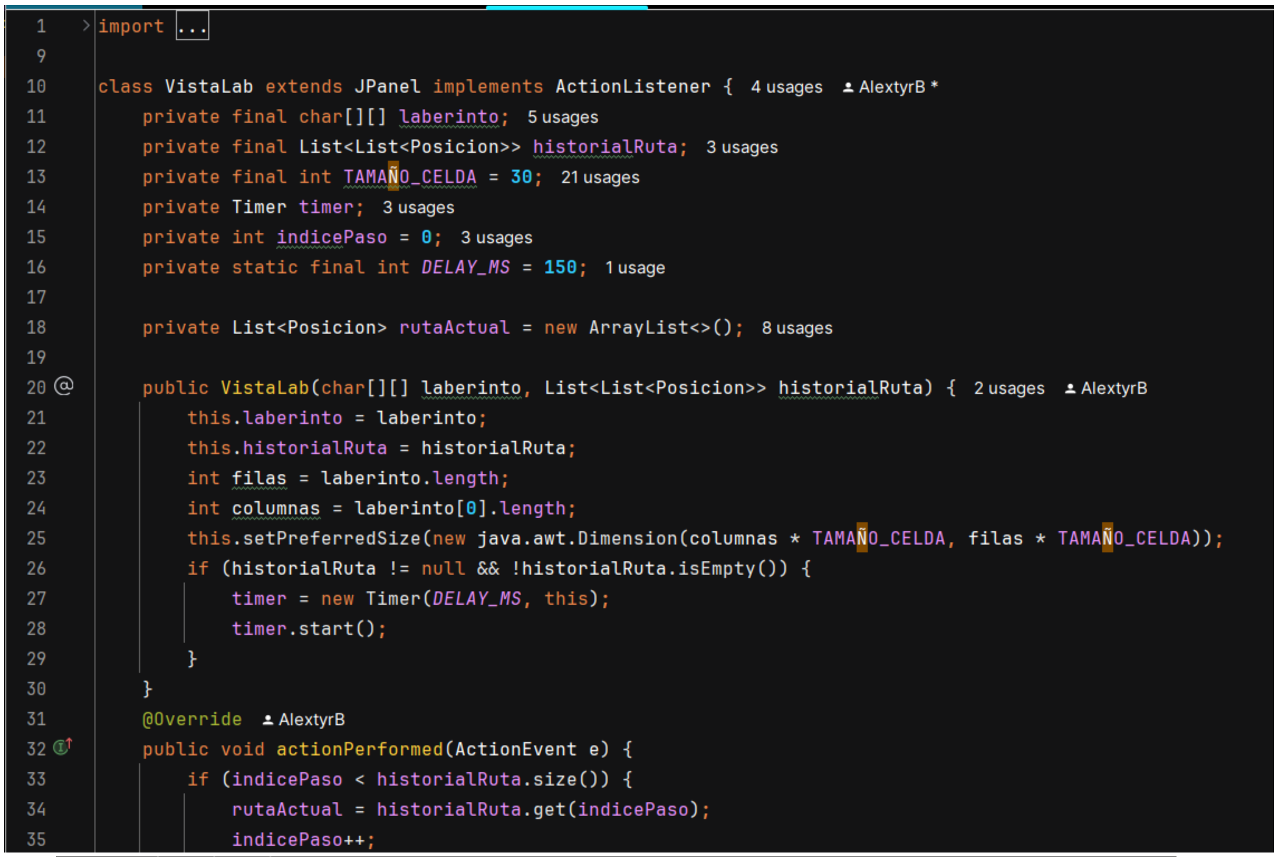


```java
        }

        Stack<Posicion> pilaRuta = new Stack<>();
        boolean[][] visitado = new boolean[filas][columnas];
        List<List<Posicion>> historialRuta = new ArrayList<>();

        pilaRuta.push(entrada);
        visitado[entrada.fila][entrada.columna] = true;

        historialRuta.add(new ArrayList<>(pilaRuta));

        while (!pilaRuta.isEmpty()) {
            Posicion actual = pilaRuta.peek();

            if (actual.fila == salida.fila && actual.columna == salida.columna) {
                historialRuta.add(new ArrayList<>(pilaRuta));
                return historialRuta;
            }

            boolean seMovio = false;
            for (int i = 0; i < 4; i++) {
                int nFila = actual.fila + DIR_FILA[i];
                int nColumna = actual.columna + DIR_COL[i];

                if (esMovimientoValido(nFila, nColumna, visitado)) {
                    Posicion siguiente = new Posicion(nFila, nColumna);
                    pilaRuta.push(siguiente);
                    visitado[nFila][nColumna] = true;
                    seMovio = true;
                    historialRuta.add(new ArrayList<>(pilaRuta));
                    break;
                }
            }
```

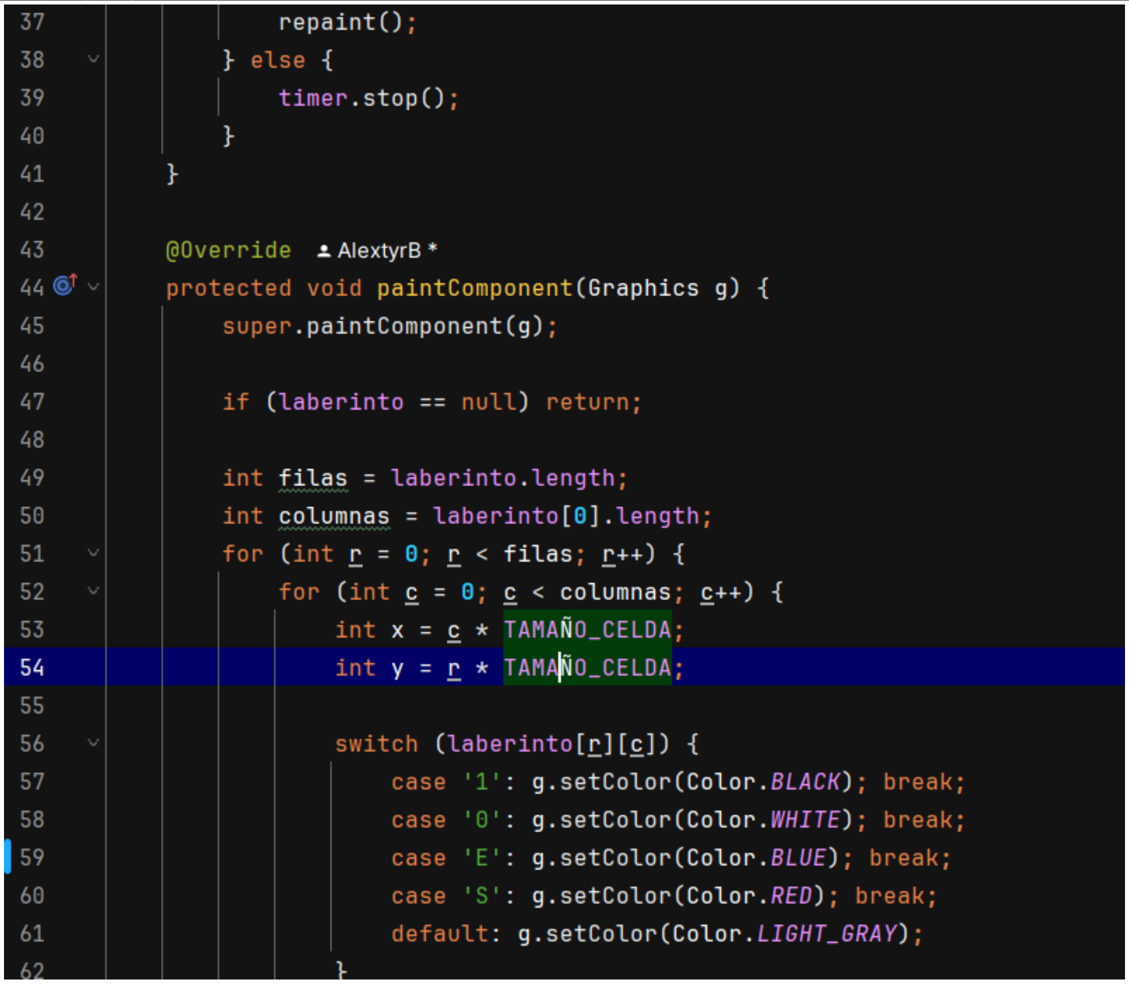


```java
                    break;
                }
            }

            if (!seMovio) {
                pilaRuta.pop();
                if (!pilaRuta.isEmpty()) {
                    historialRuta.add(new ArrayList<>(pilaRuta));
                }
            }
        }

        return historialRuta;
    }
    private boolean esMovimientoValido(int r, int c, boolean[][] visitado) {  1 usage  AlextyrB
        if (r < 0 || r >= filas || c < 0 || c >= columnas) return false;
        if (laberinto[r][c] == '1') return false;
        if (visitado[r][c]) return false;
        return true;
    }
    public char[][] getLaberinto() { return laberinto; }

    public void imprimirLaberinto() {  1 usage  AlextyrB
        if (laberinto == null) return;
        System.out.println("\n--- Representación del Laberinto Cargado (" + filas + "x" + columnas + ") ---");
        for (int i = 0; i < filas; i++) {
            for (int j = 0; j < columnas; j++) {
                System.out.print(laberinto[i][j]);
            }
            System.out.println();
        }
        System.out.println("------------------------------------------------------------------");
    }
```

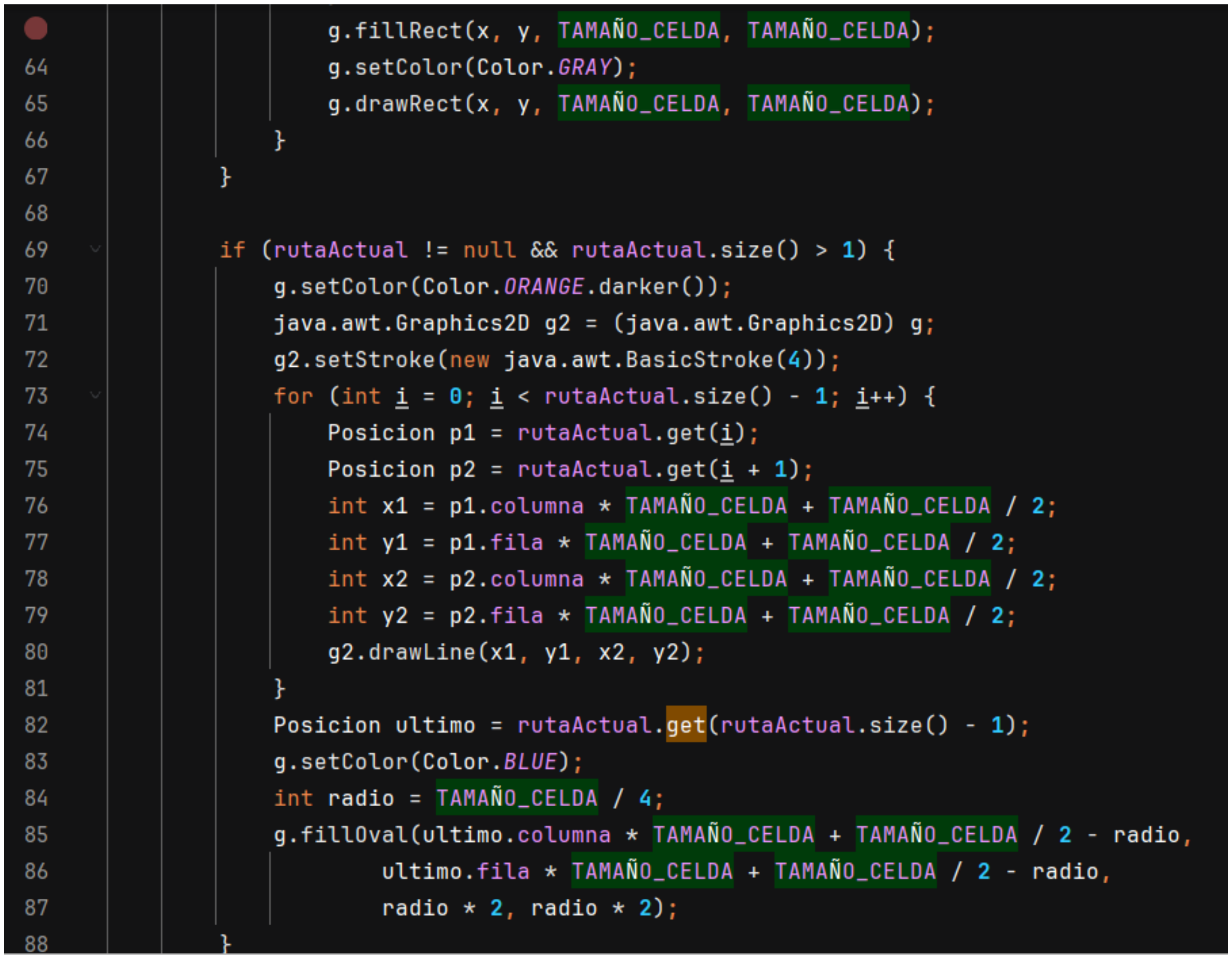
Vista Grafica



```java
import ...

class VistaLab extends JPanel implements ActionListener {  4 usages  AlextyrB *
    private final char[][] laberinto;  5 usages
    private final List<List<Posicion>> historialRuta;  3 usages
    private final int TAMAÑO_CELDA = 30;  21 usages
    private Timer timer;  3 usages
    private int indicePaso = 0;  3 usages
    private static final int DELAY_MS = 150;  1 usage

    private List<Posicion> rutaActual = new ArrayList<>();  8 usages

    public VistaLab(char[][] laberinto, List<List<Posicion>> historialRuta) {  2 usages  AlextyrB
        this.laberinto = laberinto;
        this.historialRuta = historialRuta;
        int filas = laberinto.length;
        int columnas = laberinto[0].length;
        this.setPreferredSize(new java.awt.Dimension(columnas * TAMAÑO_CELDA, filas * TAMAÑO_CELDA));
        if (historialRuta != null && !historialRuta.isEmpty()) {
            timer = new Timer(DELAY_MS, this);
            timer.start();
        }
    }
    @Override  AlextyrB
    public void actionPerformed(ActionEvent e) {
        if (indicePaso < historialRuta.size()) {
            rutaActual = historialRuta.get(indicePaso);
            indicePaso++;
```

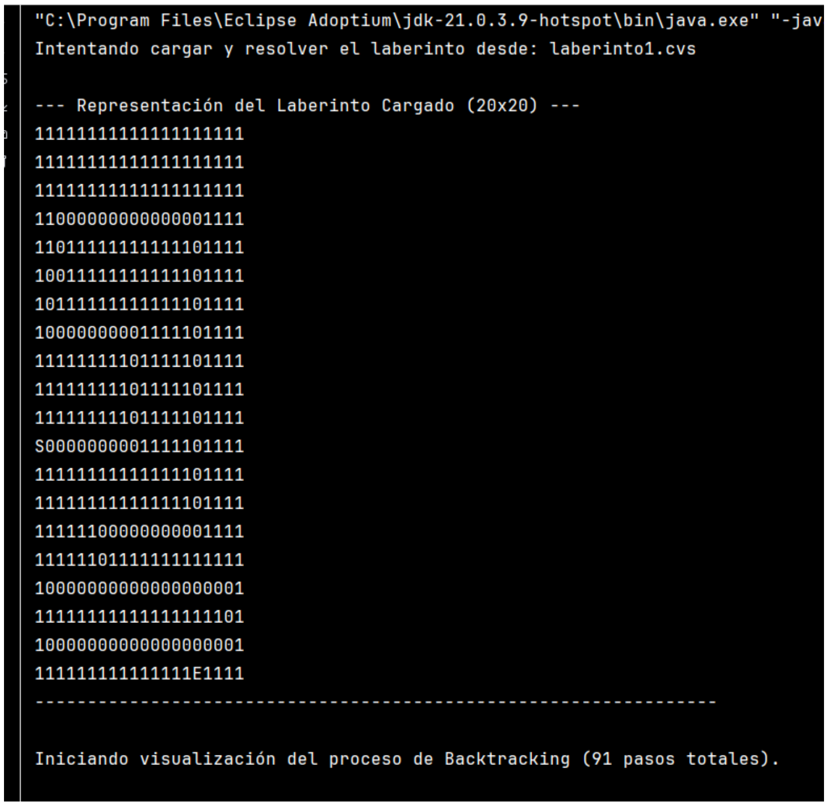


```java
            repaint();
        } else {
            timer.stop();
        }
    }

    @Override  AlextyrB *
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        if (laberinto == null) return;

        int filas = laberinto.length;
        int columnas = laberinto[0].length;
        for (int r = 0; r < filas; r++) {
            for (int c = 0; c < columnas; c++) {
                int x = c * TAMAÑO_CELDA;
                int y = r * TAMAÑO_CELDA;

                switch (laberinto[r][c]) {
                    case '1': g.setColor(Color.BLACK); break;
                    case '0': g.setColor(Color.WHITE); break;
                    case 'E': g.setColor(Color.BLUE); break;
                    case 'S': g.setColor(Color.RED); break;
                    default: g.setColor(Color.LIGHT_GRAY);
                }
```

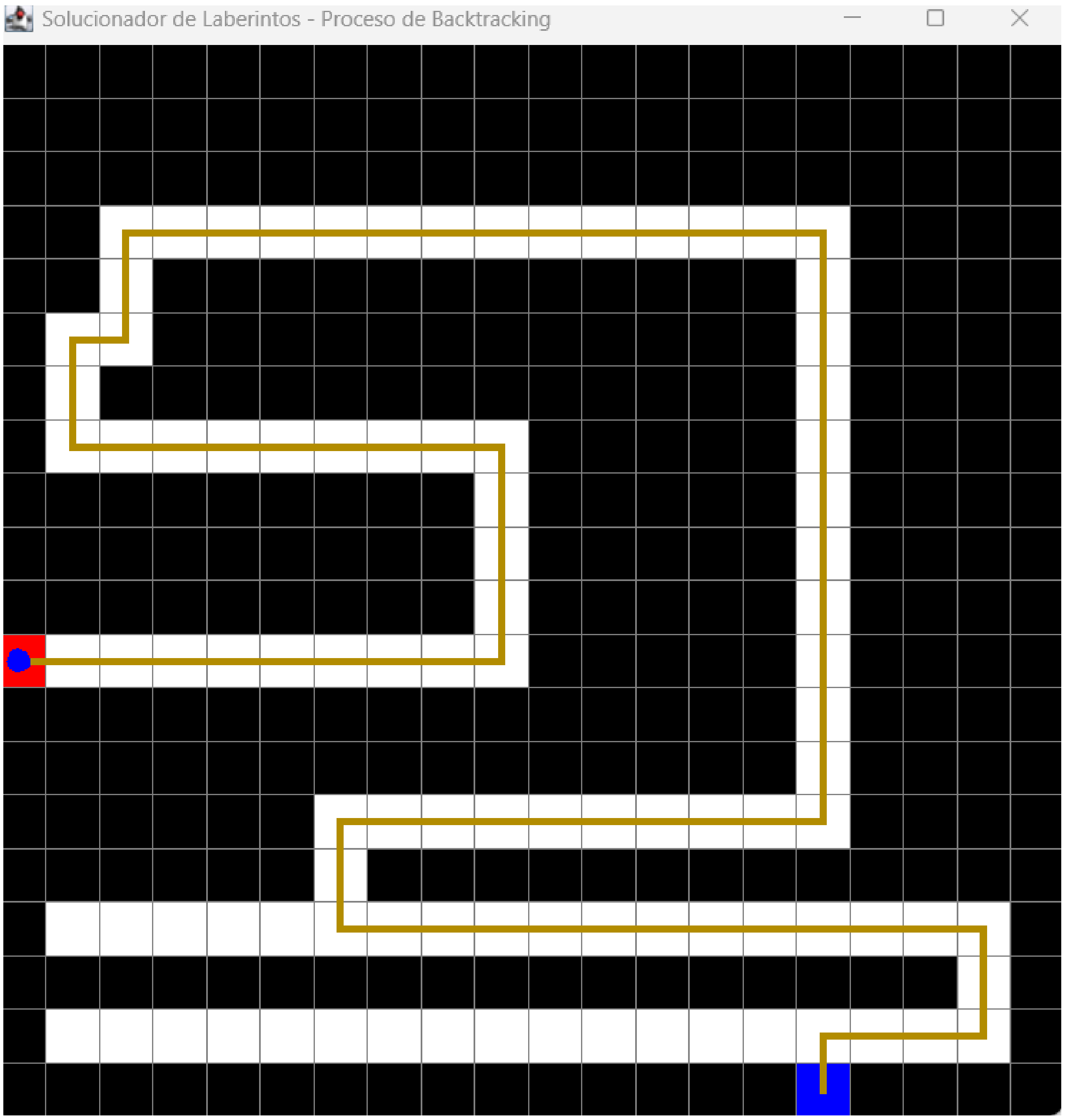


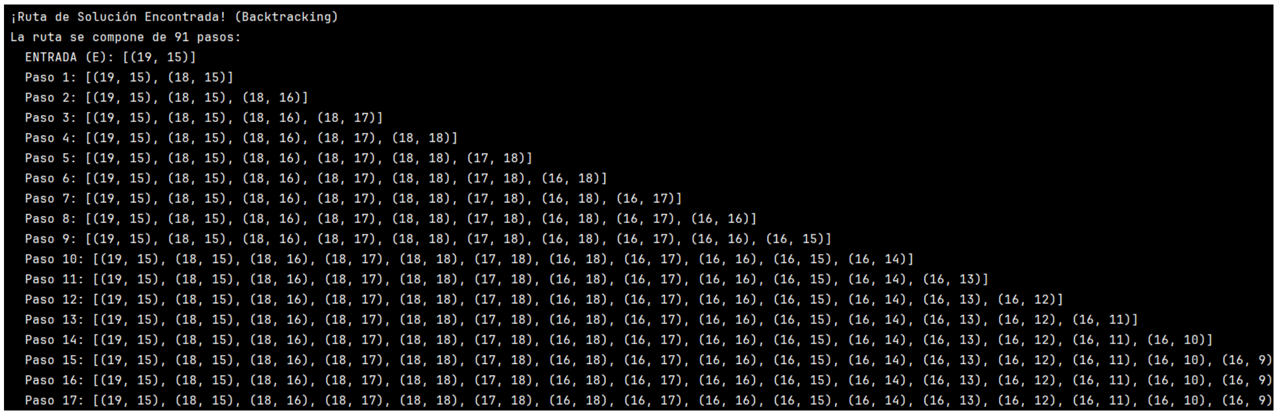
```java
                g.fillRect(x, y, TAMAÑO_CELDA, TAMAÑO_CELDA);
                g.setColor(Color.GRAY);
                g.drawRect(x, y, TAMAÑO_CELDA, TAMAÑO_CELDA);
            }
        }

        if (rutaActual != null && rutaActual.size() > 1) {
            g.setColor(Color.ORANGE.darker());
            java.awt.Graphics2D g2 = (java.awt.Graphics2D) g;
            g2.setStroke(new java.awt.BasicStroke(4));
            for (int i = 0; i < rutaActual.size() - 1; i++) {
                Posicion p1 = rutaActual.get(i);
                Posicion p2 = rutaActual.get(i + 1);
                int x1 = p1.columna * TAMAÑO_CELDA + TAMAÑO_CELDA / 2;
                int y1 = p1.fila * TAMAÑO_CELDA + TAMAÑO_CELDA / 2;
                int x2 = p2.columna * TAMAÑO_CELDA + TAMAÑO_CELDA / 2;
                int y2 = p2.fila * TAMAÑO_CELDA + TAMAÑO_CELDA / 2;
                g2.drawLine(x1, y1, x2, y2);
            }
            Posicion ultimo = rutaActual.get(rutaActual.size() - 1);
            g.setColor(Color.BLUE);
            int radio = TAMAÑO_CELDA / 4;
            g.fillOval(ultimo.columna * TAMAÑO_CELDA + TAMAÑO_CELDA / 2 - radio,
                    ultimo.fila * TAMAÑO_CELDA + TAMAÑO_CELDA / 2 - radio,
                    radio * 2, radio * 2);
        }
```

Solución de laberinto (1)

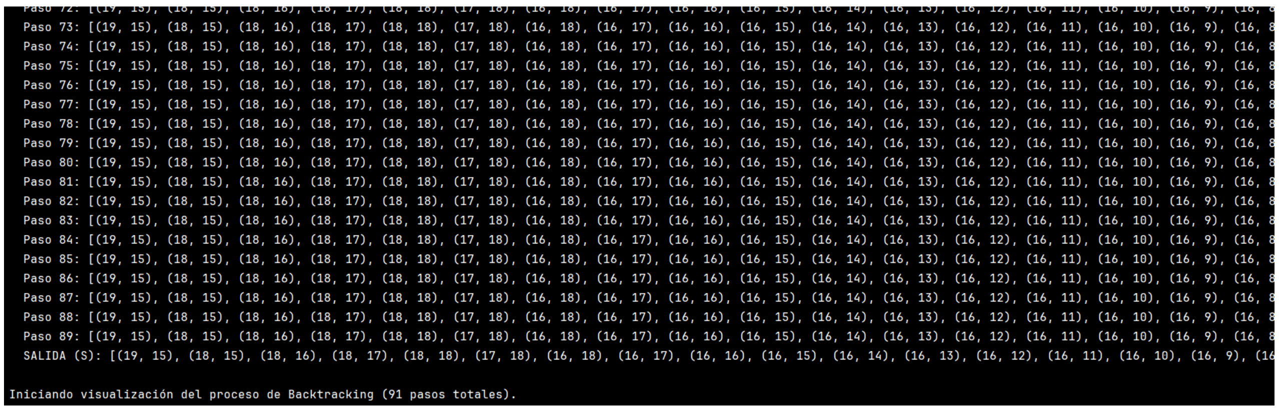


```
"C:\Program Files\Eclipse Adoptium\jdk-21.0.3.9-hotspot\bin\java.exe" "-jav
Intentando cargar y resolver el laberinto desde: laberinto1.cvs

--- Representación del Laberinto Cargado (20x20) ---
11111111111111111111
11111111111111111111
11111111111111111111
11000000000000001111
11011111111111101111
10011111111111101111
10111111111111101111
10000000001111101111
11111111101111101111
11111111101111101111
11111111101111101111
S0000000001111101111
11111111111111101111
11111111111111101111
11111000000000001111
11111011111111111111
10000000000000000001
11111111111111111101
10000000000000000001
111111111111111E1111
-----------------------------------------------------------------

Iniciando visualización del proceso de Backtracking (91 pasos totales).
```



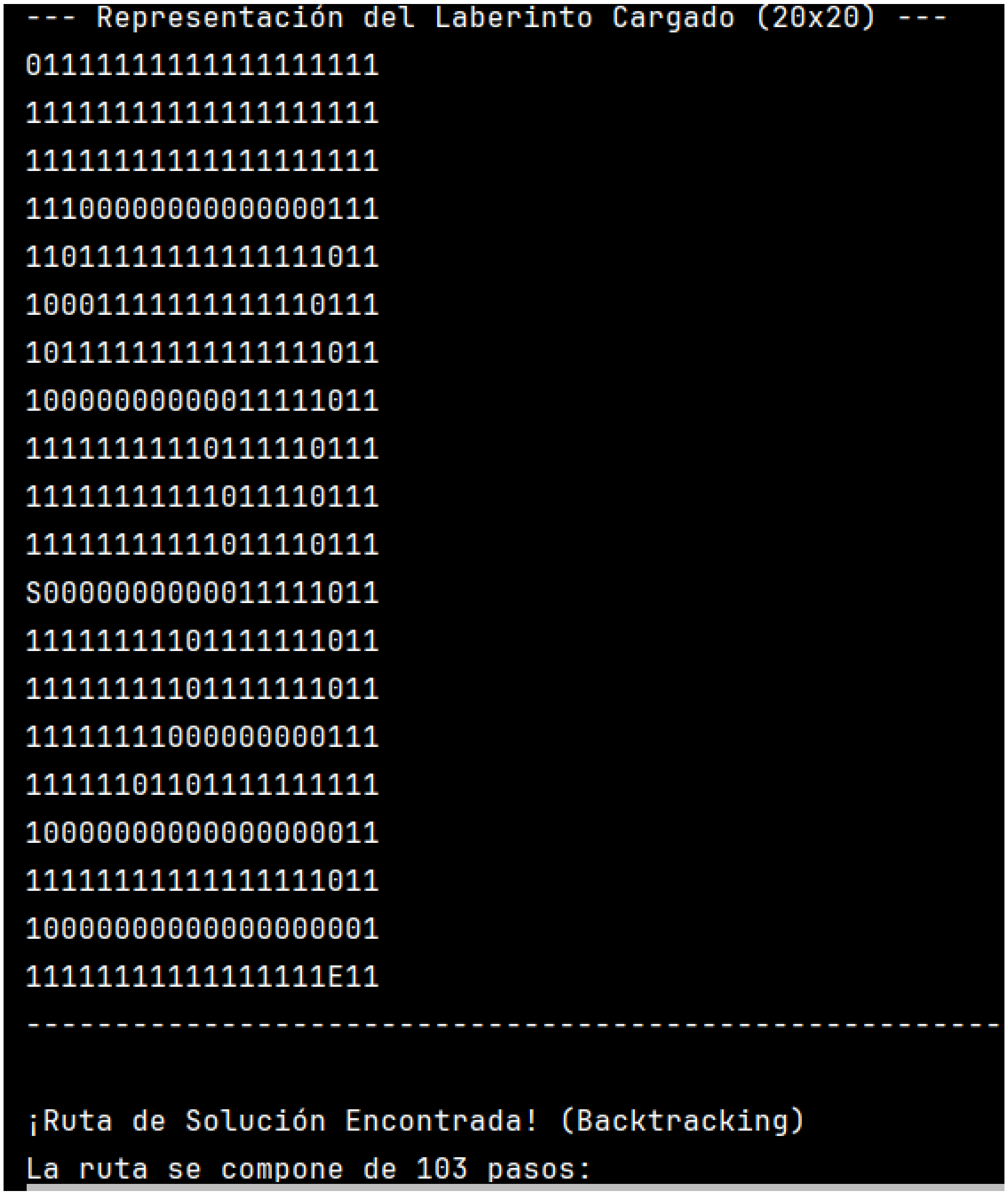


```
¡Ruta de Solución Encontrada! (Backtracking)
La ruta se compone de 91 pasos:
  ENTRADA (E): [(19, 15)]
  Paso 1: [(19, 15), (18, 15)]
  Paso 2: [(19, 15), (18, 15), (18, 16)]
  Paso 3: [(19, 15), (18, 15), (18, 16), (18, 17)]
  Paso 4: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18)]
  Paso 5: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18)]
  Paso 6: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18)]
  Paso 7: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17)]
  Paso 8: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16)]
  Paso 9: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15)]
  Paso 10: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14)]
  Paso 11: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13)]
  Paso 12: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12)]
  Paso 13: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11)]
  Paso 14: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10)]
  Paso 15: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9)
  Paso 16: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9)
  Paso 17: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9)
```

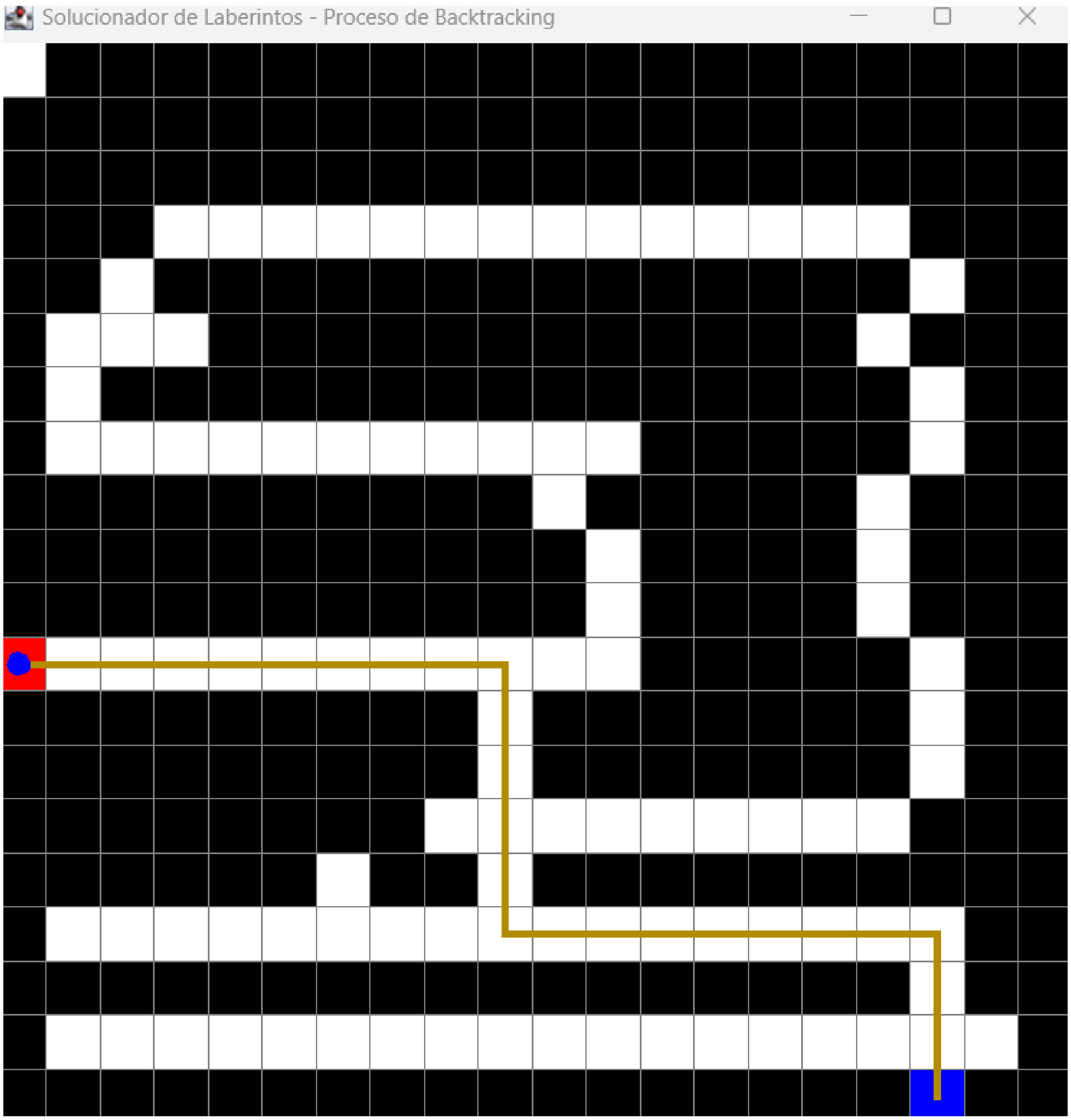


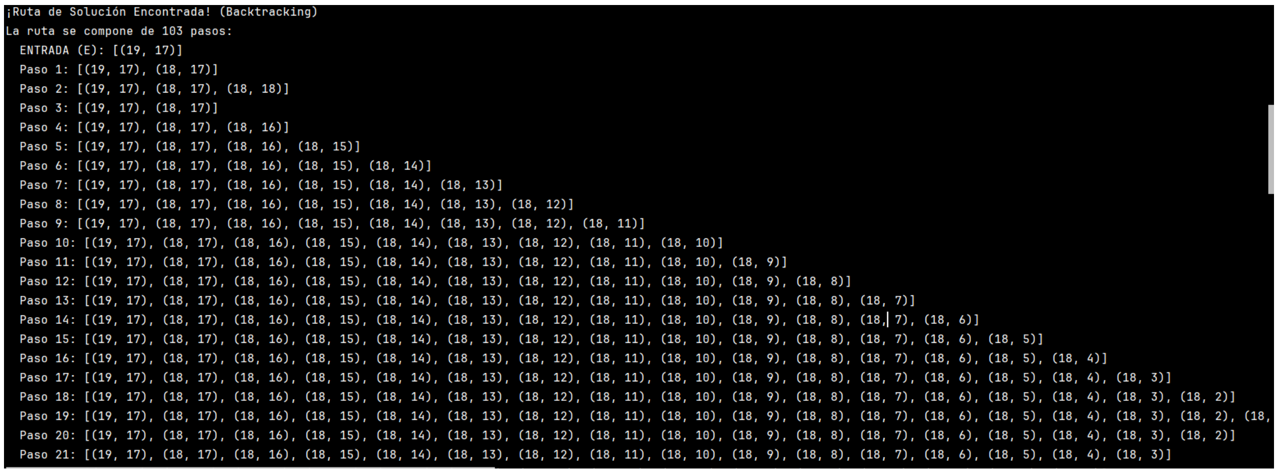
```
  Paso 72: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 
  Paso 73: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 74: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 75: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 76: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 77: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 78: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 79: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 80: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 81: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 82: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 83: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 84: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 85: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 86: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 87: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 88: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  Paso 89: [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8
  SALIDA (S): [(19, 15), (18, 15), (18, 16), (18, 17), (18, 18), (17, 18), (16, 18), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16

Iniciando visualización del proceso de Backtracking (91 pasos totales).
```

Solución de laberinto diferente(2)

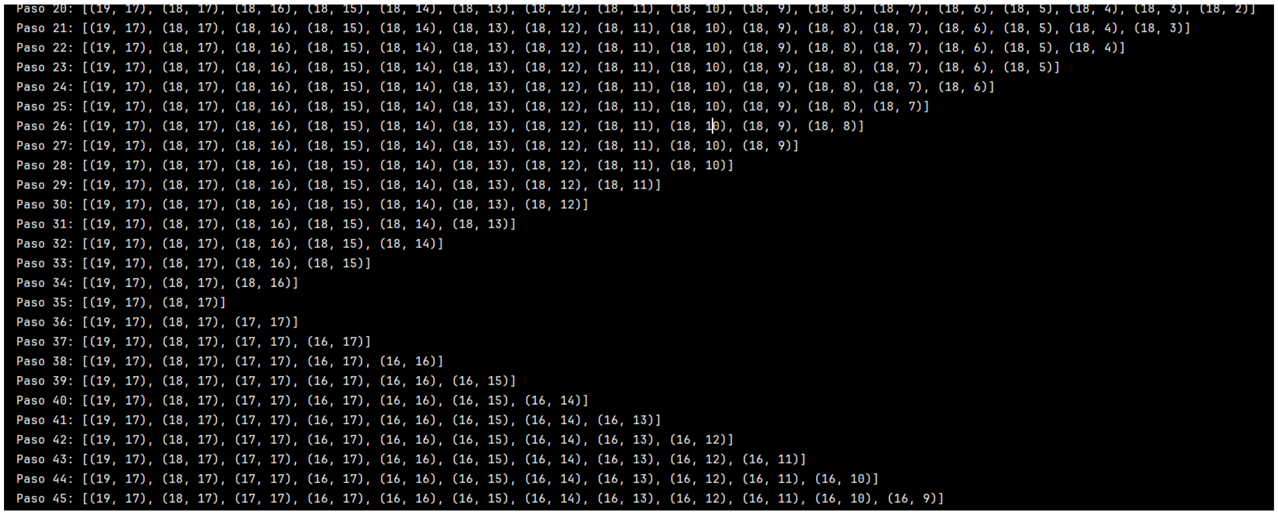


```
--- Representación del Laberinto Cargado (20x20) ---
01111111111111111111
11111111111111111111
11111111111111111111
11100000000000000111
11011111111111111011
10001111111111110111
10111111111111111011
10000000000001111011
11111111110111110111
11111111111011110111
11111111111011110111
S0000000000001111011
11111111101111111011
11111111101111111011
11111111000000000111
11111011011111111111
10000000000000000011
11111111111111111011
10000000000000000001
11111111111111111E11
-------------------------------------------------------


¡Ruta de Solución Encontrada! (Backtracking)
La ruta se compone de 103 pasos:
```

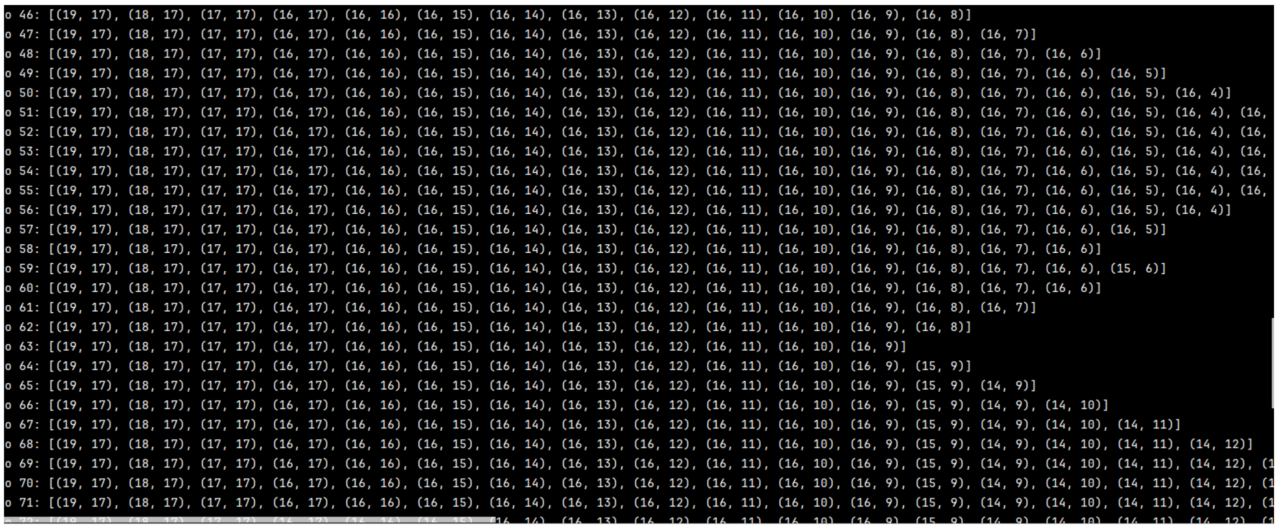




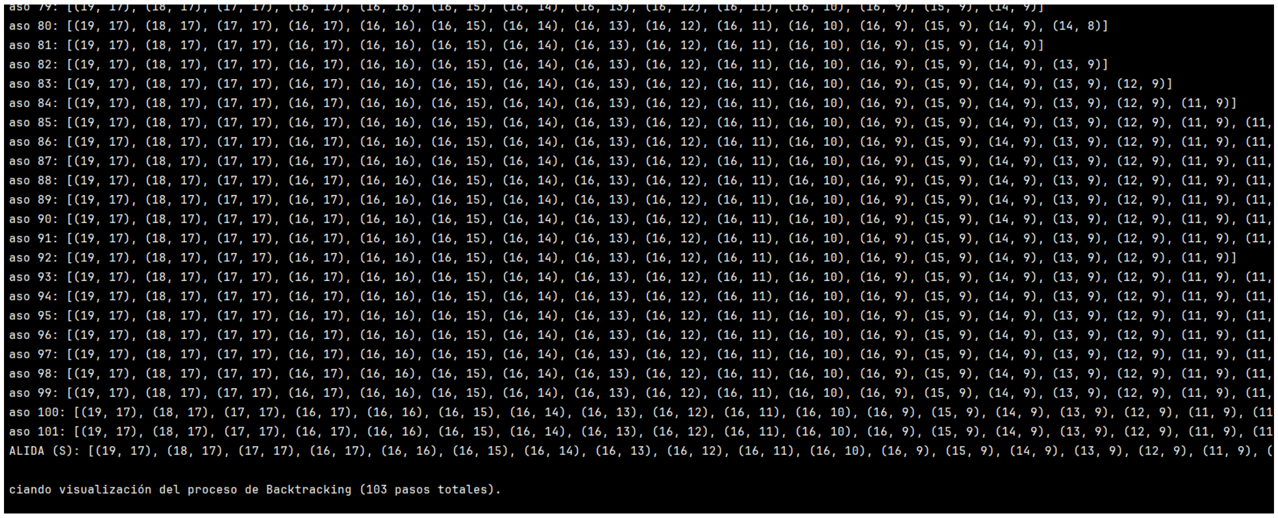
```
¡Ruta de Solución Encontrada! (Backtracking)
La ruta se compone de 103 pasos:
  ENTRADA (E): [(19, 17)]
  Paso 1: [(19, 17), (18, 17)]
  Paso 2: [(19, 17), (18, 17), (18, 18)]
  Paso 3: [(19, 17), (18, 17)]
  Paso 4: [(19, 17), (18, 17), (18, 16)]
  Paso 5: [(19, 17), (18, 17), (18, 16), (18, 15)]
  Paso 6: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14)]
  Paso 7: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13)]
  Paso 8: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12)]
  Paso 9: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11)]
  Paso 10: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10)]
  Paso 11: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9)]
  Paso 12: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8)]
  Paso 13: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7)]
  Paso 14: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6)]
  Paso 15: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5)]
  Paso 16: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4)]
  Paso 17: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3)]
  Paso 18: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3), (18, 2)]
  Paso 19: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3), (18, 2), (18,
  Paso 20: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3), (18, 2)]
  Paso 21: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3)]
```



```
 Paso 20: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3), (18, 2)]
 Paso 21: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4), (18, 3)]
 Paso 22: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5), (18, 4)]
 Paso 23: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6), (18, 5)]
 Paso 24: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7), (18, 6)]
 Paso 25: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8), (18, 7)]
 Paso 26: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9), (18, 8)]
 Paso 27: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10), (18, 9)]
 Paso 28: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11), (18, 10)]
 Paso 29: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12), (18, 11)]
 Paso 30: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13), (18, 12)]
 Paso 31: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14), (18, 13)]
 Paso 32: [(19, 17), (18, 17), (18, 16), (18, 15), (18, 14)]
 Paso 33: [(19, 17), (18, 17), (18, 16), (18, 15)]
 Paso 34: [(19, 17), (18, 17), (18, 16)]
 Paso 35: [(19, 17), (18, 17)]
 Paso 36: [(19, 17), (18, 17), (17, 17)]
 Paso 37: [(19, 17), (18, 17), (17, 17), (16, 17)]
 Paso 38: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16)]
 Paso 39: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15)]
 Paso 40: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14)]
 Paso 41: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13)]
 Paso 42: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12)]
 Paso 43: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11)]
 Paso 44: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10)]
 Paso 45: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9)]
```



```
o 46: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8)]
o 47: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7)]
o 48: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6)]
o 49: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5)]
o 50: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4)]
o 51: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4), (16,
o 52: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4), (16,
o 53: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4), (16,
o 54: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4), (16,
o 55: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4), (16,
o 56: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5), (16, 4)]
o 57: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (16, 5)]
o 58: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6)]
o 59: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6), (15, 6)]
o 60: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7), (16, 6)]
o 61: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8), (16, 7)]
o 62: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (16, 8)]
o 63: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9)]
o 64: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9)]
o 65: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9)]
o 66: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10)]
o 67: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10), (14, 11)]
o 68: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10), (14, 11), (14, 12)]
o 69: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10), (14, 11), (14, 12), (1
o 70: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10), (14, 11), (14, 12), (1
o 71: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10), (14, 11), (14, 12), (1
o 72: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 10), (14, 11), (14, 12), (1
```



```
aso 79: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9)]
aso 80: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (14, 8)]
aso 81: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9)]
aso 82: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9)]
aso 83: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9)]
aso 84: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9)]
aso 85: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 86: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 87: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 88: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 89: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 90: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 91: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 92: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9)]
aso 93: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 94: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 95: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 96: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 97: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 98: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 99: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11,
aso 100: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11
aso 101: [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (11
ALIDA (S): [(19, 17), (18, 17), (17, 17), (16, 17), (16, 16), (16, 15), (16, 14), (16, 13), (16, 12), (16, 11), (16, 10), (16, 9), (15, 9), (14, 9), (13, 9), (12, 9), (11, 9), (

ciando visualización del proceso de Backtracking (103 pasos totales).
```