Prácticas.

Sudoku

ALGORÍTMICA

VERSIÓN 0.1

AUTOR: Mª VICTORIA SANTIAGO ALCALÁ

TABLA DE CONTENIDO

ÍNDICE

PRÁCTICA. SUDOKU

1. Problema del sudoku

Descripción

Se nos presenta el problema de resolver un sudoku con backtracking el cual consiste en rellenar unas casillas divididas en celdas de 9x9 las cuales estan colocadas en grupos más pequeños de 3x3.

Estas celdas contienen números del 1 al 9 con la condición de que en cada subgrupo no se puede repetir el numero, lo mismo ocurre en las filas y en las columnas.

Inicialmente, el sudoku está inicializado con algunas celdas las cuales nos permiten poder empezar a resolver el problema.

Otra de las condiciones iniciales es que no se puede borrar ni sobrescribir una celda que ya está inicializada a algún número.

Backtracking

El código implementado está desarrollado en el lenguaje de programación java.

Se encuentra comentado con la explicación de que hace cada parte de este.

2. Implementación

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.InputStreamReader;
import java.io.LineNumberReader;
import java.io.PrintStream;

public final class SudokuBacktracking
{
	/**
```

```
	 * tablero almacena la información original de la plantilla
	 * (Utilizada para comprobar donde hay un elemento inamovible)
	 */
	private int[] tablero;

	/**
	 * Tablero es donde se calcula la solución al sudoku
	 */
	private int[] tableroTemp;

	/**
	 * nivel actual de avance de la solución
	 */
	private int nivel = 0;

	/**
	 * fin es establecerá a verdad cuando se resuelva el sudoku,
	 * lo que provocará el fin del algoritmo
	 */
	private boolean fin = false;


	public SudokuBacktracking(String entrada) throws ArrayIndexOutOfBoundsException
	{
		if(entrada.length() < 81)
			throw new ArrayIndexOutOfBoundsException("La entrada no contiene un sudoku
completo");

		tablero = new int[81];
		tableroTemp = new int[81];
		for(int i = 0; i < 81; i++)
		{
			if(entrada.charAt(i) != '.')
			{
				tableroTemp[i] = tablero[i] = Integer.parseInt(entrada.substring(i, i+1));
			}
		}
```

```
}

public void pintaTablero()
{
	PrintStream out = System.out;

	System.out.println("------------------------");
	for(int i = 0; i < 9; i++)
	{
		for(int j = 0; j < 9; j++)
		{
			out.print(tableroTemp[(i * 9) + j]+" ");

			if((j + 1) % 3 == 0)
				out.print("| ");
		}

		if((i + 1) % 3 == 0)
			System.out.println("\n------------------------");
		else
			System.out.println();
	}

	System.out.println();
}

public void resolverBacktracking()
{
	/**
	 *  Elementos ya colocados por la plantilla
	 *  no son editables
	 */
	while(tablero[nivel] > 0)
		nivel++;

	do
	{
```

```
                generar();

                if(solucion())
                        fin = true;
                else if(criterio())
                {
                        incrementarNivel();
                }
                else
                {
                        while(!masHermanos())
                                retroceder();
                }
        }
        while(!fin);
}

private void incrementarNivel()
{
        nivel++;

        /**
         *  Evitamos sobreescribir elementos ya colocados por la plantilla
         */
        while(nivel < 81 && tablero[nivel] > 0)
                nivel++;

        if(nivel > 80)
                fin = true;
}

private void generar()
{
        tableroTemp[nivel] += 1;
}

private boolean solucion()
```

```
{
	return nivel == 80;
}

private boolean criterio()
{
	/**
	 * Comprobamos si el valor está en el mismo cuadrante
	 */
	int cuadrantei = (nivel / 3) / 9;
	int cuadrantej = (nivel % 9) / 3;
	int [] utilizados = new int[9];
	for(int i = 0; i < 3; i++)
	{
		for(int j = 0; j < 3; j++)
		{
			int pos = (cuadrantei * 27) + (cuadrantej * 3) + (i * 9) + j;
			int valor = tableroTemp[pos];
			if(valor > 0)
			{
				if(utilizados[valor - 1] > 0)
					return false;

				utilizados[valor - 1] = 1;
			}
		}
	}

	/**
	 * Comprobamos si el valor está en la misma fila y/o columna
	 */
	int comprobacionFila [] = new int[9];
	int comprobacionColumna [] = new int[9];

	int fila = nivel / 9;
	int columna = nivel % 9;
```

```
		for(int i = 0; i < 9; i++)
		{
			int valorFila = tableroTemp[(fila * 9) + i];

			if(valorFila > 0)
			{
				if(comprobacionFila[valorFila - 1] > 0)
				{
					return false;
				}
				comprobacionFila[valorFila - 1] = 1;
			}

			int valorColumna = tableroTemp[columna + (9 * i)];

			if(valorColumna > 0)
			{
				if(comprobacionColumna[valorColumna - 1] > 0)
				{
					return false;
				}
				comprobacionColumna[valorColumna - 1] = 1;
			}
		}

		return true;
	}

	private boolean masHermanos()
	{
		return tableroTemp[nivel] < 9;
	}

	public void retroceder()
	{
		tableroTemp[nivel] = 0;
		nivel--;
```

```
		/**
		 * Retrocedemos también sobre aquellas posiciones
		 * prefijadas por la plantilla
		 */
		while(tablero[nivel] > 0)
			nivel--;
	}

	//
############################################################################
#########################
	//
############################################################################
#########################
	// FUNCIONES AUXILIARES PARA LANZAR EL PROGRAMA / LEER ENTRADA DEL USUARIO

	public static void main(String [] args) throws Exception
	{
		if(args.length < 1)
		{
			System.out.println("Uso: java SudokuBacktracking <fichero_plantilla_sudokus>");
			return;
		}

		File plantilla = new File(args[0]);
		if(!plantilla.exists())
		{
			System.out.println("No se puede hallar el archivo plantilla
"+plantilla.getAbsolutePath());
			return;
		}

		BufferedReader br = new BufferedReader(new FileReader(plantilla));
		String ln;
		int opcion = 1;
		while((ln = br.readLine()) != null && opcion > 0)
```

```
		{

			opcion = menuOpcion();
			SudokuBacktracking sudoku = null;
			switch(opcion)
			{
				case 1:
					try
					{
						sudoku = new SudokuBacktracking(ln);
						sudoku.pintaTablero();
						sudoku.resolverBacktracking();
						sudoku.pintaTablero();
					}
					catch(Exception e)
					{
						if(sudoku != null)
							System.out.println(sudoku.nivel);
						System.err.println("Entrada de sudoku inválida");
						e.printStackTrace();
					}
					break;
			}
		}

		br.close();
	}

	private static int menuOpcion()
	{
		System.out.println("Que hacer:");
		System.out.println("1 - Resolver el siguiente sudoku del fichero");
		System.out.println("0 - Salir");

		int opcion = -1;
		LineNumberReader reader = new LineNumberReader(new
InputStreamReader(System.in));
```

```
                boolean repetir = true;
                do
                {
                        try
                        {
                                opcion = Integer.parseInt(reader.readLine());
                        }
                        catch(Exception e)
                        {
                        }
                        finally
                        {
                                if(opcion < 0 || opcion > 1)
                                        System.out.println("Opción inválida");
                                else
                                        repetir = false;
                        }
                }
                while(repetir);

                return opcion;
        }
}
```

BIBLIOGRAFÍA

ENLACES:

(1) **Web de la asignatura:** http://cvg.ugr.es/docencia
(2)**decsai.ugr.es**