PROGRAMACIÓN Y ESTRUCTURAS DE DATOS AVANZADAS

PRÁCTICA 1

Luis Daniel Guardado Garcia
77175422A
Lguardado7@alumno.uned.es

CUESTIONES TEÓRICAS DE LA PRÁCTICA

1) Analiza el coste computacional del algoritmo.

Al tratarse de algoritmos recursivos, es necesario plantear la ecuación de recurrencia. Al estar dividiendo el problema, normalmente se podrá aplicar la formula para reducciones por división:

$$T(n) = \begin{cases} cn^k, & si\ 1 \leq n < b \\ aT\left(\frac{n}{b}\right) + cn^k, & si\ n \geq b \end{cases}$$

$$f(n) = \begin{cases} \theta(n^k), & si\ a < b^k \\ \theta(n^k \log n), & si\ a = b^k \\ \theta(n^{log_b a}), & si\ a > b^k \end{cases}$$

Por lo tanto, hay que plantear la ecuación de recurrencia para $T(n)$ e identificar en ella las constantes a, b y k.

2) Describe alternativas al esquema utilizado, si las hay, y compara su coste con el de la solución realizada.

La única alternativa que se me ocurre es utilizando el algoritmo programación dinámica ya que es el único que puede descomponer el problema en subproblemas de tamaño menor del mismo tipo para llegar a la solución. Su coste es el siguiente:

$$M(i,j,k) = \begin{cases} 0, & si\ k = 0\ y\ i = j \\ A[i,j], & si\ k = 0\ y\ i \neq j \\ \min\ (M(i,j,k-1), & si\ k > 0 \\ M(i,k,k-1) + M(k,j,j-1)) & \end{cases}$$

El coste temporal de este algoritmo, con los tres bucles anidados, está en $\theta(N^3)$, mientras el coste espacial está en $\theta(N^2)$.

Diferencia entre ambos:

1. Divide y vencerás a menudo utiliza cálculos recursivos, cálculos de arriba hacia abajo y la programación dinámica calcula directamente de abajo hacia arriba

2. Los pequeños problemas de divide y vencerás pueden calcularse repetidamente en el proceso de recursión. Los pequeños problemas en la programación dinámica se almacenan después del cálculo y se llamarán directamente cuando se encuentren la próxima vez.

3. La solución al pequeño problema del método de divide y vencerás se usa solo una vez, y la solución del pequeño problema de la programación dinámica se almacena para llamadas repetidas al resolver el gran problema

Condiciones generales de construcción de la programación dinámica:

1. Existe una subestructura óptima: la solución óptima en el siguiente nivel puede usarse como base para la solución óptima en el nivel anterior

2. Hay subproblemas duplicados: muchos subproblemas se llamarán varias veces (eficiencia)

CÓDIGO COMPLETO

```
import java.io.*;
import java.util.*;

public class skyline {

    //DyV
    public List<int[]> dyv(int[][] problema)
    {
        System.out.println(" ========== DYV ==========");
        System.out.println();
        System.out.println(" Se detecta " + problema.length + " edificios");
        System.out.println(" Pasamos al método edificios");
        System.out.println();

        if (problema.length == 0)
            return new ArrayList<int[]>();
        System.out.println(" ========== EDIFICIOS ==========");
        System.out.println();
        return edificios(problema, 0, problema.length - 1);
    }


    //Edificios
    private ArrayList<int[]> edificios(int[][] C, int i, int j)
    {
        int m, n;
        m = (i+j-1)/2;
        n = j-i+1;

        if (n == 1)
        {
            ArrayList<int[]> s = new ArrayList<int[]>();
            s.add(new int[] { C[i][0], C[i][2] });
            s.add(new int[] { C[i][1], 0 });
            System.out.println("    Obtenemos skyline del edificio " + (i+1)
+ ": " + "(" + C[i][0] + "," + C[i][2] + "," + C[i][1] + "," + 0 +")");
            return s;
        }
```

```
        else
        {
            System.out.println(" Se divide el array en 2 hasta obtener dos
edificios: " + n);
            ArrayList<int[]> sa = edificios(C,i,m);
            ArrayList<int[]> sb = edificios(C, m + 1, j);
            return combinar(sa, sb);
        }
    }


    //Combinar
    private ArrayList<int[]> combinar(ArrayList<int[]> sa, ArrayList<int[]>
sb)
    {
        System.out.println();
        System.out.println(" Pasamos al método combinar");
        System.out.println();
        System.out.println(" ========== COMBINAR ==========");
        System.out.println();
        ArrayList<int[]> s = new ArrayList<>();
        int ha = 0, hb = 0, uh = 0;
        int a[],b[];
        int ia, ib, nx, nh;

        ia = 0;
        ib = 0;

        System.out.println(" Combinamos los " + sa.size() + " skylines");

        while (ia < sa.size() && ib < sb.size())
        {
            a = sa.get(ia);
            b = sb.get(ib);

            if(a[0] == b[0])
            {
                nx = a[0];
                nh = Math.max(a[1], b[1]);
                ha = a[1];
                hb = b[1];
                ia = ia + 1;
                ib = ib + 1;
            }
            else
            {
                if (a[0] < b[0])
                {
                    nx = a[0];
                    nh = Math.max(a[1], hb);
                    ha = a[1];
                    ia = ia + 1;
                }
                else
                {
```

```
                    nx = b[0];
                    nh = Math.max(b[1], ha);
                    hb = b[1];
                    ib = ib + 1;
                }
            }
            if (uh != nh)
            {
                s.add(new int[] { nx, nh });
                uh = nh;
            }
        }
        while (ia < sa.size())
        {
            s.add(sa.get(ia++));
        }
        while (ib < sb.size())
        {
            s.add(sb.get(ib++));
        }
        for (int[] j : s)
        {
            System.out.println();
            System.out.print(Arrays.toString(j));
        }
        System.out.println();
        return s;
    }


    public static void ayuda(){
        System.out.println();
        System.out.println(" SINTAXIS: skyline [-t] [-h]  [fichero entrada]
[fichero salida]");
        System.out.println("      -t          Traza cada llamada recursiva y
sus parámetros");
        System.out.println("      -h          Muestra esta ayuda");
        System.out.println("      [fichero entrada]   Conjunto de edificios
de la ciudad");
        System.out.println("      [ficharo salida]   Secuencia que
representan el skyline de la ciudad");
        System.out.println();
    }

    public static void programa(String t, String h, String entrada, String
salida){

        try
        {
            String prueba;

            File fentrada = new File(entrada);
            File fsalida = new File(salida);

            if (fentrada.exists())
```

```
            {
                BufferedReader bufferedReader = new BufferedReader(new
FileReader(fentrada));
                while ((prueba = bufferedReader.readLine()) != null)
                {
                    prueba = prueba.replace("}","");
                    prueba = prueba.replace('{',' ');
                    prueba = prueba.replaceAll("\\(","");
                    prueba = prueba.replaceAll("\\),","&");
                    prueba = prueba.replaceAll("\\)","");
                    prueba = prueba.trim();

                    String[] row = prueba.split("&");
                    int [][] matrix = new int [row.length][];
                    for (int i = 0; i < row.length; i++)
                    {
                        String currentline = row[i].trim();
                        int [] temp = new int[currentline.length()-2];
                        int cont = 0;
                        for (int j = 0; j < currentline.length(); j = j+2)
                        {
                            temp[cont] =
Character.getNumericValue(currentline.charAt(j));
                            cont++;
                        }
                        matrix[i] = temp;
                    }
                    String cadena;
                    skyline ejemplo = new skyline();
                    cadena = "(";

                    for(int[] p: ejemplo.dyv(matrix))
                    {
                        for (int j: p)
                        {
                            if (j != 0)
                            {
                                cadena += j + ",";
                            }
                            else
                            {
                                cadena += j;
                            }
                        }
                    }

                    if (salida.equals("0"))
                    {
                        System.out.println();
                        System.out.print(cadena + ")");
                    }
                    else
                    {
                        if (fsalida.exists()){
                            System.out.println();
```

```
                            System.out.println(cadena + ")");
                        }
                        else
                        {
                            BufferedWriter bufferedWriter = new
BufferedWriter(new FileWriter(fsalida));
                            bufferedWriter.write(cadena + ")");
                            bufferedWriter.newLine();
                            bufferedWriter.close();
                        }

                    }
                }
            }
            else
            {
                System.out.println(" El fichero de entrada no existe");
            }
        }
        catch (IOException e)
        {
            System.out.println(" Error E/S: " + e);
        }
    }

    public static void main(String[] args)
    {
        if (args.length == 1)
        {
            File entrada = new File(args[0]);
            String h = args[0];
            String t = "t";
            if (h.equals("-h"))
            {
                ayuda();
            }
            else if (h.equals("-t"))
            {
                System.out.println(" Faltan más parámetros para que se pueda
ejecutar este comando");
            }
            else if (entrada.exists())
            {

                programa(h,t,h,"0");
            }
        }

        if (args.length == 2)
        {
            String t = "t";
            String h = "h";
            String fentrada = args[0];
            String fsalida = args[1];
```

```
            programa(t,h,fentrada,fsalida);
        }

        if (args.length > 4)
        {
            System.out.println(" Sobran parámetros");
        }

        if (args.length == 3)
        {
            String t = args[0];
            String h = args[1];
            String fentrada = args[2];
            String fsalida = "0";

            programa(t,h,fentrada,fsalida);
        }

        if (args.length == 4)
        {
            String t = args[0];
            String h = args[1];
            String fentrada = args[2];
            String fsalida = args[3];

            programa(t,h,fentrada,fsalida);
        }
    }
}
```