PRÁCTICA 1. ANÁLISIS Y DISEÑO DE ALGORITMOS. ANALIZADOR

María Peinado Toledo. Doble Grado Ingeniería Informática y Matemáticas

25/10/2021

```java
public class Analizador {

    public static String masCercano(double r) {
        if(r>0.95&&r<1.02) {
            return "1";
        }else if (0.8<=r&&r<1.18) {
            return "LOGN";
        }else if (1.18<=r&&r<2.09) {
            return "N";
        }else if (2.09<=r&&r<2.85) {
            return "NLOGN";
        }else if (2.86<=r&&r<6.5) {
            return "N2";
        }else if (6.5<=r&&r<11) {
            return "N3";
        }else {
            return "NF";
        }
    }
```

La primera función que nos encontramos en la clase Analizador es masCercano el cual recibe un double, que más adelante explicaré el significado de este. Esta función asocia este valor a las diferentes complejidades de un algoritmo proporcionado. Los valores se han obtenido tras varias pruebas con distintos algoritmos, tanto los que se proporcionan en la prueba del campus, como algoritmos diseñados y comprobados a mano.

```java
public static void ratios(int[] n1,int[] n2,boolean factorial, double[] res,Temporizador t) {
    double tiempo1,tiempo2;
    int suman1,suman2,i=0;

    while(i<n1.length&&!factorial) {
        suman1=0;
        suman2=0;
        t.iniciar();
        Algoritmo.f(n2[i]);
        t.parar();
        if(t.tiempoPasado()>6*Math.pow(10, 9)) {
            factorial=true;
        }else {

            for(int j=0;j<7;j++) {
                Algoritmo.f(n1[i]);
                t.reiniciar();
                t.iniciar();
                Algoritmo.f(n1[i]);
                t.parar();
                suman1+=t.tiempoPasado();

                Algoritmo.f(n2[i]);
                t.reiniciar();
                t.iniciar();
                Algoritmo.f(n2[i]);
                t.parar();
                suman2+=t.tiempoPasado();
            }

            tiempo1=suman1/7;
            tiempo2=suman2/7;
            res[i]=tiempo2/tiempo1;
            i++;
        }
    }
}
```

Ahora nos encontramos con el siguiente método llamado ratios, que recibe como valores los dos arrays con los distintos tamaños de entrada, un boolean para la complejidad factorial, un array de double donde se guardaran los resultados de la comparación de los tiempos y el temporizador.
Utilizamos un while para recorrer los distintos tamaños de entrada que termina cuando o bien termina de recorrer el array o se encuentra la complejidad factorial ya que ésta es mucho mayor y proporcionaría problemas en nuestra clase temporizador.
Lo primero que hace es comprobar que no sea factorial por lo ya mencionado. En el caso de que no lo sea prueba 8 veces el tiempo de duración del algoritmo para una misma entrada, tanto en el primer array como en el segundo y va sumando los valores en los valores suman1 y suman2 para después poder hacer la media y obtener un valor mucho más preciso. Por último, se obtiene la comparación dividiendo ambos resultados y se almacena en el array de las soluciones. Este proceso se hace para todos los tamaños enunciados en n1 y en n2.

```
public static void main(String arg[]) {

    boolean factorial=false;
    int[]n1= {1,2,4,6,10,12,200,400,800,1000,1400};
    int[]n2= new int[n1.length];

    for(int i=0;i<n1.length;i++) {
        n2[i]=2*n1[i];
    }

    double[] res=new double[n1.length];
    Temporizador t = new Temporizador();
    t.iniciar();
    Algoritmo.f(28);
    t.parar();

    if((double)t.tiempoPasado()>=(Math.pow(10, 9))) {
        System.out.println("2N");
    }else {
        ratios(n1,n2,factorial,res,t);

        double media = (res[n1.length/2]+res[n1.length-1])/2;

        if(factorial) {
            System.out.println("NF");
        }else {
            System.out.println(masCercano(media));
        }

    }

}
```

Para terminar, tenemos el main, donde primero se declaran todas las variables que van a ser utilizadas posteriormente. Se inicia el temporizador y primero se comprueba si su complejidad es de 2^n ya que es el que más tardará en calcularse. En el caso de que no sea 2^n se invoca la función ratio con los valores ya declarados. Se calcula una media con un valor intermedio del array de soluciones y otro casi final para obtener un valor preciso de la prueba y ya por último se escribe por pantalla la complejidad del algoritmo obtenida en la función masCercano proporcionándole el valor media calculado anteriormente.