Algoritmos http://www.lfcia.org/alg/ Octubre de 2007

Ingeniería Informática

Ingeniería Técnica de Informática de Gestión

Práctica 1

La **sucesión de Fibonacci** se define inductivamente del modo siguiente:

$$\text{fib(n)} = \begin{cases} 0 & \text{si n=0;} \\ 1 & \text{si n=1;} \\ \text{fib(n-1)+fib(n-2)} & \text{si n>=2.} \end{cases}$$

El objetivo de la práctica es comprobar empíricamente el análisis teórico de la eficiencia de tres algoritmos diferentes que permiten calcular esta sucesión, familiarizándose además con el problema de la medición de tiempos.

Algoritmo *fib1*: $O(\phi^n)$**,** $\phi = \frac{1+\sqrt{5}}{2}$

```
función fib1 (n);
    si n<2 entonces devolver n
    sino devolver fib1(n-1) + fib1(n-2)
    fin si
fin función
```

Algoritmo *fib2*: $O(n)$

```
función fib2 (n);
    i := 1; j := 0;
    para k := 1 hasta n hacer
        j := i + j; i := j - i
    fin para;
    devolver j
fin función
```

Algoritmo *fib3*: $O(\log n)$

```
función fib3 (n);
    i := 1; j := 0; k := 0; h := 1; t := 0
    mientras n>0 hacer
        si n es impar entonces
            t := jh;
            j := ih + jk + t;
            i := ik + t
        fin si;
        t := h^2;
        h := 2kh + t;
        k := k^2 + t;
        n := n div 2
    fin mientras;
    devolver j
fin función
```

1. Implemente en JAVA[1] los tres algoritmos y compare sus tiempos de ejecución, tomando como referencia para el cálculo de `Fib1` los valores: 2, 4, 8, 16 y 32; y para `Fib2` y `Fib3`: 1.000, 10.000, 100.000, 1.000.000 y 10.000.000 (nota: la sucesión de Fibonacci crece muy deprisa: fib(100) tiene ya 21 cifras decimales. Para efectos de esta práctica no es necesario tener en cuenta los problemas de desbordamiento).

2. Analice los resultados obtenidos realizando una comprobación empírica de la complejidad teórica. Igualmente se realizará una comprobación empírica utilizando un cota subestimada y otra sobreestimada para cada algoritmo:

   - En el caso del primero algoritmo use la función $f(n) = 1{,}1^n$ para la cota subestimada, y $f(n) = 2^n$ para la sobreestimada.
   - En el segundo algoritmo se proponen $f(n) = n^{0{,}8}$ y $f(n) = n\log(n)$.
   - Y en el tercero, $f(n) = \sqrt{(\log(n))}$ para la cota subestimada, y $f(n) = n^{0{,}5}$ para la sobreestimada.

[1] Realize la implementación tomando como referencia el código de las figuras 1 y 2.

```
public class Crono {
    private java.util.Date t1, t2;

    // iniciar() fija la hora del sistema
    public void iniciar() {
        t1 = new java.util.Date();
    }

    // tiempo() devuelve el tiempo (milisegs) transcurrido desde la
    // última inicialización
    public long tiempo() {
        t2 = new java.util.Date();
        return (t2.getTime() - t1.getTime());
    }
}
```

Figura 1: clase `Crono`

```
class Fib {
    static long fib1(long n) {
        if (n<2)
            return n;
        else
            return (fib1(n-1)+fib1(n-2));
    }
    static long fib2(long n) {
        long i=1, j=0, k;
        for (k=1;k<=n;k++) {
            j=i+j;
            i=j-i;
        }
        return j;
    }

    // Definición de fib3 ...

    public static void main(String args[]) {
        long t, x;
        Crono crono = new Crono();

        crono.iniciar();
        x = fib1(30);
        t = crono.tiempo();

        System.out.println("fib1(30)=" + x + " Tiempo = " + t + " ms");
    }
}
```

Figura 2: clase `Fib`