Algoritmos

Septiembre de 2014

Grado en Ingeniería Informática

Práctica 1

Fecha límite de entrega: lunes, 29 de septiembre

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 C (véanse las figuras 1 y 2) los tres algoritmos.
2. Valide que los algoritmos funcionan correctamente.
3. 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).
4. 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, $O(\phi^n)$ con $\phi = \frac{1+\sqrt{5}}{2}$, use la función $f(n) = 1{,}1^n$ para la cota subestimada, y $f(n) = 2^n$ para la sobreestimada.
   - En el segundo algoritmo $O(n)$ se proponen $f(n) = n^{0{,}8}$ y $f(n) = n\log(n)$ como cotas subestimada y sobreestimada.
   - Y en el tercero $O(\log n)$, use $f(n) = \sqrt{(}\log(n))$ para la cota subestimada, y $f(n) = n^{0{,}5}$ para la sobreestimada.
5. Deposite, desde alguna de las *máquinas de referencia*, en `/PRACTICAS/GEI/Alg/P1/` (existe un directorio para cada estudiante) los ficheros `C` y el informe con el estudio empírico de la complejidad. Consulte las *máquinas de referencia* disponible en la página `wiki.fic.udc.es`.

```c
#include <sys/time.h>
#include <stdio.h>

/* obtiene la hora actual en microsegundos */
double microsegundos() {
  struct timeval t;
  if (gettimeofday(&t, NULL) < 0 )
    return 0.0;
  return (t.tv_usec + t.tv_sec * 1000000.0);
}
```

Figura 1: Módulo del control de tiempo

```c
#include <stdio.h>
#include <math.h>

int fib3(int n) {
  int i, j, k, h, t;
  i = 1; j = 0; k = 0; h = 1;
  while ( n > 0 ) {
    if ((n % 2) != 0) {
      t = j * h;
      j = (i * h) + (j * k) + t;
      i = (i * k) + t;
    }
    t = h * h;
    h = (2 * k * h) + t;
    k = (k * k) + t;
    n = n / 2;
  }
  return j;
}

int main() {
  int n;
  double t1, t2, t, x, y, z;
  n=1000000;
  t1 = microsegundos();
  fib3(n);
  t2 = microsegundos();
  t = t2-t1;
  x = t / sqrt(log(n));
  y = t / log(n);
  z = t / pow(n, 0.5);
  printf("%12d%15.3f%15.6f%15.6f%15.6f\n", n, t, x, y, z);
}
```

Figura 2: función `fib3`