Algorithms

September 2022

Degree in Computer Science Engineering

Practical 1

Submission deadline: Friday, 23 September, 23:59

The **Fibonacci sequence** is defined inductively as follows:

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

The goal is to empirically verify the theoretical analysis of the efficiency of three different algorithms that can be used to calculate this sequence, while becoming familiar with the problem of measuring times.

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

```
function fib1 (n);
    if n<2 then return n
    else return fib1(n-1) + fib1(n-2)
    end if
end function
```

Algorithm *fib2*: $O(n)$

```
function fib2 (n);
    i := 1; j := 0;
    for k := 1 to n do
        j := i + j; i := j - i
    end for;
    return j
end function
```

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

```
function fib3 (n);
    i := 1; j := 0; k := 0; h := 1; t := 0
    while n>0 do
        if n is odd then
            t := jh;
            j := ih + jk + t;
            i := ik + t
        end if;
        t := h^2;
        h := 2kh + t;
        k := k^2 + t;
        n := n div 2
    end while;
    return j
end function
```

1. Implement in C (see Figures 1 and 2) the three algorithms.
2. Validate that the algorithms work correctly.
3. Compare their execution times, taking as a reference for the calculation of `Fib1` the values: 2, 4, 8, 16 and 32; and for `Fib2` and `Fib3`: 1,000, 10,000, 100,000, 1,000,000 and 10,000,000 (note: the Fibonacci sequence grows very fast, fib(100) already has 21 digits. For the effects of this practical, it is not necessary to take overflow problems into account).
4. Analyse the obtained results by means of an empirical verification of the theoretical complexity. You also need to perform an empirical check using an underestimated and overestimated bound for each algorithm:
   - In the case of the first algorithm, $O(\phi^n)$ with $\phi = \frac{1+\sqrt{5}}{2}$, use function $f(n) = 1.1^n$ as the underestimated bound, and $f(n) = 2^n$ as the overestimated bound.
   - For the second $O(n)$ algorithm, we propose $f(n) = n^{0.8}$ and $f(n) = n\log(n)$ as the underestimated and overestimated bounds.
   - And in the third algorithm, $O(\log n)$, use $f(n) = \sqrt{(}\log(n))$ as the underestimated bound, and $f(n) = n^{0.5}$ as the overestimated bound.
5. Submit the `C` source files and the `.txt` file with the report by means of the task *Practical 1 Submission* at the Algorithms page at `https://campusvirtual.udc.gal`. The submission deadline is Friday, 23 September, at 23:59, and once the files have been uploaded they cannot be changed. **All members of each team have to submit the work.**

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

/* obtains the current time in microseconds */
double microseconds() {
  struct timeval t;
  if (gettimeofday(&t, NULL) < 0 )
    return 0.0;
  return (t.tv_usec + t.tv_sec * 1000000.0);
}
```

Figure 1: Obtention of system time

```
#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 = microseconds();
  fib3(n);
  t2 = microseconds();
  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);
}
```

Figure 2: function `fib3`