Análisis y Algoritmos

Pineda Chavez Luis Alberto
Universidad de artes digitales

Guadalajara, Jalisco

Email: funkalux@gmail.com

Profesor: Efraín Padilla

Abril 9, 2019

1) Homework 1:

Fibonacci Sequence

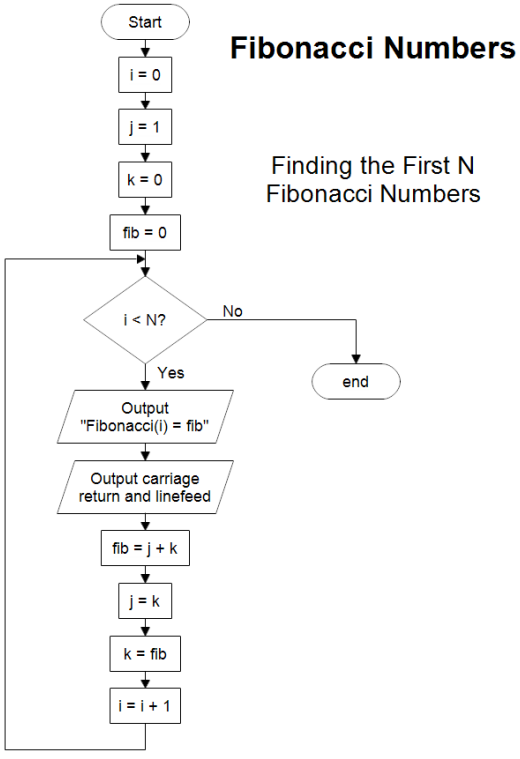
Described by Leonardo de Pisa, better known as Fibonacci, this sequence has numerous aplications in computer science, math and game theory.

It also appears in biological configurations, such as tree branches, in the arrangement of the leaves, on the stem, in the flowers of artichokes and sunflowers or in how DNA encodes the growth of complex organic forms. In the same way, it is found in the spiral structure of the shell of some mollusks.

The Objetive:

To calculate any number in the sequence, is necessary to code an algorithm that allows the user to insert a position in the sequence and return the corresponding value of that position.

To solve the problem, we can use this approach:



This is the code that corresponds to the recursive algorithm:

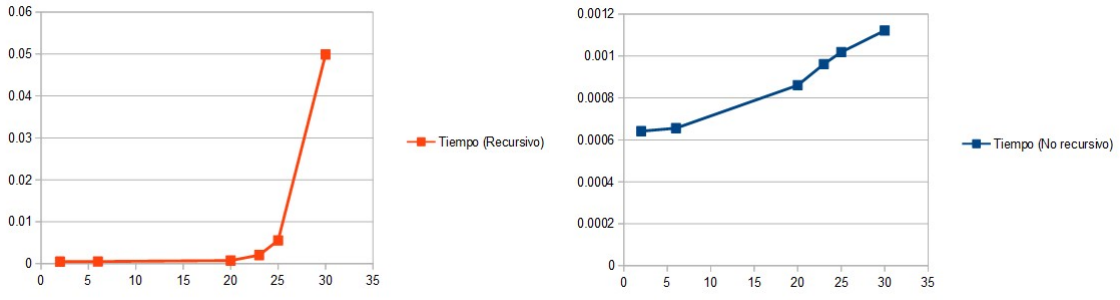
```
unsigned long Fibonnacci::Recursive(unsigned int index)
{
  if (index <= 1)
  {
    //the first two positions are the same value that the index they are in
    return index;
  }
  else
  {
    //Call the method again to calculate the value of n-1 and n-2
        return Recursive(index - 1) + Recursive(index - 2);
  }
}//this will repeat until n-1 and n-2 results in 0 and 1 respectibly
```

This is the non-recursive implementation:

```
unsigned long Fibonnacci::Calc(unsigned int index)
{
  m_value = 0; //Initialize the return value

  //This two variables allocates the values of n-1 and n-2 when index is 2
  unsigned int secondLast = 0;
  unsigned int last = 1;

  if (index <= 1)
  {
    //the first two positions are the same value that the index they are in
        m_value = index;
  }

  for (unsigned int i = 1; i < index; ++i)
  {//This loop iterates staring in index 2 until it reaches the desired index

        m_value = secondLast + last; //Adds n-1 and n-2 to get the current result

        //This prepares the new values of n-1 and n-2 for the next iterarion
        secondLast = last;
        last = m_value;
  }

  return m_value; //Returns the result
}
```

In terms of performance, the recursive implementation's time to calculate the result grows exponentially, meanwhile the non recursive implementation doesn't. Both has it's own advantages, each one is better in some situations, it depends where and how to use them.

This graphics represents how much time it takes to each implementation to calculate certain positions in the sequence:



REFERENCIAS

[1] Cormen, T., Leiserson, C., Rivest, R. and Stein, C. (2009). Introduction to algorithms. Cambridge (England): Mit Press.