Fibonacci

Santiago Toll Leyva
Universidad de Artes Digitales

Guadalajara, Jalisco

Email: idv16a.stoll@uartesdigitales.edu.mx

Profesor: Efraín Padilla

Mayo 08, 2019

Tarea 1: Fibonacci

Teoria

La serie Fibonacci describe una secuencia infinita de numeros ordenados. Esta serie describe un espiral perfecto que se encuentra en la naturaleza; los terminos de la sucesion tambien establecen la ”proporcion aurea”. Los numeros de la serie se obtienen al sumar los dos numeros previos en la serie.

Planteamiento del problema

La serie Fibonacci se define a partir de la siguienta funcion:

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f(n-1) + f(n-2) & n \geq 2 \end{cases}$$

A partir de la funcion anterior podemos determinar que podemos obtener el siguiente numero de la serie utilizando los valores de las dos posiciones anteriores.

Solucion del problema

Para resolver el problema se utilizaron dos funciones.

Funcion recursiva

Complejidad n.

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f(n-1) + f(n-2) & for\ \text{x} < n \end{cases}$$

Esto significa que la funcion se llamara a si misma mientras x sea menor a n.

Funcion no recursiva

Complejidad n^2.

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f(n-1) + f(n-2) & n \geq 2 \end{cases}$$

Codigo

Listing 1. Fibonacci C++

```
1  /*****************************************************************************/
2  /**
3  * @file         Fibonacci.cpp
4  * @authors      Santiago Toll (santiago.toll97@gmail.com)
5  * @date         May 2019
6  * @brief        Fibonacci in c++
7  *
8  * @bug      No known bugs.
9  */
10 /*****************************************************************************/
11
12 /*****************************************************************************/
13 /*
14  * Includes
15  */
16  /*****************************************************************************/
17 #include "pch.h"
18 #include <iostream>
19 #include <chrono>
20 #include <string>
21
22 using std::cin;
23 using std::cout;
24 using std::string;
25 using std::chrono::high_resolution_clock;
26 using::std::chrono::duration;
27
28
29 /*****************************************************************************/
30 /* Fibonacci using recursivity.                                              */
31 /*****************************************************************************/
32 int
33 RecursiveFibonacci(int numberber)
34 {
35   if (numberber <= 1)
36   {
37     return numberber;
38   }
39   else
40   {
41     return RecursiveFibonacci(numberber - 1) + RecursiveFibonacci(numberber - 2);
42   }
43 }
44
45 /*****************************************************************************/
46 /* Fibonacci(non recursive)                                                  */
47 /*****************************************************************************/
48 int
49 Fibonacci(int n)
50 {
51   //initialize variable
52   int n1 = 0, n2 = 1;
53
54   if (n <= 1)
55   {
```

```
56      return n;
57    }
58    else
59    {
60      int tmp = 0;
61      for (int i = 0; i < n - 1; ++i)
62      {
63        tmp = n1 + n2;
64        n1 = n2;
65        n2 = tmp;
66      }
67      return tmp;
68    }
69  }
70
71  /*****************************************************************/
72  /* Function used to print results                                */
73  /*****************************************************************/
74  void
75  PrintResult(int result, double duration)
76  {
77    std::string  text = "Result: " + std::to_string(result) + "\n" +
78      "Elapsed time: " + std::to_string(duration) + "\n";
79    std::cout << text << std::endl;
80  }
81
82  int main()
83  {
84    int number = 0;
85
86    std::cout << "Enter the position in sequence: ";
87    std::cin >> number;
88
89    //get the start time
90    auto startTime = high_resolution_clock::now();
91    //Call Non recursive fibonacci function.
92    int result = Fibonacci(number);
93    //Get end time.
94    auto endTime = high_resolution_clock::now();
95    //Calculate elapsed time.
96    duration<double> duration = endTime - startTime;
97
98    PrintResult(result, duration.count());
99
100   startTime = high_resolution_clock::now();
101   //Call Non recursive fibonacci function.
102   result = RecursiveFibonacci(number);
103   //Get end time.
104   endTime = high_resolution_clock::now();
105   //Calculate elapsed time.
106   duration = endTime - startTime;
107   PrintResult(result, duration.count());
108
109   return 0;
110 }
```