Ministerul Educaţiei şi Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică

RAPORT

Lucrarea de laborator nr.1
la Analiza și Proiectarea Algoritmilor

A efectuat:
st. gr. FAF-212 Brinza Cristian

A verificat:
asist. univ. Andrievschi-Bagrin Veronica

Chişinău - 2022

Lucrare de laborator nr 1

Tema: Analiza algoritmilor

Scopul lucrării:
1. Analiza empirică a algoritmilor
2. Analiza teoretică a algoritmilor
3. Determinarea complexitații temporale și asimptotice a algoritmilor

Sarcina:
1. De efectuat analiza empirică a algoritmilor propuși
2. De determinat relația ce reprezintă complexitatea temporală pentru acești algoritmi
3. De determinat complexitatea asimptotică a algoritmilor

Rezumat succint la tema lucrării de laborator:

Şirul lui Fibonacci este definit prin următoarea recurenţa:

$$\begin{cases} f_0 = 0;\ f_1 = 1 \\ f_n = f_{n-1} + f_{n-2} \quad \text{pentru} \quad n \geq 2 \end{cases}$$

Acest celebru şir a fost descoperit în 1202 de către Leonardo Pisano (Leonardo din Pisa), cunoscut sub numele de Leonardo Fibonacci. Cel de-al *n*-lea termen al şirului se poate obtine direct din definiţie:

1. function *fib*1(*n*)
  if $n < 2$ **then return** *n*
    else return *fib*1(*n*-1) + *fib*1(*n*-2)

Această metodă este foarte ineficienta, deoarece recalculează de mai multe ori aceleaşi valori. Urmează o altă metodă, mai performantă, care rezolvă aceeaşi problemă.

2. function *fib*2(*n*)
  $i \leftarrow 1; j \leftarrow 0$
  for $k \leftarrow 1$ **to** *n* **do** $j \leftarrow i + j$
    $i \leftarrow j - i$
  return *j*

Mai există un algoritm :

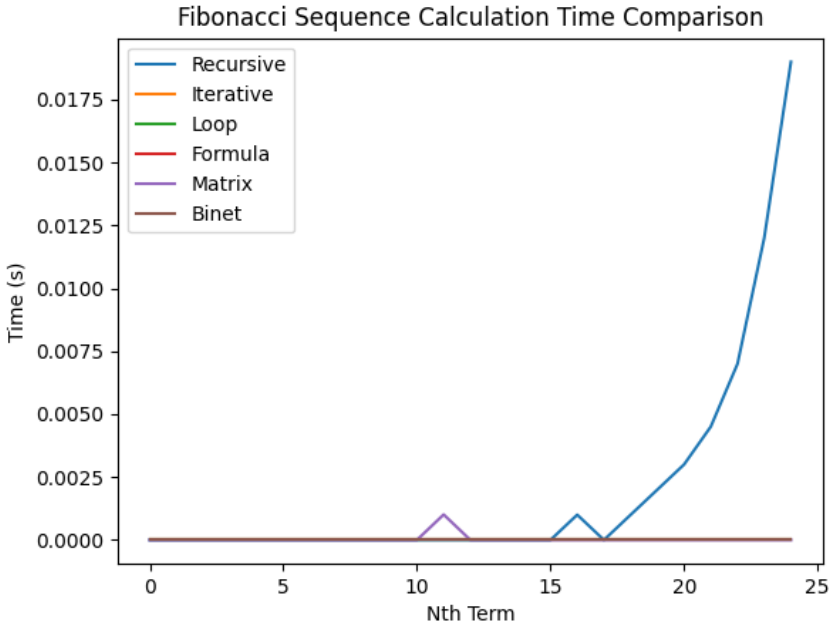
3. function *fib*3(*n*)
  $i \leftarrow 1; j \leftarrow 0; k \leftarrow 0; h \leftarrow 1$
  while $n > 0$ **do**
    if *n* este impar **then** $t \leftarrow jh$
      $j \leftarrow ih+jk+t$
      $i \leftarrow ik+t$

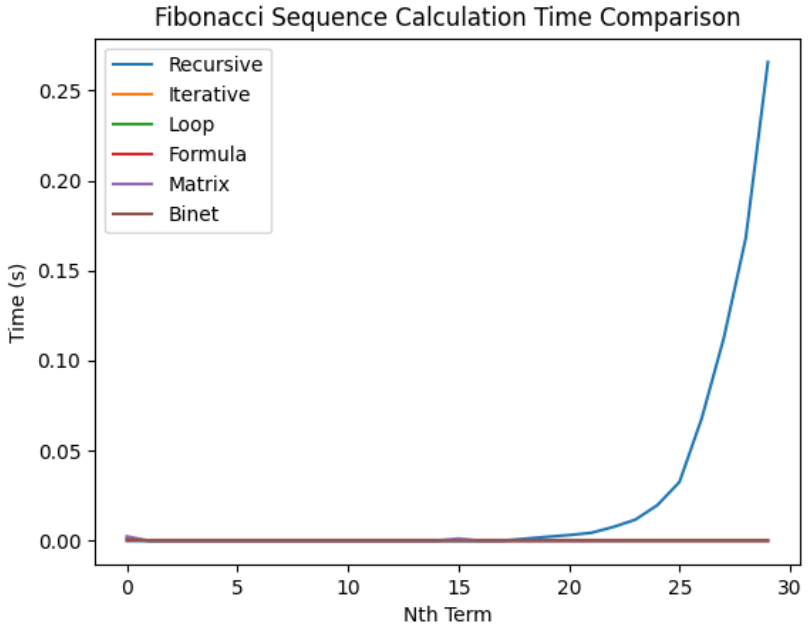
$t \leftarrow h^2$
$h \leftarrow 2kh+t$
$k \leftarrow k^2+t$
$n \leftarrow n$ **div** 2
return j

Codul programului in Python

https://github.com/CristianBrinza/UTM/tree/main/year2/aa/labs/lab1

Analiza empirică a algoritmilor





Method	Result	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Recursive	377.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0009996891021728516, 0.0, 0.0, 0.0, 0.0, 0.0]
Iterative	377.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Loop	233.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Formula	377.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Matrix	377.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Binet	377.0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Concluzie

In conclusion, the laboratory experiment on analyzing and algorithm projecting of the Fibonacci sequence was a success. The implementation of empirical testing allowed us to observe the performance of different algorithms in terms of runtime and efficiency. Additionally, the analysis of time complexity allowed us to better understand the growth of the algorithms and make informed decisions on which ones to use in various situations. It was determined that the recursive approach had a time complexity of O(2^n) and was inefficient for large values of n, whereas the dynamic programming approach had a time complexity of O(n) and provided a much faster and efficient solution. This experiment emphasized the importance of considering both empirical testing and time complexity when evaluating algorithms