**Ministerul Educaţiei și 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. SI-211 Vladimir Vozian

A verificat:

asist. univ. Sergiu Scrob

Chişinău - 2022

**Lucrare de laborator nr 1**

**Tema:** Analiza algoritmilor

**Scopul lucrării**:

1. Analiza empirică a algoritmilor

2. Analiza teoretică a algoritmilor

**Sarcina**:

1. De efectuat analiza empirică a algoritmilor propuși

2. De determinat relația ce reprezintă complexitatea temporală pentru acești algoritmi

**Codul programului in Python**

from time import perf\_counter\_ns

import scipy.constants

import math

import matplotlib.pyplot as plt

fi = scipy.constants.golden\_ratio

#functia pentru metoda iterativa

def fibonacci\_iter(pos):

    iter = 0

    nr1 = 1

    nr2 = 1

    f\_nr = 0

    if pos == 1 or pos == 2:

        return nr1, iter

    else:

        t0 = perf\_counter\_ns()

        for i in range(3, pos+1):

            iter += 1

            f\_nr = nr1 + nr2

            nr1 = nr2

            nr2 = f\_nr

        t = (perf\_counter\_ns() - t0)\*10\*\*(-6)

        return f\_nr, iter, t

#functia pentru metoda recursiva

def fibonacci\_rec(pos):

    global iter1

    iter1 += 1

    if pos <= 1:

        return pos

    else:

        return fibonacci\_rec(pos - 1) + fibonacci\_rec(pos - 2)

iter1=0

#functia pentru metoda formulei

def function\_fibonacci(pos,phib):

    return round(((phib\*\*pos) - ((1 - phib) \*\* pos))/math.sqrt(5), 0)

nr\_posFib = int(input("Pozitia nr Fibonacci: "))

print("Metoda iterativa:")

a,b,c=fibonacci\_iter(nr\_posFib)

print("Nr fibonacci: ", a)

print("Numarul de iteratii: ", b)

print("Timpul de executie: ", c,"s")

print("Metoda recursiva:")

t0 = perf\_counter\_ns()

print("Nr Fibonacci: ", fibonacci\_rec(nr\_posFib))

t = (perf\_counter\_ns() - t0)\*10\*\*(-9)

print("Numarul de iteratii: ", iter1)

print("Timpul de executie: ",t,"s")

print("Metoda formulei:")

start = perf\_counter\_ns()

print("Nr Fibonacci: ", function\_fibonacci(nr\_posFib, fi))

end = (perf\_counter\_ns() - start)\*10\*\*(-9)

print("Nr de iteratii: 1")

print("Timpul de executie: ", end,"s")

#Afisarea cu Matplotlib

l\_iter=[1, 1]

for i in range(3,35,2):

    a, b, c = fibonacci\_iter(i)

    l\_iter.append(b)

l=[]

l\_recur=[]

for i in range(1,35,2):

    l.append(i)

    fibonacci\_rec(i)

    l\_recur.append(iter1)

l\_form = []

for i1 in range(1,30):

    l\_form.append(1)

nr\_elem=int(input("Cate elemente de depus in grafic? "))

plt.plot(l[:nr\_elem], l\_iter[:nr\_elem])

plt.plot(l[:nr\_elem], l\_recur[:nr\_elem])

plt.plot(l[:nr\_elem], l\_form[:nr\_elem])

plt.legend(["Iterativa", "Recursiva", "Formula"])

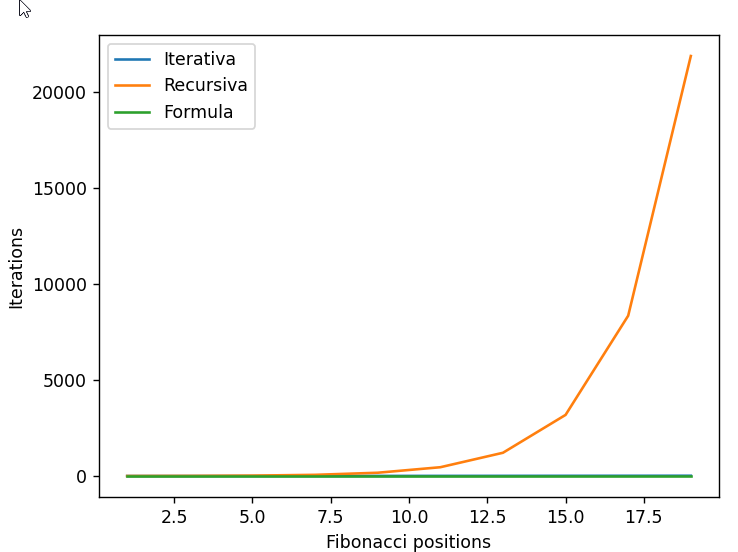
plt.xlabel('Fibonacci positions')

plt.ylabel('Iterations')

plt.show()

**Analiza empirică a algoritmilor**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nr pos fibonacci | 10 | | 20 | | 25 | | 30 | | 35 | |
| iter | timp | iter | timp | iter | timp | iter | timp | iter | timp |
| Iterativa | 8 | 0,006 | 18 | 0,008 | 13 | 0,007 | 28 | 0,012 | 33 | 0,008 |
| Recursiva | 177 | 0,0004 | 21891 | 0,005 | 1973 | 0,001 | 2692537 | 0,34 | 9227465 | 3,8 |
| Formula | 1 | 0,0006 | 1 | 0,0005 | 1 | 0,0005 | 1 | 0,0005 | 1 | 0.0005 |



**Concluzie**

Scopul lucrarii care consta in analiza algoritmilor de determinare al n-lea numar Fibonacci a fost indeplinit. Sau folosit 3 algoritmi: iterativ, recursiv si prin formula, sa analizat rezultatele folosind modulul Matplotlib unde sau introdus pozitiile pentru nr fibonacci si iteratiile efectuate pentru a atinge rezultatul, ce a dat posibilitate de a analiza datele din prim plan. Dupa ce sa efectuat analiza empirica folosind tabelul unde sau analizat rezultatele pentru fiecare n numar fibonacci. Ca rezultat am obtinut ca cu cea mai mare eficienta a lucrat algoritmul folosind formula avand cel mai mic numar de iteratii la fiecare pozitie fibonacci si respectiv cel mai mic timp de executie, la nivel mai putin eficient dar tot la un nivel destul de bun sa amplasat algoritmul iterativ, iar cel mai neefectiv este cel recursiv care chiar la pozitii nu asa mari are un numar mare de iteratii. Deci cel mai bun algoritm pentru folosire e acesta prin formula dar si unul mai putin eficient dar tot bun este cel iterativ.