**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. FAF-212 Smocvin Denis

A verificat:

asist. univ. Fistic Cristofor

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:



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 (recursiv):

**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 iterativ.

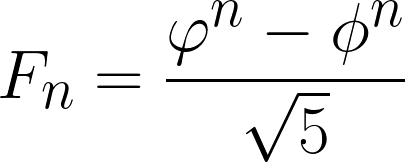
**2. function** *fib*2(*n*)  
     *i*  1; *j*  0  
     **for** *k*  1 **to** *n* **do** *j*  *i* + *j*                                 *i*  *j* - *i* **return** *j*

Daca aplicam tehnici ale programarii dinamice, putem optimiza complexitatea temporala:

1. Function fib3(n)  
    fib[0]0; fib[1]1  
    **for** i 2 **to** *n+1* **do** fib[i]fib[i-1]+fib[i-2]

**return** fib[n]

1. Acest algoritm este mai complicat. Se folosesc eigenvectori si eigen valori. Calculezi acestea pentru matricea [[1, 1], [1, 0]]. Apoi iai eigenvectorii si faci inmultirea matricieala cu diagonala matricei eigenvectorilor la puterea n. Cu rezultatul mai facem o inmultire matriceala cu transpusa matricei eigenvectorilor.
2. Algoritm cu eigenvectori optimizat implementeaza unele operatii intr-un mod mai eficient.
3. Algoritmul ratiei de aur foleste formula lui Binet:

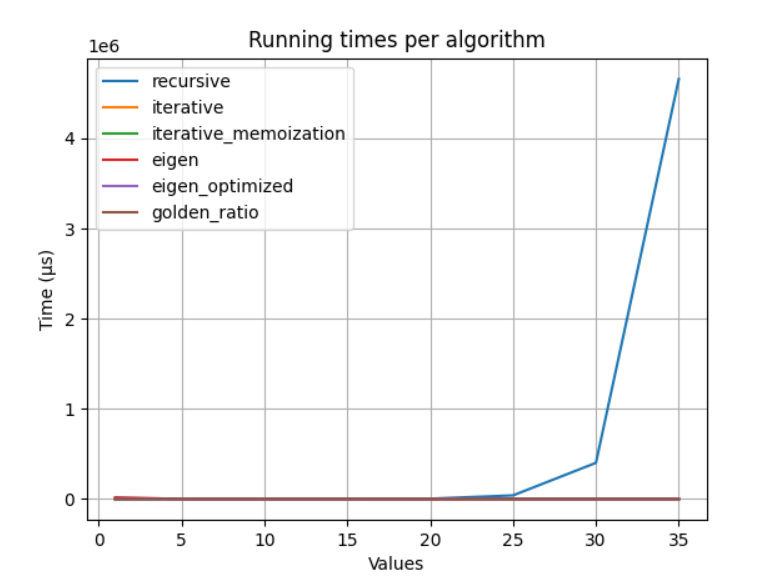


**Codul**

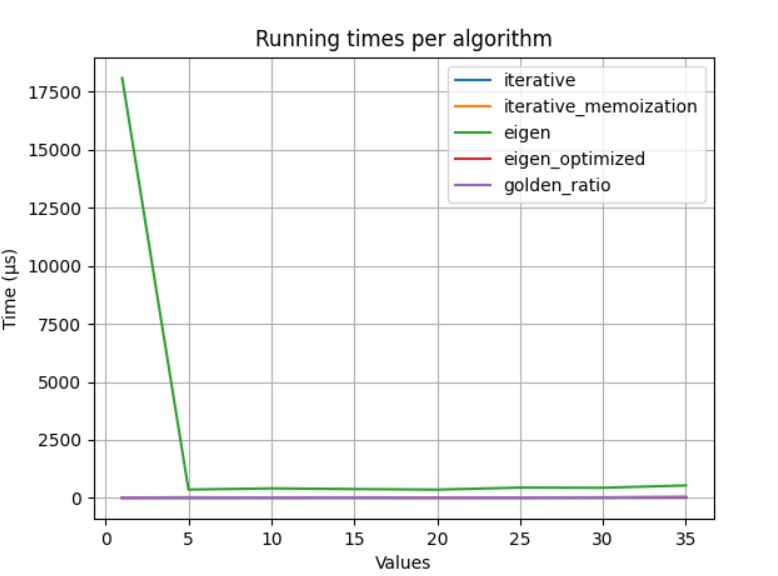
In scopul analizei au fost create functii care printeaza timpul de executiei pentru fiecare algoritm si o functie de printare a grafurilor care prezinta relatia dintre timpul de executie si al catale nr din sirul Fibbonacci este cerut.

Codul poate fi gasit in repozitoriul de pe urmatorul link: <https://github.com/JustLearningThings/APA_Labs/blob/main/lab%201/fibonacci.py>

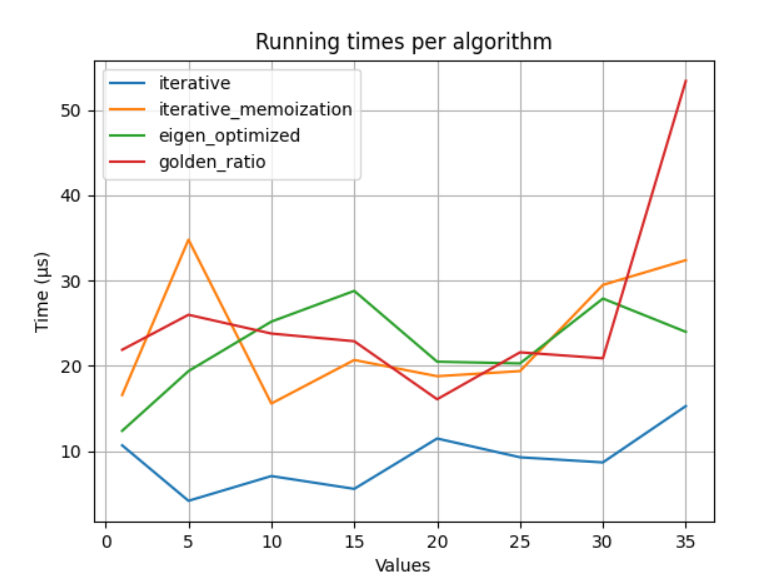
**Grafurile**



Vedem ca algoritmul recursiv ocupa un timp excesiv de mult comparativ cu alte algoritmuri, incepand cu valoare 25. Il stergem pentru a compara restul rezultatelor.

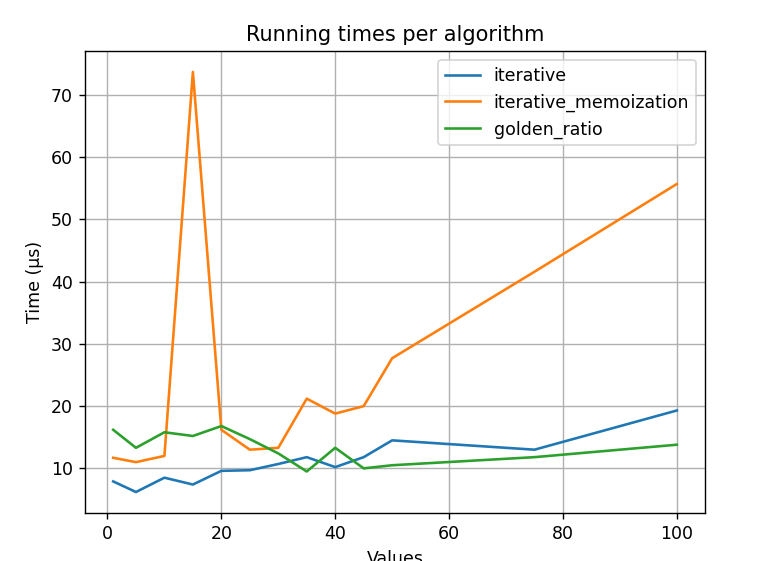


Acelasi lucru il observam si la algoritmul cu eigenvectori, insa la valori mici (desi odata cu cresterea valorilor acesta oricum se ispraveste mai rau). Acest lucru are loc deoarece sunt efectuate calcule “scumpe” independent de valoare lui n. Il excludem si pe acesta.



Observam ca notiunea de algoritmul “optimal” fluctueaza pentru acest interval de valori. Desi aici castigatorul este algoritmul iterativ, acest lucru nu demonstreaza ca va fi la fel de bun pentru valori mult mai mare ale lui n.

Daca facem acelasi lucru insa pentru intervalul 1-100, obtinem acest grafic (care include 3 cele mai bune algoritme:



**Concluzie**

In scopul analizei algoritmilor de obtinerea al n-ulea numar din sirul Fibbonacii au fost analizati 6 algoritmi: recursiv, iterativ, iterativ cu memoizare, cu eigenvectori, cu eigenvectori optimizat, cu ratia de aur. In intervalul de valori 1-35 cel mai efectiv a fost algoritmul iterativ, iar cel mai rau cel recursiv. Pentru valorile 0-100 ale lui n avem algoritmul iterativ cel ca mai bun pana la n aprox. 32, apoi cel cu ratia de aur se reprezinta cel mai bine.

Analiza a inclus implementarea algoritmilor, masurarea timpului de executie pentru fiecare cu mai multe valori, construirea grafurilor cu timpul de executie si analiza acestuia. Aceasta este o metoda empirica de analiza si este focusata pe un interval mic de valori.