**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-221 Ceclea Victor

A verificat:

asist. univ. Andrievschi-Bagrin Veronica

Chişinău - 2023

**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:

**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*  1; *j*  0  
     **for** *k*  1 **to** *n* **do** *j*  *i* + *j*                                 *i*  *j* - *i* **return** *j*

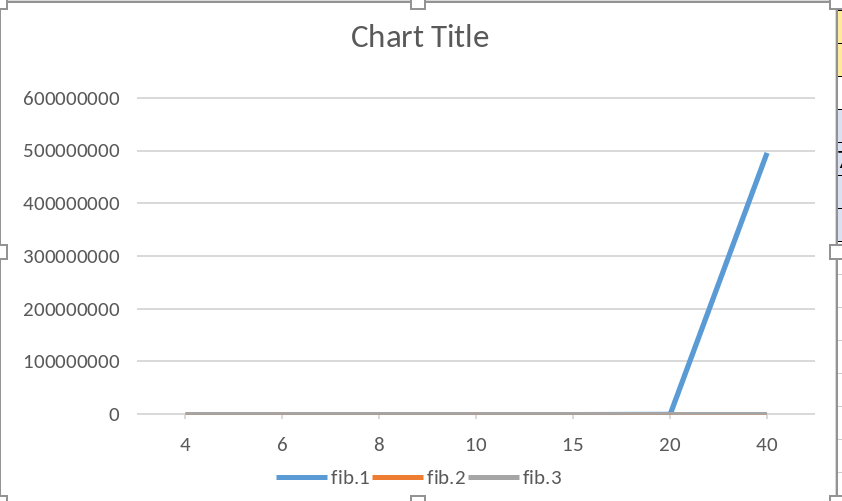
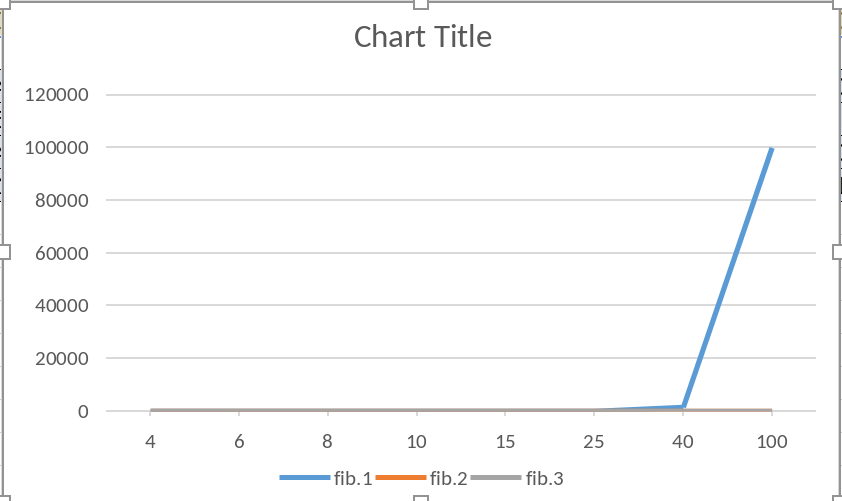
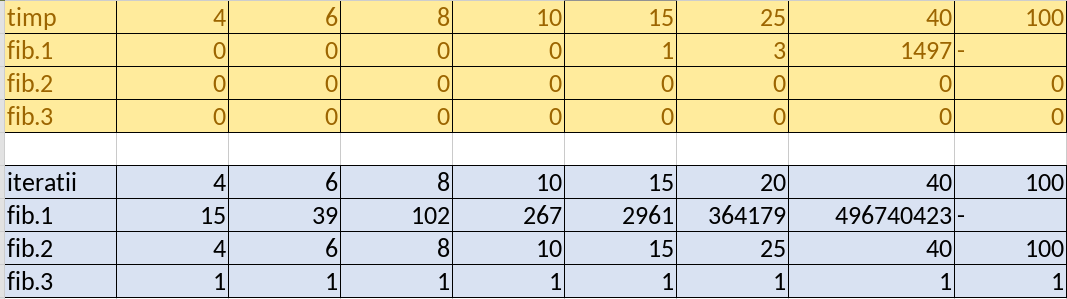
Mai există un algoritm :

**3. function** *fib*3(*n*)  
     *i*  1; *j*  0; *k*  0; *h*  1  
     **while** *n* > 0 **do**          **if** *n* este impar **then** *t*  *jh  
                                           j*  *ih*+*jk*+*t  
                                           i*  *ik*+*t*          *t*   *h*2  
          *h*  2*kh*+*t*          *k*  *k*2+*t*          *n*  *n* **div** 2  
     **return** *j*

**Codul programului in JS**

const recursie = (n) =>{  
 if(n<2) {  
 it1 = it1 + 3;  
 return n;  
 }  
 return recursie(n-1) + recursie(n-2)  
}  
  
  
  
  
const fib2 = (n) =>{  
 let i = 1  
 let j = 0  
 for(let k = 1; k<=n; k++)  
 {  
 j = i+j;  
 i = j-i;  
 }  
 it2 = it2 + n  
  
 return j  
}  
  
const fib3 = (n) =>{  
 let phi = (1 + Math.sqrt(5)) / 2;  
 let result = Math.round(Math.pow(phi, n) / Math.sqrt(5));  
 it3 = it3 + 1  
 return result;  
}  
  
const n = 41  
let it1 = 0  
console.log('Recursie')  
const start1 = new Date()  
console.log(`Numarul Fiboncai${recursie(n)} iteratii ${it1}`);  
const end1 = new Date()  
console.log(end1-start1)  
  
let it2 = 0  
console.log('Fib 2')  
const start2 = new Date()  
console.log(`Numarul Fiboncai${fib2(n)} iteratii ${it2}`);  
const end2 = new Date()  
console.log(end2-start2)  
  
let it3 = 0  
console.log('fib3')  
const start3 = new Date()  
console.log(`Numarul Fiboncai${fib3(n)} iteratii ${it3}`);  
const end3 = new Date()  
console.log(end3-start3)

**Analiza empirică a algoritmilor**



**Concluzie**

Scopul lucrării propuse, ce contă în analiza empirică și teoretică a algoritmilor de determinare al n-lea număr Fibonacci, s-a îndeplinit cu succes. S-au folosit trei algoritmi diferiți: recursiv, iterativ și prin formulă, pentru a determina complexitatea temporală și asimptotă a acestora și pentru a evidenția cel mai eficient algoritm, ce ne va afișa rezultatul dorit folosind cât mai puține iterații. Comparația algoritmilor s-a efectuat cu ajutorul tabelului în care colectam datele empirice despre iterațiile primite pe parcursul lucrării și a graficelor construite in Excel, ce ne afișează diferanța vizibilă dintre complexitățile algoritmilor. De aici observăm că metoda recursivă este cea mai ineficientă, deoarece recalculează de mai multe ori aceleași valori, în timp ce metoda prin formulă ne-a oferit rezultate mult mai favorabile. Algoritmul iterativ este la fel unul suficient de bun, avâd o complexitate liniara. În urmare, pentru determinarea al n-lea număr Fibonacci se va folosi metoda iterativă și cea prin formulă.