Ministerul Educaţiei, Culturii și Cercetării

al Republicii Moldova   
Universitatea Tehnică a Moldovei

FCIM  
  
  
  
  
  
  
  
RAPORT

# LUCRARE DE LABORATOR NR. 1

**La disciplina Analiza şi Proiectarea Algoritmilor**

**Tema : Analiza algoritmilor**

A efectuat: st. gr. TI-211 Popa Cătălin   
  
A verificat: Bagrin Veronica

UTM, Chișinău 2022

**Tema:**

Analiza algoritmilor.

**Scopul lucrării:**

Analiza empirica a algoritmilor.

Aanaliza teoretica a algoritmilor.

Determinarea complexitatii temporale si asimptotice a algoritmilor.

**Sarcinile de baza:**

1.Efectuati analiza empirica a algoritmilor propusi.

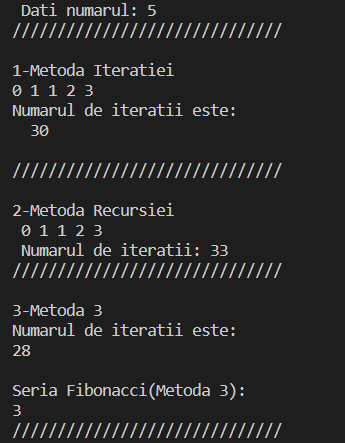
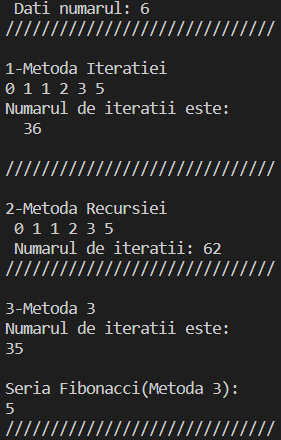
2.Determinati relatia ce reprezinta complexitatea temporala pentru acesti algoritmi.

3.Determinati complexitatea asimptotica a algoritmilor.

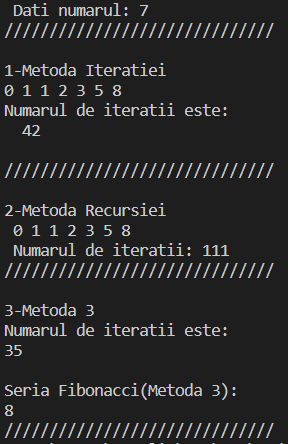
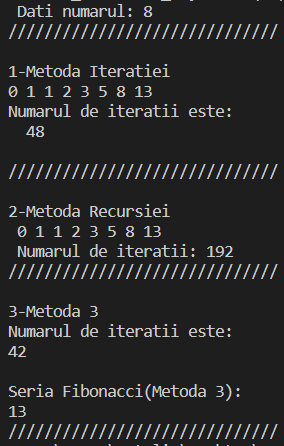
4.Faceti o concluzie asupra lucrarii effectuate.

În următoarele poze voi prezenta datele privind numarul de iteratii realizate de fiecare algoritm.

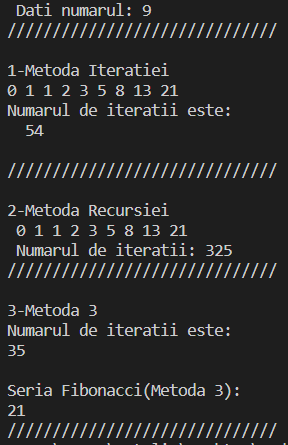
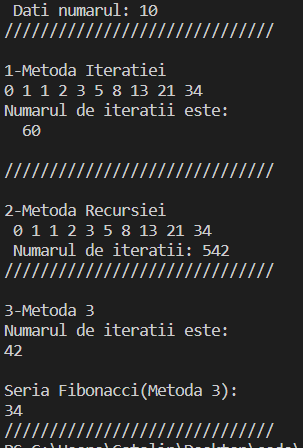
1. Datele pentru pozitia 5: 2. Datele pentru pozitia 6:

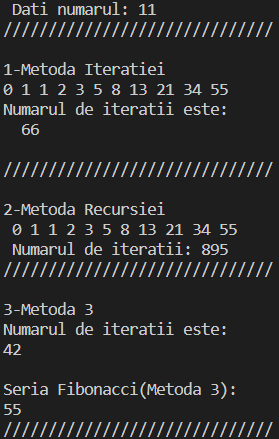
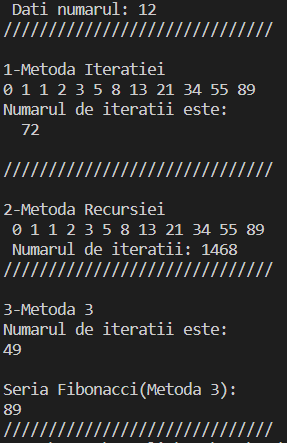
3. Datele pentru pozitia 7: 4. Datele pentru pozitia 8:

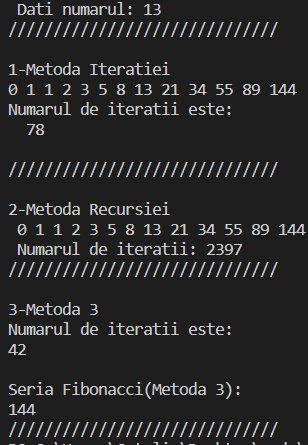
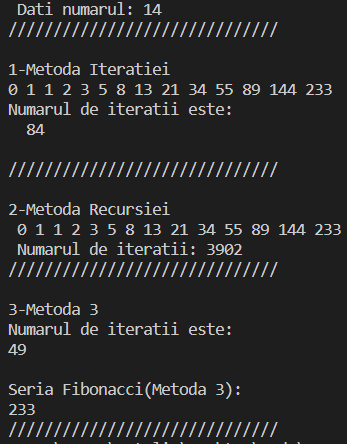
5. Datele pentru pozitia 9: 6. Datele pentru pozitia 10:

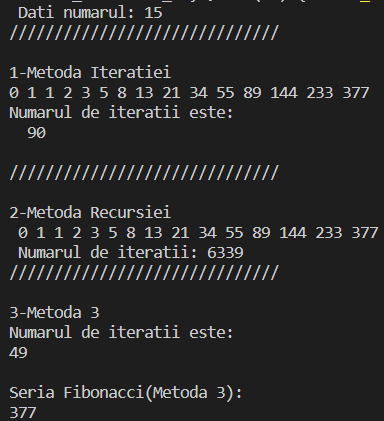
7. Datele pentru pozitia 11: 8. Datele pentru pozitia 12:

9. Datele pentru pozitia 13: 10. Datele pentru pozitia 14:

11. Datele pentru pozitia 15:



De asemenea datele obtinute le-am plasat intr-o diagrama si un table. Am realizate o diagrama suplimentara pentru a observa mai concret care este mai eficient algoritm.

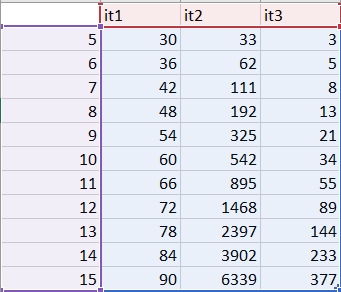


Diagrama principala:

Diagrama suplimentara:

In diagrame observam ca ,,it 1’’ (metoda iteratiei), este liniara, cee ace inseamna ca este cel mai efficient algoritm. Spre deosebire de algortimul respectiv, metoda recursiei se observa ca creste in sus eficient, ceea ce insemana ca cu fiecare termen complexitatea creste si nu este un algoritm favorabil. De asemenea se refera si la algoritmul trei, care pana la terminul 11 are o complexitate mai mica decat cea a iteratiei, iar dupa termenul 12 va creste si la el complexitatea.

Codul sursa:

#include<iostream>

#include<string.h>

#include<stdlib.h>

#include<conio.h>

using namespace std;

int count = 0;

int fib1(int num)

    {

        if((num == 0) || (num == 1))

        {

            count++;

            return(num);

        }

        else

        {

            count = count + 3;

           return(fib1(num-1) + fib1(num-2));

        }

    }

int fib2(int num)

{

    int count2 = 0;

     int n1 = 0, n2 = 1, n;

    for(int i = 0; i < num; i++)

    {

        count2 =count2 + 3;

        cout << n1 << " ";

        n = n1 + n2;

        count2++;

        n1 = n2;

        count2++;

        n2 = n;

        count2++;

    }

    cout << "\nNumarul de iteratii este:\n  " << count2 << endl;

}

int fib3(int num)

{

    num = num - 1;

    int i = 1,j = 0,k = 0,h = 1,t = 1;

    int count3 = 0;

    while(num > 0)

    {

        if(num%2 != 0)

        {

            t = h\*j;

            j = i\*h+j\*k+t;

            i = i\*k+t;

            count3 = count3 + 7;

        }

        {

            t = h\*h;

            h = 2\*k\*h+t;

            k = k\*k+t;

            num = (num / 2);

            count3 = count3 + 7;

        }

    }

    cout << "\nNumarul de iteratii este:  \n" << count3 << endl;

    return j;

}

int main()

{

    int num,num1,i=0;

    // int count=0;

    cout << " Dati numarul: ";

    cin >> num;

    cout <<"//////////////////////////////\n";

///////////////////////////////////

      cout <<"\n1-Metoda Iteratiei\n";

        fib2(num);

      cout <<"\n//////////////////////////////\n";

///////////////////////////////////

          cout <<"\n2-Metoda Recursiei\n";

        while(i < num)

        {

            cout << " " << fib1(i);

            i++;

        }

        cout << "\n Numarul de iteratii: ";

        cout << count;

        cout <<"\n//////////////////////////////\n";

/////////////////////////////////////

    cout <<"\n3-Metoda 3";

    cout << "\nSeria Fibonacci(Metoda 3):     \n" << fib3(num) << endl;

cout <<"//////////////////////////////\n";

/////////////////////////////////////////////

return 0;

}

**Concluzie:**

In urma realizarii lucrarii de laborator nr.1, am efectuat analiza empirica a agoritmilor de determinare a termenului n prezent in sirul Fibonacci.

In concluzie , dupa complexitate se observa a fi cel mai eficient metoda Iteratiei. Aceasta metoda are complexitatea lineara, spre deosebire de celelalte 2 metode, care au o crestere in complexitate si nu sunt eficiente. Se obseva ca pana la 11 element, metoda 3 este eficienta dupa complexitate, iar dupa elementul 11 complexitatea sa creste si nu este favorabil ca metoda iteratiei.

**Intrebari de control**:

**1. Enumeraţi factorii ce influenţează timpul de execuţie al algoritmului.**

Factorii ce influenţează timpul de execuţie al algoritmului sunt datele de intrare și numărul acestora, calitatea codului generat de compilator, natura și viteza de execuție a instrucțiunilor programului, complexitatea algoritmului care stă la baza programului.

**2. Care sunt regulile generale pentru evaluarea timpului de execuţie?**

Dacă în program, unde sunt mai multe algoritmi aplicați, este un algoritm cu complexitatea cunoscută a tunci se ia pentru întregul program complexitatea cea mai mare.

**3. Care sunt etapele analizei empirice şi în care cazuri se face analiza empirică a algoritmilor?**

Etapele analizei empirice:

1. Se stabileşte scopul analizei.

2. Se alege metrica de eficienţă ce va fi utilizată (număr de execuţii ale unei/unor operaţii sau timp de execuţie a întregului algoritm sau a unei porţiuni din algoritm.

3. Se stabilesc proprietăţile datelor de intrare în raport cu care se face analiza (dimensiunea datelor sau proprietăţi specifice).

4. Se implementează algoritmul într-un limbaj de programare.

5. Se generează mai multe seturi de date de intrare.

6. Se execută programul pentru fiecare set de date de intrare.

7. Se analizează datele obţinute.

**4. Proprietatile ale algoritmilor:**

1. Generalitate

2.Rigurozitate

3.Eficienta

4.Finitudinea.