Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra ATI

**RAPORT**

Lucrarea de laborator nr.1

*Analiza si proiectarea algoritmilor*

A efectuat:

st. gr. TI-154 V.Suruceanu

A verificat:

lect., univ. V.Bagrin

Chisinau 2016

**LUCRAREA DE LABORATOR NR.1**

**Tema:** Analiza algoritmilor

**Scopul lucrării:**

1. Analiza empirică a algoritmilor.
2. Analiza teoretică a algoritmilor.
3. Determinarea complexităţii temporale şi asimptotice a algoritmilor

### Sarcina pentru lucrarea nr.1: Numerele lui Fibonacci

### Şirul lui Fibonacci este definit prin următoarea recurenţa:

http://vega.unitbv.ro/~andonie/Cartea%20de%20algoritmi/images/cap1/cap1_3.gif

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:

**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. Iată acum o altă metodă, mai performantă, care rezolvă aceeaşi problemă.

**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 :

**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*

**Sarcina de baza:**

1. Efectuaţi analiza empirică a algoritmilor propuşi.
2. Determinaţi relaţia ce determină complexitatea temporală pentru aceşti algoritmi.
3. Determinaţi complexitatea asimptotică a algoritmilor.
4. Faceţi o concluzie asupra lucrării efectuate.

**Descrierea algoritmilor**

Algoritmul este noțiunea fundamentală a informaticii. Totul este construit în jurul algoritmilor (și a [structurilor de date](https://ro.wikipedia.org/wiki/Structuri_de_date), cum ar fi [listele](https://ro.wikipedia.org/w/index.php?title=Listă&action=edit&redlink=1) sau [grafurile](https://ro.wikipedia.org/wiki/Graf)). Este și un concept fundamental al matematicii moderne alături de cel de [sistem axiomatic](https://ro.wikipedia.org/wiki/Sistem_axiomatic)

**Timpul de execuţie al algoritmilor**

De multe ori, pentru rezolvarea unei probleme, trebuie ales un algoritm dintre mai mulţi posibili, două criterii principale de alegere fiind contradictorii: algoritmul să fie simplu de înţeles, de codificat şi de depanat; algoritmul să folosească eficient resursele calculatorului, să aibă un timp de execuţie redus.  
Dacă programul care se scrie trebuie rulat de un număr mic de ori, prima cerinţă este mai importantă; în această situaţie, timpul de punere la punct a programului e mai important decât timpul lui de rulare, deci trebuie aleasă varianta cea mai simplă a programului.  
Daca programul urmează a fi rulat de un număr mare de ori, având şi un număr mare de date de prelucrat, trebuie ales algoritmul care duce la o execuţie mai rapidă. Chiar în această situaţie, ar trebui implementat mai înainte algoritmul mai simplu şi calculată reducerea de timp de execuţie pe care ar aduce-o implementarea algoritmului complex.

**Analiza empirică a complexităţii algoritmilor**

O alternativă la analiza matematică a complexităţii o reprezintă analiza empirică.  
Aceasta poate fi utilă pentru:  
- a obţine informaţii preliminare privind clasa de complexitate a unui algoritm;  
- pentru a compara eficienţa a doi (sau mai mulţi) algoritmi destinaţi rezolvării aceleiaşi probleme; - pentru a compara eficienţa mai multor implementări ale aceluiaşi algoritm;   
- pentru a obţine informaţii privind eficienţa implementării unui algoritm pe un anumit calculator

**Codul programului in Java**

import java.util.\*;

class sarcina1 {

static List<Integer> f = new ArrayList<Integer>();

static long iteratii=0;

static long startTime,stopTime,duration;

public static void main(String[] args) {

int n,

opt;

char exit = 'n';

Scanner input = new Scanner(System.in);

while ((exit == 'n') || (exit == 'N')) {

System.out.printf("Introduceti numarul de inceput: ");

n = input.nextInt();

System.out.printf("1.Algoritmul recursiva\n2.Algoritmul iterativa\n3.Algoritmul divizarii\n~:");

opt = input.nextInt();

switch ((int)opt) {

case 1: {

startTime = System.nanoTime();

int fibb = fib1(n);

stopTime = System.nanoTime();

duration = stopTime - startTime;

System.out.print("Numarul fibonaci: "+fibb+"\tIteratii: "+iteratii+"\tTimp: "+duration+"nS");

break;

}

case 2: {

startTime = System.nanoTime();

int fib2 = fib2(n);

stopTime = System.nanoTime();

duration = stopTime - startTime;

System.out.println("Numarul fibonaci: "+fib2+"\tIteratii: "+iteratii+"\tTimp: "+duration+"nS");

break;

}

case 3: {

startTime = System.nanoTime();

int fib3 = fib3(n);

stopTime = System.nanoTime();

duration = stopTime - startTime;

System.out.println("Numarul fibonaci: "+fib3+"\tIteratii: "+iteratii+"\tTimp: "+duration+"nS");

break;

}

default: {

System.out.print("Optiune gresita!!!");

break;

}

}

System.out.print("\nEsiti? [y/n]: ");

exit = input.next().charAt(0);

}

}

public static int fib1(int n) {

if (n<2) {

iteratii++;

return n;

}

else {

iteratii++;

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

}

}

public static int fib2(int n) {

int i=1,j=0,k;

iteratii = 0;

for (k=0;k<n;k++) {

j=i+j;

i=j-i;

iteratii++;

}

return j;

}

public static int fib3(int n) {

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

iteratii = 0;

while (n>0) {

if (n % 2 != 0) {

t=j\*h;

j=i\*h+j\*k+t;

i=i\*k+t;

}

t = (int)Math.pow(h,2);

h = 2\*k\*h+t;

k = (int)Math.pow(k,2)+t;

n = n/2;

iteratii++;

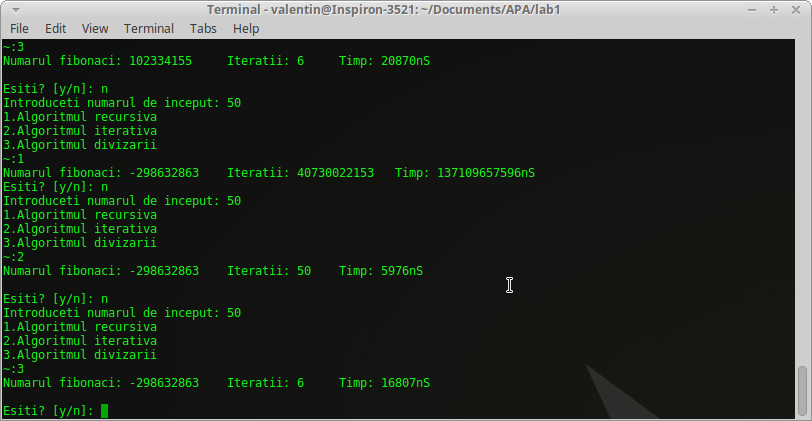
}

return j;

}

}

**Screen**



**Reprezentarea rezultatelor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. Alg. | n=10 | n=20 | n=30 | n=40 | n=50 |
| 1 | Iteratii: 177  Timp: 147673 nS | Iteratii: 21895  Timp: 1097956 nS | Iteratii: 2692542  Timp: 19324865 nS | Iteratii: 331160286  Timp: 1158067109 nS | Iteratii: 40730022153  Timp: 137109657596 nS |
| 2 | Iteratii: 10  Timp: 69153 nS | Iteratii: 20  Timp: 5459 nS | Iteratii: 30  Timp: 7470 nS | Iteratii: 40  Timp: 22032 nS | Iteratii: 50  Timp: 5976 nS |
| 3 | Iteratii: 4  Timp: 317797 nS | Iteratii: 5  Timp: 23138 nS | Iteratii: 5  Timp: 17167 nS | Iteratii: 6  Timp: 20870 nS | Iteratii: 6  Timp: 16807 nS |



**Concluzie**

In urma efectuarii acestei lucrari de laborator am analizat 3 metode diferite de calcul a sirului lui Fibonaci. In urma rularii programului am ajuns la concluzia ca pentru diferite situații putem folosi diferite tipuri de algoritmi. Pentru calcule a numerelor mici e eficient cel recursiv iar pentru calculele numerelor mari cel iterativ și al diviziunii e cu mult mai eficient. Deoarece timpul de executie este foarte mic si numarul de iteratii pentru numerele mari nu creste foarte repede.