O APLICAȚIE PENTRU PARTIȚII FIBONACCI

Radu-Ioan MIHAI[1]

Maria-Raluca STĂNESCU[2]

Abstract

În acest articol vom prezenta o definiție a partițiilor Fibonacci, împreună cu o aplicație rezolvată.

Introducere

Teoria numerelor reprezintă una dintre cele mai importante ramuri ale algebrei, dar este extrem de folosită și cunoscută în domeniul informaticii.

Cunoscătorii cărților lui Mario Livio ([2], [3]) sau pasionații de cultură generală cunosc deja importanța Șirului lui Fibonacci, a aplicațiilor acestuia, dar și cât de răspândit este acesta în viața noastră de zi cu zi, dar și în natură ([1]).

Șirul lui Fibonacci ([4], [5]) $(F_n)_{n\geq 0}$ este șirul de numere 0, 1, 1, 2, 3, 5, ... cu termenii

$$F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, \dots$$

dat prin formula de recurență

$$F_{n+2} = F_{n+1} + F_n.$$

Definiții

Definim *o partiție a unui număr natural n* ca o sumă de numere naturale a căror sumă este egală cu n.

Denumim *o partiție Fibonacci* o partiție a unui număr Fibonacci ce conține doar numere distincte din Șirul lui Fibonacci.

Aplicație

Se citește din fișierul “data.in” un număr natural N.

[1] Student, Universitatea din București, Facultatea de Matematică și Informatică, Specializarea Informatică. E-mail: rimihai2001@gmail.com, radu.mihai4@s.unibuc.ro

[2] Student, Universitatea din București, Facultatea de Matematică și Informatică, Specializarea Informatică. E-mail: stanescu.raluca.thg@gmail.com, maria.stanescu4@s.unibuc.ro

Verificați dacă N este termen al Șirului lui Fibonacci. În caz afirmativ, afișați partițiile și numărul de partiții Fibonacci al acelui număr în fișierul “data.out”, altfel afișați câte partițiile numărului respectiv care au în componența sa doar termeni ai Șirului lui Fibonacci și numărul acestora în același fișier de ieșire.

Date de intrare: Un număr natural N ce se află în fișierul “data.in”.

Date de ieșireȘ Un număr natural ce se află în fișerul “data.out”, ce reprezintă numărul cerut în enunț.

Restricții**:** $0 \leq N \leq 28657$.

Soluția acestei probleme o vom da folosind metoda **Backtracking.**

Descriere: Verificăm dacă N este termen al Șirului lui Fibonacci și generăm toate partițiile Fibonacci ale numărului N. Apoi, calculăm cate partiții Fibonacci au fost generate.

Cod (C++):

```
#include <bits/stdc++.h>

using namespace std;

ifstream fin("data.in");
ofstream fout("data.out");

long long dp[200];
long long count = 0;

bool PP(long long x)
{
    long long s = sqrt(x);
    return (s*s == x);
}

long long fib(long n)
{
    return PP(5*n*n + 4) || PP(5*n*n - 4);
}

long long ordinfibonacci(long long n)//termenul n al sirului lui Fibonacci
{
    long long a = 1, b = 1, c, i;
    if( n == 0)
        return a;
    for(i = 2; i <= n; i++)
    {
        c = a + b;
        a = b;
        b = c;
    }
    return b;
}
```

```
long long minfib(long n)//cel mai mare numar Fibonacci, mai mic sau egal cu n
{
    long copn=n;
    if(fib(n)==1)
        return n;
    else
    {
        while(fib(copn)==0)
            copn--;
        return copn;
    }
}

void printpart(long long idx)
{
    for (long long i = 1; i < idx; i++)
    {
        cout << dp[i] << " ";
    }
    cout << endl;
}

long long partfib(long long idx)//verificare partitie Fibonacci
{
    long long ok=1,ct=1;
    while(ordinfibonacci(ct)<dp[1])
        ct++;
    for(long long i=1; i<idx; i++)
    {
        if(ordinfibonacci(ct)!=dp[i])
        {
            ok=0;
            break;
        }
        ct--;
    }
    return ok;
}

void solve(long long remSum, long long maxVal, long long idx, long long&
count)
{

    // Daca remSum == 0 inseamna ca suma este atinsa
    // deci se poate contoriza partitia
    if (remSum == 0)
    {
        //printpart(idx);
        count++;
        return;
    }
    // Vom porni de la maxVal care este
    // cea mai mare valoarea ce poate fi folosita in suma
    for (long long i = maxVal; i >= 1; i--)
    {
        if (i > remSum )
        {
```

```
            continue;
        }
        else if (i <=remSum)
        {
            if(fib(i)==0)
            {
                while(fib(i)==0)
                    i--;
            }
            dp[idx] = i;
            solve(remSum - i, i-1, idx + 1, count);
        }
    }
}

int main()
{
    long long n, count = 0;
    fin>>n;
    solve(n, n, 1, count);
    fout<<endl<<count;
    return 0;
}
```

Exemple

I

data.in

8

data.out

8

5 3

5 2 1

3

II

data.in

9

data.out

8 1

5 3 1

2

Observații

Aceasta problemă poate reprezenta un punct de plecare pentru viitoare probleme de informatică, ce pot fi selectate la concursuri pentru elevi sau studenții din anii începători, dar și un material util pentru predarea unor aplicații suplimentare a backtracking-ului.

Idei viitoare

În studiile și articolele viitoare, vom folosi rezultatele și aplicația din acest articol pentru a face legături între partițiile Fibonacci și partițiile Lucas, dar vom studia și alte modalități de a rezolva problema propusă într-un mod cât mai eficient.

Bibliografie

[1] T. Britz, *Fibonacci numbers: A brief history and counting problems,* Parabola **56 (1)** (2020), UNSW, Sydney, Australia

[2] M. Livio, *Este Dumnezeu matematician?,* Editura Humanitas, București, 2017

[3] M. Livio, *Secțiunea de aur: Povestea lui phi, cel mai uimitor număr,* Editura Humanitas, București, 2012

[4] E. Lucas, *Recherches sur plusieurs ouvrages de Léonard de Pise*, Imprimerie des sciences mathématiques et physiques, Roma, Italia, 1877

[5] Fibonacci numbers, *The On-Line Encyclopedia of Integer Sequences*, https://oeis.org/A000045 , accesat pe 13.02.2021.