INFORMATICĂ · CLASA A IX-A · ALGORITMI ELEMENTARI · LECȚIA 12

Șiruri recurente (Fibonacci) și prelucrarea secvențelor de valori

Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-08

CE VEI ȘTI DUPĂ LECȚIE

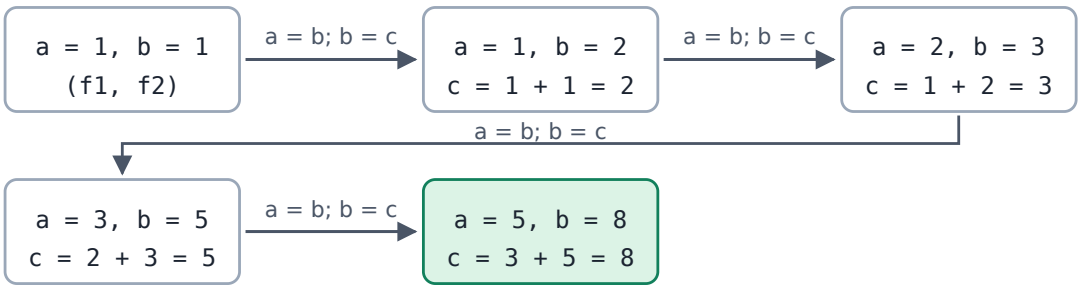
- Generezi termenii unui șir definit prin recurență (Fibonacci și variante) folosind doar 2–3 variabile.
- Verifici dacă un număr este termen Fibonacci.
- Prelucrezi o secvență de valori citite comparând elementul curent cu cel anterior (vârfuri, platouri, monotonie).
- Generezi șiruri cu o regulă dată (1, 2, 1, 3, 2, 1, ...).

1. Șiruri definite prin recurență

Un șir este definit **recurent** când fiecare termen se calculează din termenii anteriori. Exemplul celebru, șirul lui Fibonacci:

$f_1 = 1$, $f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$ → 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Nu avem nevoie să păstrăm tot șirul: ne ajung **ultimii doi termeni**. Imaginează-ți o fereastră cu două geamuri care „alunecă” peste șir.



păstrăm doar ultimii doi termeni, nu tot șirul

Fereastra Fibonacci

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int a = 1, b = 1;          // a = f(i-2), b = f(i-1)
    if (n >= 1) cout << a << ' ';
    if (n >= 2) cout << b << ' ';
    for (int i = 3; i <= n; i++)
    {
        int c = a + b;         // termenul nou
        cout << c << ' ';
        a = b;                 // „glisăm” fereastra
        b = c;
    }
    return 0;
}
```

Urmărire pentru `n = 6`:

i	a	b	c = a + b	se afișează	după glisare: a, b
(început)	1	1		1 1	
3	1	1	2	2	1, 2
4	1	2	3	3	2, 3
5	2	3	5	5	3, 5
6	3	5	8	8	5, 8

DE CE AȘA?

De ce `a = b; b = c;` în această ordine? `a` trebuie să primească vechea valoare a lui `b` **înainte** ca `b` să se schimbe. Dacă ai scrie `b = c; a = b;`, ambele ar ajunge egale cu `c` și ai pierde un termen. Același tipar ca la interschimbarea cu „pahar”.

ATENȚIE — GREȘELI FRECVENTE

Termenii cresc foarte repede: f_{47} depășește `int`, iar f_{93} depășește `long long`. Verifică restricțiile.

Termenii Fibonacci ≤ n

Când nu știi câți termeni sunt, dar știi unde te oprești, folosești `while`:

```cpp
int a = 1, b = 1;
while (a <= n)
{
    cout << a << ' ';
    int c = a + b;  a = b;  b = c;
}
```

Este x un termen Fibonacci?

Generăm termeni până ajungem la `x` sau îl depășim:

```
int a = 1, b = 1;
while (b < x) { int c = a + b; a = b; b = c; }
bool esteFibo = (x == 1 || b == x);
```

> **DE CE AȘA?**
>
> **De ce condiția `x == 1 ||`?** Dacă x = 1, bucla nu se execută deloc și `b` e deja 1, dar scriem explicit cazul pentru a nu depinde de inițializare. Pentru x = 4: bucla se oprește cu b = 5 > 4, deci `b != x`: nu e termen Fibonacci.

2. Recurențe în pseudocod la Bac

La Bac apar des formule de tipul „x ← x + y; y ← x − y" sau recurențe cu trei termeni. Metoda: **tabel de valori**, rând cu rând, cu toate variabilele.

> **LA BAC**
>
> Atenție la ordinea atribuirilor! Secvența `a ← b; b ← a + b` **nu** calculează Fibonacci, pentru că `a` a fost deja modificat. Scrie tabelul și o să vezi diferența.

Exemplu: `a ← 0; b ← 1`, apoi de 3 ori `scrie b; b ← a + b; a ← b − a`:

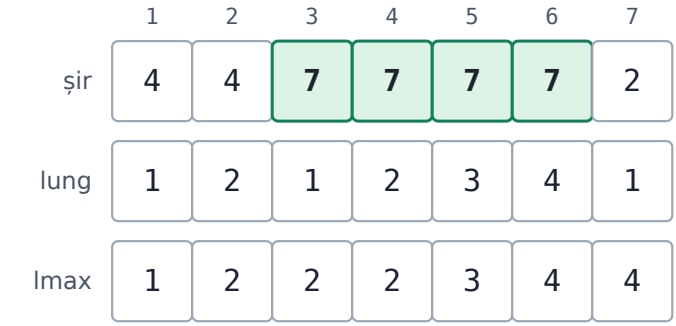
iterația	se scrie	b nou = a + b	a nou = b − a
1	1	0 + 1 = 1	1 − 0 = 1
2	1	1 + 1 = 2	2 − 1 = 1
3	2	1 + 2 = 3	3 − 1 = 2

Trucul: după `b ← a + b`, valoarea veche a lui `b` este `b − a`, deci `a ← b − a` pune în `a` vechiul `b`. Este un „schimb" fără variabilă auxiliară; apare ca grilă la BAC.

3. Secvențe de valori: compară cu elementul anterior

Multe probleme cer ceva despre **vecinii** din șir, fără să memorezi tot șirul.

Cea mai lungă secvență de elemente egale (platou)



ıng crește când x = ant și revine la 1 când secvența se rup

Platou

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n, x, ant;
    cin >> n >> ant;
    int lung = 1, lmax = 1;
    for (int i = 2; i <= n; i++)
    {
        cin >> x;
        if (x == ant) lung++;
        else lung = 1;                // începe o secvență nouă
        if (lung > lmax) lmax = lung;
        ant = x;
    }
    cout << lmax;
    return 0;
}
```

DE CE AȘA?

De ce `lung = 1` și nu `0` când secvența se rupe? Elementul curent `x` este el însuși începutul unei secvențe noi, deci are lungime 1. Și de ce `lmax` separat de `lung`? Pentru că `lung` se resetează: `lmax` păstrează cel mai bun rezultat văzut. Același tipar ca „maxim parțial”.

Același tipar pentru: cea mai lungă secvență crescătoare (`x > ant`), de numere pare (`x % 2 == 0`) etc.

Vârfuri (element mai mare decât ambii vecini)



7 și 9 sunt vârfuri; 8 nu (are un vecin egal)

Vârfuri

Păstrăm ultimele **trei** valori:

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n, a, b, c, nrVarfuri = 0;
    cin >> n >> a >> b;            // a = penultimul, b = ultimul citit
    for (int i = 3; i <= n; i++)
    {
        cin >> c;
        if (b > a && b > c) nrVarfuri++;
        a = b;  b = c;
    }
    cout << nrVarfuri;
    return 0;
}
```

citit c	a	b	`b > a && b > c` ?	nrVarfuri
3	2	7	da	1
8	7	3	nu	1
8	3	8	nu (8 = 8)	1
4	8	8	nu	1
9	8	4	nu	1
1	4	9	da	2

DE CE AȘA?

De ce bucla începe de la `i = 3`? Primele două valori au fost deja citite (în `a` și `b`). Primul și ultimul element nu au doi vecini, deci nu pot fi vârfuri; pentru n numere există n − 2 candidați.

Este șirul crescător?

```cpp
bool cresc = true;
cin >> ant;
for (int i = 2; i <= n; i++)
{
    cin >> x;
    if (x < ant) cresc = false;   // o singură „cădere” strică monotonia
    ant = x;
}
```

SFAT

La problema de eficiență de la Bac (III.3) apare des „cea mai lungă secvență cu proprietatea P” într-un fișier foarte mare. Tiparul de mai sus (o parcurgere, memorie constantă) este răspunsul de punctaj maxim.

4. Șiruri generate după o regulă

Exemplu: `1, 2, 1, 3, 2, 1, 4, 3, 2, 1, …` (grupa k conține k, k−1, ..., 1). Afișează primii n termeni:

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n, afisati = 0;
    cin >> n;
    for (int k = 1; afisati < n; k++)
        for (int j = k; j >= 1 && afisati < n; j--)
        {
            cout << j << ' ';
            afisati++;
        }
    return 0;
}
```

DE CE AȘA?

De ce două bucle și un contor `afisati` ? Șirul are o structură pe **grupe**: bucla exterioară alege grupa, cea interioară afișează elementele ei. Nu știm câte grupe încap în n termeni, deci ne oprim cu un contor, nu cu o limită fixă. Descompunerea „grupe + elemente" este gândirea de bază pentru șiruri generate.

5. Greșeli frecvente

Greșeala	Efect	Corect
`b = c; a = b;`	pierzi un termen	`a = b; b = c;`
`lung = 0` la ruperea secvenței	lungime cu 1 mai mică	`lung = 1`
uiți `ant = x;` la sfârșit	compari mereu cu primul element	actualizează `ant`
`int` pentru Fibonacci mare	depășire	`long long`
nu tratezi n = 1 sau n = 2	afișezi prea mulți termeni	`if (n >= 2)`

6. La BAC

LA BAC

Recurențele apar în grile (Fibonacci în secvențe de cod, „ce afișează...") și în Subiectul III cu fișier (platouri, secvențe). Pentru grile, tabelul de valori; pentru probleme, tiparul „element anterior + lungime curentă + lungime maximă".

7. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Scrie primii 12 termeni ai șirului $f_1 = 1$, $f_2 = 2$, $f_n = f_{n-1} + 2 \cdot f_{n-2}$.

2. Ce afișează pentru n = 6?

```
a ← 0
b ← 1
┌pentru i ← 1, n execută
│   scrie b
│   b ← a + b
│   a ← b - a
└■
```

3. Se citesc n numere. Afișează lungimea celei mai lungi secvențe de numere consecutive **pare**.

4. Câte vârfuri are șirul `1 5 2 2 6 3 3 8 1`?

5. Care este al 10-lea termen din Fibonacci (cu $f_1 = f_2 = 1$)?

6. Ce valori au `a` și `b` după secvența următoare?

```
int a = 3, b = 5;
a = b;
b = a + b;
```

Care este diferența față de `c = a + b; a = b; b = c;`?

7. Pentru șirul `1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, …`, care e al 15-lea termen?

8. Mai departe (pentru cine vrea mai mult)

Raportul de aur. Fibonacci ascunde un număr celebru: f_{n+1} / f_n se apropie de 1,618... (raportul de aur) pe măsură ce n crește. Calculează `13.0 / 8`, `21.0 / 13`, `34.0 / 21` și vezi cum converg. Apare în natură (spirala de floarea-soarelui), artă și arhitectură.

Recurență cu trei termeni. Tribonacci: `t = a + b + c; a = b; b = c; c = t;`. Fereastra are trei geamuri; ideea e aceeași. La problemele de vârfuri ai folosit deja o fereastră de trei.

Programare dinamică, preview. Multe probleme grele se rezolvă calculând o valoare din cele anterioare, ca la Fibonacci. De exemplu: „în câte moduri urci n trepte, urcând câte 1 sau 2?" are exact răspunsul f_{n+1}. Vei reîntâlni ideea în clasa a XI-a.

Optimizare avansată. Fibonacci poate fi calculat în `O(log n)` prin ridicarea la putere a unei matrici. Încearcă să cauți „Fibonacci matrix exponentiation" când vei învăța matrice.

Rezumat

- Recurențe: păstrezi doar ultimii termeni necesari și „glisezi" fereastra (`a = b; b = c;`).
- Ordinea atribuirilor contează: verifică cu tabel.
- Secvențe: compară elementul curent cu anteriorul; resetează lungimea la 1 când se rupe secvența; păstrează maximul separat.

- Vârfuri: fereastră de trei valori.

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] #255 **fibonacci** — ușoară
  Primii n termeni.
  https://www.pbinfo.ro/probleme/255/fibonacci
- [ ] #423 **fibonacci1** — ușoară
  Termenii mai mici sau egali cu n.
  https://www.pbinfo.ro/probleme/423/fibonacci1
- [ ] #424 **fibonacci-generalizat** — ușoară
  https://www.pbinfo.ro/probleme/424/fibonacci-generalizat
- [ ] #277 **varfuri** — medie
  Ai nevoie de ultimele trei valori citite.
  https://www.pbinfo.ro/probleme/277/varfuri

Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #256 **fiboverif** — medie
  https://www.pbinfo.ro/probleme/256/fiboverif
- [ ] #2420 **easyrow** — ușoară
  https://www.pbinfo.ro/probleme/2420/easyrow
- [ ] #56 **numarareperechi** — medie
  https://www.pbinfo.ro/probleme/56/numarareperechi
- [ ] #1875 **platou1** — ușoară
  Platou de lungime maximă (date din fișier).
  https://www.pbinfo.ro/probleme/1875/platou1
- [ ] #1879 **platou2** — ușoară
  Aceeași idee, pe numere — o singură parcurgere.
  https://www.pbinfo.ro/probleme/1879/platou2

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #257 **fibosum** — medie
  https://www.pbinfo.ro/probleme/257/fibosum

- [ ] #243 **sir1** — dificilă
  https://www.pbinfo.ro/probleme/243/sir1

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #1273 **Exercițiul #1273**
  șirul lui Fibonacci într-o secvență (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1273
- [ ] #1053 **Exercițiul #1053**
  secvență cu numere naturale (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1053
- [ ] #1062 **Exercițiul #1062**
  secvență cu numere naturale, variantă (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1062
- [ ] #1059 **Exercițiul #1059**
  bucle imbricate cu i<7 (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1059
- [ ] #920 **Exercițiul #920**
  secvență de cod cu n natural
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=920

Citește și exersează în plus

- Șirul lui Fibonacci https://www.pbinfo.ro/articole/5537/sirul-lui-fibonacci
- Python Tutor pentru C++ — vezi cum se „glisează” a și b la fiecare iterație
  https://pythontutor.com/cpp.html
- OEIS — The On-Line Encyclopedia of Integer Sequences — caută orice șir de numere (de exemplu 1, 1, 2, 3, 5, 8) și vezi toate legăturile lui https://oeis.org/
- CSES — Increasing Array — o problemă de parcurgere cu element anterior, în engleză
  https://cses.fi/problemset/task/1094

Răspunsuri

Exersează pe hârtie

1. 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 ($f_3 = 2 + 2 \cdot 1 = 4$; șirul este 2^{n-1}).

2. `1 1 2 3 5 8` (trucul `a ← b − a` păstrează vechiul b în a, deci e Fibonacci).

3.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n, x, lung = 0, lmax = 0;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> x;
        if (x % 2 == 0) lung++;
        else lung = 0;           // aici 0: un număr impar nu face parte dintr-o secvență de pare
        if (lung > lmax) lmax = lung;
    }
    cout << lmax;
    return 0;
}
```

Observă că aici resetarea este `lung = 0`, nu 1: un element impar nu începe o secvență de numere pare.

4. 3 vârfuri: 5 (între 1 și 2), 6 (între 2 și 3) și 8 (între 3 și 1). Elementele 2 și 3 nu sunt vârfuri (au un vecin egal).

5. f_{10} = **55** (1, 1, 2, 3, 5, 8, 13, 21, 34, 55).

6. `a = 5`, `b = 10` (b = a + b folosește noul a = 5). Cu `c = a + b; a = b; b = c;` ar fi `a = 5`, `b = 8`. Diferența este ordinea: în prima variantă `a` a fost modificat înainte de a fi folosit.

7. `1`. Grupele sunt (1), (2 1), (3 2 1), (4 3 2 1), (5 4 3 2 1): primele patru au 1 + 2 + 3 + 4 = 10 termeni, iar termenii 11–15 sunt 5 4 3 2 1; al 15-lea este ultimul din grupa 5.