INFORMATICĂ · CLASA A IX-A · ALGORITMI · LECȚIA 14

Eficiența algoritmilor și recapitularea clasei a IX-a

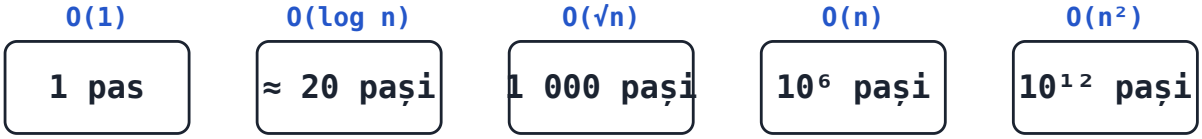
Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-10, ix-12

CE VEI ȘTI DUPĂ LECȚIE

- Estimezi câte operații face un algoritm (O(1), O(log n), O(√n), O(n), O(n²)).
- Alegi varianta eficientă pentru probleme clasice (sumă cu formulă, divizori până la √n, o singură parcurgere).
- Rezolvi grile de tip Subiectul I pe materia clasei a IX-a.
- Știi ce trebuie să scrii la descrierea metodei eficiente (III.3 a).

1. De ce contează eficiența

Un calculator face aproximativ **10^8 operații simple pe secundă**. Pe pbinfo și la olimpiade limita de timp e de obicei 0,1–1 secunde. De aceea două programe care dau același răspuns pot fi unul „corect" și celălalt „depășit de timp".



la 10^8 operații pe secundă: primele patru sunt instant, ultima durează ore

Câți pași face un program

Complexitate	n = 1 000	n = 1 000 000	n = 10^9	Exemplu
O(1)	1	1	1	formula lui Gauss
O(log n)	~10	~20	~30	parcurgerea cifrelor, căutarea binară
O(√n)	~32	1 000	~31 600	divizori, primalitate
O(n)	1 000	10^6	10^9 (prea lent)	o parcurgere a datelor
O(n²)	10^6	10^{12} (imposibil)	—	două bucle imbricate

DE CE AȘA?

De ce notația O(...) și nu secunde? Secundele depind de calculator; numărul de pași în funcție de `n` nu. O(n) înseamnă „timpul crește cam cât crește n": dacă îl dublezi pe n, timpul se dublează. La O(n²), dublezi n și timpul se **împătrește**. Asta decide dacă un program „supraviețuiește" când datele cresc.

SFAT

Uită-te la restricții înainte să scrii codul. Dacă $n \le 10^9$, orice buclă până la n e prea lentă. Dacă $n \le 10^5$, o soluție $O(n^2)$ (10^{10} operații) nu trece.

2. Cum număr operațiile

- o buclă `for i = 1..n` → n pași;
- bucle imbricate → se înmulțesc;
- o buclă care împarte la 10 (cifre) → câte cifre are n ≈ $\log_{10}$ n;
- o buclă până la √n → √n pași.

Exemplu pentru $n = 10^6$:

Cod	Pași	Timp aproximativ
`for (i = 1; i <= n; i++)`	10^6	0,01 s
`for (i...) for (j...)` pe n	10^{12}	10^4 s ≈ 3 ore
`while (n > 0) { n /= 10; }`	7	instant
`for (d = 2; d * d <= n; d++)`	1000	instant

3. Exemple: varianta lentă vs. varianta eficientă

Problemă	Lent	Eficient
1 + 2 + ... + n	buclă, O(n)	n(n+1)/2, O(1)
n prim?	d de la 2 la n−1	d · d ≤ n
numărul lipsă din 1..n	pentru fiecare k, caută-l în șir: $O(n^2)$	suma așteptată − suma citită: O(n)
suma 1! + 2! + ... + n!	recalculează fiecare factorial: $O(n^2)$	actualizează factorialul: O(n)
maximul a n numere	sortare	o parcurgere

DE CE AȘA?

De ce „suma așteptată − suma citită" bate căutarea? Dacă lipsește un singur număr din 1..n, suma citită este cu exact acel număr mai mică decât n(n+1)/2. Un singur calcul înlocuiește n căutări. Ideea generală: caută o **proprietate globală** (sumă, produs, xor) în loc să verifici fiecare element.

4. Problema III.3 de la Bac: cum se punctează

Enunțul are două părți:

- **a)** descrii în limbaj natural metoda și **justifici eficiența** (2 puncte);

- **b)** scrii programul C++ (8 puncte), din care o parte se acordă doar dacă algoritmul e eficient.

Un algoritm este considerat eficient dacă folosește **puțină memorie** (de multe ori fără vectori mari, prelucrând datele „din mers") și **un timp liniar** (o singură parcurgere a fișierului).

> **LA BAC**
>
> Model de răspuns pentru a): „Se citesc numerele din fișier unul câte unul, fără a fi memorate. Pentru fiecare număr citit se actualizează ... . Algoritmul este eficient din punctul de vedere al timpului de executare, deoarece datele sunt parcurse o singură dată (complexitate liniară), și din punctul de vedere al memoriei, deoarece se folosesc doar câteva variabile simple, indiferent de numărul de valori din fișier."

5. Simulare de examen (60 de minute)

Cronometrează-te. Regula: lucrează singur, **pe hârtie**, apoi verifică la finalul lecției.

Subiectul I (grile, 5 minute fiecare, maximum 25 de minute)

1. Expresia C++ `17 / 3 * 3 + 17 % 3` are valoarea:
a) 15
b) 17
c) 18
d) 6

2. Care expresie are valoarea 1 dacă și numai dacă `x` este număr **par de două cifre**?
a) `x % 2 == 0 || (x >= 10 && x <= 99)`
b) `x % 2 == 0 && x >= 10 && x < 100`
c) `!(x % 2) || x / 100 == 0`
d) `x / 10 % 2 == 0`

3. Variabila `n` are valoarea 2024. Valoarea expresiei `n / 100 + n % 100` este:
a) 44
b) 2024
c) 4
d) 20

4. De câte ori se afișează `*`? `for (i = 1; i <= 20; i += 4) cout << '*';`
a) 4
b) 5
c) 6
d) 20

5. Care este valoarea lui `s` după secvență? `s = 0; for (i = 1; i <= 5; i++) if (i % 2) s += i; else s -= i;`
a) 3
b) −3
c) 15
d) 9

6. Expresia echivalentă cu `!(a <= b && c != 0)` este:
a) `a > b && c == 0`
b) `a > b || c == 0`
c) `a >= b || c == 0`
d) `a < b || c != 0`

7. Numărul 45 scris în baza 2 este:
a) 101101
b) 110101
c) 100111
d) 101011

8. De câte ori se execută corpul pentru `n = 100`? `for (i = 1; i <= n; i++) for (j = i; j <= n; j++) cnt++;`
a) 100
b) 5050
c) 10000
d) 4950

9. Cea mai eficientă verificare de primalitate pentru $n \le 10^9$ este:
a) `for (d = 2; d < n; d++)`
b) `for (d = 2; d <= n / 2; d++)`
c) `for (d = 2; d * d <= n; d++)`
d) `for (d = 2; d <= n; d += 2)`

10. Ce valoare are `cmmdc(1000, 625)`?
a) 25
b) 125
c) 625
d) 5

Subiectul II (pseudocod, 20 de minute)

```
citește n                    (număr natural nenul)
p ← 1
k ← 0
┌cât timp n > 0 execută
│   c ← n % 10
│   ┌dacă c ≠ 0 atunci
│   │   p ← p * c
│   │   k ← k + 1
│   └■
│   n ← [n / 10]
└■
scrie p, " ", k
```

a) Ce se afișează pentru `n = 20305`? (completează un tabel de urmărire)
b) Scrie două numere distincte, de exact 3 cifre, pentru care se afișează `6 3`.
c) Scrie programul C++ corespunzător.
d) Scrie un algoritm echivalent care folosește `repetă … până când`.

Subiectul III (15 minute)

Se citesc numere naturale până la apariția lui `0` (cel puțin un număr nenul). Afișează cea mai mare **diferență în valoare absolută** dintre două numere citite **consecutiv**.

Exemplu: pentru `3 8 2 10 0` se afișează `8` ($|10 - 2|$).

a) Descrie metoda și justifică eficiența (2–3 rânduri).
b) Scrie programul C++.

6. Fișă de recapitulare a clasei a IX-a

- **Operatori:** `/` întreg, `%`, prioritatea, `&& || !`, De Morgan.
- **Structuri:** if / else, while, do-while, for; transformarea `cât timp` ↔ `repetă … până când`.
- **Cifre:** `n % 10`, `n / 10`; oglindit; număr nou cu `p = 1, 10, 100…`.
- **Divizibilitate:** divizori până la √n; prim; Euclid; cmmmc; factori primi.
- **Șiruri:** sume, produse, numărări, maxim/minim, două maxime, Fibonacci, secvențe cu elementul anterior.
- **Baze:** împărțiri repetate, Horner.
- **Eficiență:** O(1) < O(log n) < O(√n) < O(n) < O(n^2).

> **DE ȘTIUT**
>
> Pentru fiecare greșeală din simulare, notează în caiet **regula** pe care ai uitat-o (de exemplu „`/` între întregi taie zecimalele") și întoarce-te la lecția respectivă.

7. Mai departe (pentru cine vrea mai mult)

Cazul cel mai rău contează. Notația O(...) descrie mereu cazul **cel mai defavorabil**. Un program care verifică primalitatea unui număr compus poate termina după 2 pași, dar pentru un număr prim va face toți √n pași: aceasta e complexitatea.

Complexitate versus memorie. Uneori cumperi viteză cu memorie: un **vector de frecvență** (`f[x]++`) transformă o căutare O(n) într-una O(1). Vei vedea în clasa a X-a cum „cheltui" memorie ca să economisești timp.

Verifică-ți programul cu un test mare. Generează o intrare cu n = 10^6 și măsoară timpul (`clock()` din `<ctime>`): dacă durează sub 1 secundă, ești în limite. Antrenamentul acesta te scapă de surprizele de la evaluarea automată.

De la O(n^2) la O(n log n). Sortarea naivă face O(n^2); algoritmii buni (merge sort, `sort` din C++) fac O(n log n): pentru n = 10^6 înseamnă $2 \cdot 10^7$ pași față de 10^{12}. Este diferența dintre „trece" și „nu trece", și motivul pentru care sortarea apare în clasa a X-a.

Rezumat

- Un program face ~10^8 operații pe secundă; estimează numărul de pași **înainte** să scrii codul.
- O(1) < O(log n) < O(√n) < O(n) < O(n^2); buclele imbricate se înmulțesc.
- Divizori și primalitate: până la √n. Gauss și formulele înlocuiesc buclele.

- La III.3: o parcurgere, memorie constantă, justificare scurtă.

Probleme de lucru (pbinfo)

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

- [ ] #1360 **suma-gauss** — ușoară
  Formulă în loc de buclă.
  https://www.pbinfo.ro/probleme/1360/suma-gauss
- [ ] #1255 **lipsa** — ușoară
  Numărul lipsă: suma așteptată minus suma citită.
  https://www.pbinfo.ro/probleme/1255/lipsa
- [ ] #3977 **expon** — ușoară
  https://www.pbinfo.ro/probleme/3977/expon

Temă (pbinfo)

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

- [ ] #1284 **carte1** — ușoară
  https://www.pbinfo.ro/probleme/1284/carte1
- [ ] #482 **auto** — ușoară
  https://www.pbinfo.ro/probleme/482/auto
- [ ] #3072 **sumprodmax3** — ușoară
  https://www.pbinfo.ro/probleme/3072/sumprodmax3
- [ ] #4493 **minnm** — medie
  https://www.pbinfo.ro/probleme/4493/minnm
- [ ] #1798 **trepte** — medie
  Recunoști un șir celebru?
  https://www.pbinfo.ro/probleme/1798/trepte

Grile din subiectele de BAC (pbinfo)

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

- [ ] #1377 **Exercițiul #1377**
  22/20*20/2: ordinea operațiilor (Bac 2022)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1377
- [ ] #1738 **Exercițiul #1738**
  20/25*20/2 (Bac 2026)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1738

- [ ] #61 **Exercițiul #61**
  !((x>=5)&&(x<10)): De Morgan (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=61

- [ ] #1245 **Exercițiul #1245**
  instrucțiune if echivalentă cu x = -abs(y) (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1245

- [ ] #1414 **Exercițiul #1414**
  n=19; while(n!=0): ce afișează (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1414

- [ ] #1262 **Exercițiul #1262**
  maxime și minime într-o secvență (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1262

- [ ] #1355 **Exercițiul #1355**
  secvență do-while (Bac 2022)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1355

- [ ] #1693 **Exercițiul #1693**
  expresia care completează o buclă (Bac 2025)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1693

Citește și exersează în plus

- Eficiența algoritmilor https://www.pbinfo.ro/articole/39626/eficienta-algoritmilor
- infoarena — probleme și concursuri — platforma clasică de olimpiade, cu arhivă de probleme pe categorii https://infoarena.ro/
- CSES Problem Set — probleme cu restricții mari, ideale pentru a exersa estimarea timpului https://cses.fi/problemset/
- Big-O Cheat Sheet — tabel de complexități (engleză), util pentru clasele X–XI https://www.bigocheatsheet.com/
- Python Tutor pentru C++ — numără pașii unui program rulat pe un exemplu mic https://pythontutor.com/cpp.html

Răspunsuri

Subiectul I

1. b) 17 (17 / 3 = 5, apoi 5 · 3 = 15, apoi + 17 % 3 = 2).

2. b). Varianta a) e adevărată pentru orice număr par; c) pentru orice număr cu o cifră sau 3 cifre; d) e adevărată pentru 0–9, 20–29 etc.

3. a) 44 (20 + 24).

4. b) 5 (i = 1, 5, 9, 13, 17).

5. a) 3 (+1 −2 +3 −4 +5).

6. b) `a > b || c == 0` (De Morgan: `!(A && B)` = `!A || !B`).

7. a) 101101 (32 + 8 + 4 + 1).

8. b) 5050 (100 + 99 + … + 1).

9. c). Celelalte au O(n) pași, imposibil pentru n = 10^9.

10. b) 125 (1000 = $2^3 \cdot 5^3$, 625 = 5^4; cmmdc = 5^3).

Subiectul II

a) Pentru `n = 20305`:

n	c	`c ≠ 0`?	p	k
20305	—	—	1	0
→ 2030	5	da	5	1
→ 203	0	nu	5	1
→ 20	3	da	15	2
→ 2	0	nu	15	2
→ 0	2	da	30	3

Se afișează `30 3`: produsul cifrelor nenule și câte sunt.

b) De exemplu **123** și **321** (1·2·3 = 6, trei cifre nenule). Merge și 611 sau 116 (1·1·6 = 6).

c)

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

int main()
{
    int n, p = 1, k = 0, c;
    cin >> n;
    while (n > 0)
    {
        c = n % 10;
        if (c != 0)
        {
            p = p * c;
            k = k + 1;
        }
        n = n / 10;
    }
    cout << p << " " << k;
    return 0;
}
```

d)

```
citește n
p ← 1
k ← 0
┌dacă n > 0 atunci
│  ┌repetă
│  │   c ← n % 10
│  │   ┌dacă c ≠ 0 atunci
│  │   │   p ← p * c
│  │   │   k ← k + 1
│  │   └■
│  │   n ← [n / 10]
│  └până când n ≤ 0
└■
scrie p, " ", k
```

Enunțul garantează `n` nenul, deci `dacă`-ul de la început se poate omite.

Subiectul III

a) Se citesc numerele unul câte unul, fără a fi memorate. Se păstrează valoarea anterioară și cea mai mare diferență văzută până acum; la fiecare număr nou se calculează diferența în modul față de cel anterior și se actualizează maximul. Algoritmul parcurge datele o singură dată (timp liniar) și folosește doar câteva variabile (memorie constantă).

b)

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

int main()
{
    int ant, x, dmax = 0;
    cin >> ant;
    cin >> x;
    while (x != 0)
    {
        int d = x > ant ? x - ant : ant - x;
        if (d > dmax) dmax = d;
        ant = x;
        cin >> x;
    }
    cout << dmax;
    return 0;
}
```
