INFORMATICĂ · CLASA A IX-A · ALGORITMI ELEMENTARI · LECȚIA 10

Divizibilitate: divizori, numere prime, cmmdc și cmmmc

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

CE VEI ȘTI DUPĂ LECȚIE

- Parcurgi divizorii unui număr eficient, până la √n.
- Verifici dacă un număr este prim în O(√n).
- Calculezi cmmdc cu algoritmul lui Euclid (prin împărțiri și prin scăderi) și cmmmc.
- Recunoști numerele perfecte, prime între ele și alte proprietăți uzuale.

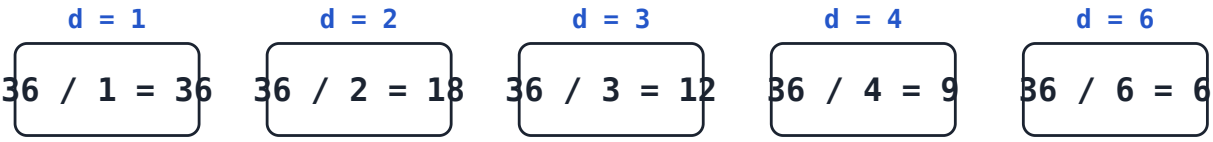
1. Divizori: varianta simplă și varianta eficientă

`d` este divizor al lui `n` dacă `n % d == 0`.

Varianta simplă (n pași): încerci toate numerele de la 1 la n.

```
for (int d = 1; d <= n; d++)
    if (n % d == 0) cout << d << ' ';
```

Varianta eficientă (√n pași): divizorii vin **în perechi** `(d, n / d)`. Dacă `d ≤ √n`, perechea lui e `≥ √n`. Deci e de ajuns să căutăm până la √n, adică până când `d · d > n`.



ne oprim când d · d > n; la d = 6 perechea e chiar 6 (numărat o singură dată)

Perechile de divizori

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

int main()
{
    int n, nrDiv = 0, sumaDiv = 0;
    cin >> n;
    for (int d = 1; d * d <= n; d++)
        if (n % d == 0)
        {
            nrDiv++;  sumaDiv += d;
            if (d != n / d) { nrDiv++;  sumaDiv += n / d; }   // perechea, dacă e diferită
        }
    cout << nrDiv << " " << sumaDiv;
    return 0;
}
```

Urmărire pentru `n = 12`:

d	`d * d <= 12` ?	`12 % d == 0` ?	perechea n/d	nrDiv	sumaDiv
1	da	da	12	2	13
2	da	da	6	4	21
3	da	da	4	6	28
4	nu (16 > 12) → stop			6	28

DE CE AȘA?

De ce se poate opri la √n? Dacă `d · e = n` și ambii ar fi mai mari decât √n, produsul ar depăși n. Deci cel puțin unul dintre ei e `≤ √n`: găsind pe cel mic, îl obții pe cel mare ca `n / d`. Pentru n = 10^9 treci de la un miliard de pași la circa 31 600.

ATENȚIE — GREȘELI FRECVENTE

Pentru pătratele perfecte (de exemplu 36 = 6 · 6), perechea lui 6 este tot 6: nu o număra de două ori. De aici condiția `d != n / d`.

SFAT

Un număr are un **număr impar de divizori** dacă și numai dacă este **pătrat perfect** (unicul divizor fără pereche este √n). Apare des la grile.

2. Numere prime

Un număr este **prim** dacă are exact doi divizori: 1 și el însuși. 0 și 1 **nu** sunt prime; 2 este singurul număr prim par.

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

int main()
{
    int n;
    cin >> n;
    bool prim = (n >= 2);
    for (int d = 2; d * d <= n && prim; d++)
        if (n % d == 0)
            prim = false;
    cout << (prim ? "DA" : "NU");
    return 0;
}
```

n	d încercați	rezultat
97	2, 3, 4, 5, 6, 7, 8, 9 (10·10 = 100 > 97)	niciun divizor → **prim**
91	2, 3, 4, 5, 6, 7 → `91 % 7 == 0`	**nu e prim** (7 · 13)

Pentru n ≈ 10^9, verificarea face cel mult ~31 623 pași în loc de un miliard.

DE CE AȘA?

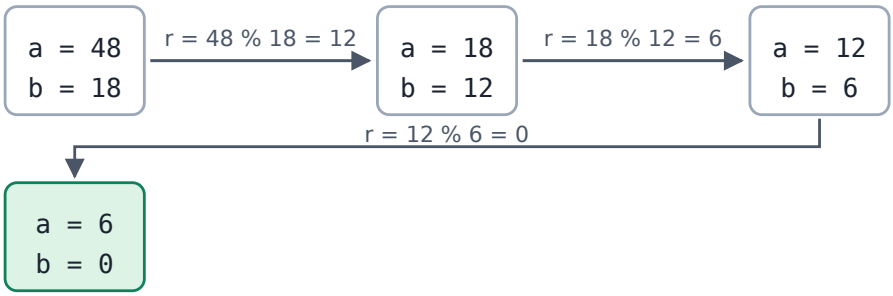
De ce variabila `prim` și nu `break`? Ambele merg. Cu `prim` în condiția buclei (`&& prim`) ieșirea e vizibilă chiar în antet, iar la BAC se cere des „fără instrucțiuni break/continue". De reținut: pornim cu „prim = adevărat" și căutăm **un contra-exemplu**; un singur divizor îl dărâmă.

LA BAC

La Subiectul III (eficiență) se punctează diferit o verificare de primalitate până la n, până la n/2 sau până la √n. Folosește **mereu** varianta `d * d <= n`.

3. Cel mai mare divizor comun: algoritmul lui Euclid

Ideea: orice număr care divide pe `a` și pe `b` divide și restul `a % b`. Deci `cmmdc(a, b) = cmmdc(b, a % b)`, iar problema devine mai mică la fiecare pas, până când restul e 0.



Euclid pentru 48 și 18

Prin împărțiri (rapid):

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

int main()
{
    int a, b;
    cin >> a >> b;
    while (b != 0)
    {
        int r = a % b;
        a = b;
        b = r;
    }
    cout << a;                // cmmdc
    return 0;
}
```

pas	a	b	r = a % b
1	48	18	12
2	18	12	6
3	12	6	0
4	6	0	(b = 0: stop) → cmmdc = **6**

DE CE AȘA?

De ce rămâne `a` și nu `b` la final? Când `b` devine 0, ultimul rest nenul s-a mutat în `a` prin `a = b`. Din același motiv `cmmdc(a, 0) = a`: orice număr se divide cu el însuși, iar 0 se divide cu orice.

Prin scăderi repetate (apare la Bac în pseudocod, trebuie să-l recunoști):

```
┌cât timp a ≠ b execută
│   ┌dacă a > b atunci a ← a - b
│   │altfel b ← b - a
│   └■
└■
scrie a
```

CAPCANĂ

Varianta cu scăderi intră în buclă infinită dacă unul dintre numere este 0. Varianta cu împărțiri funcționează și atunci: cmmdc(a, 0) = a.

Cel mai mic multiplu comun

`cmmmc(a, b) = a · b / cmmdc(a, b)`. Ca să nu depășești, împarte întâi:

```cpp
long long cmmmc = 1LL * a / g * b;   // g = cmmdc(a, b), calculat pe copii ale lui a și b
```

> **DE CE AȘA?**
>
> **De ce `a / g * b` și nu `a * b / g`?** Rezultatul e același, dar `a * b` poate depăși `int` înainte să împarți. `a / g` este întreg (g îl divide pe a), apoi înmulțești cu `b`. Ordinea operațiilor contează la numere mari.

Proprietăți utile

- `a` și `b` sunt **prime între ele** dacă `cmmdc(a, b) = 1`.
- Fracția `a/b` se simplifică împărțind ambii termeni la `cmmdc(a, b)`.
- cmmdc pentru n numere: `g = cmmdc(g, x)` pentru fiecare număr nou.

4. Alte numere „cu nume”

Denumire	Definiție	Exemple
perfect	egal cu suma divizorilor săi mai mici decât el	6, 28, 496
prime între ele	cmmdc = 1	8 și 15
pătrat perfect	`r * r == n`, cu `r = (int)sqrt(n)`	49, 144
prime gemene	prime cu diferența 2	11 și 13
numere prietene	fiecare = suma divizorilor proprii ai celuilalt	220 și 284

Pentru pătrat perfect, verifică vecinii lui `r` ca să eviți erorile de rotunjire ale lui `sqrt`:

```
int r = sqrt(n);
while (r * r > n) r--;
while ((r + 1) * (r + 1) <= n) r++;
bool patrat = (r * r == n);
```

5. Greșeli frecvente

Greșeala	Efect	Corect
`d <= n` în loc de `d * d <= n`	prea lent la n mare	`d * d <= n`
1 declarat prim	răspuns greșit	`n >= 2`
pătrat perfect numărat de două ori	un divizor în plus	`d != n / d`
`a * b / g` la numere mari	depășire	`a / g * b`
Euclid prin scăderi cu 0	buclă infinită	împărțiri

6. La BAC

> **LA BAC**
>
> Subiectul I conține adesea o grilă cu **divizori** (câți, care e cel mai mare) și una cu **pseudocod pe cmmdc** (ce afișează pentru a, b date). Subiectul III cere un program cu verificare de primalitate sau numărarea divizorilor la valori mari (n până la $2 \cdot 10^9$): numai varianta `d * d <= n` ia punctajul complet.

7. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Ce afișează pentru `a = 12`, `b = 30`? Ce calculează?

```
citește a, b
┌cât timp a ≠ b execută
│   ┌dacă a > b atunci a ← a - b
│   │altfel b ← b - a
│   └■
└■
scrie a
```

2. Câte numere din intervalul [1, 100] au exact 3 divizori? (Indiciu: pătrate de numere prime.)

3. Câți divizori are 36? Este un număr impar? De ce?

4. Calculează `cmmdc(84, 36)` și `cmmmc(84, 36)` cu Euclid.

5. De câte ori se execută corpul buclei `for (d = 2; d * d <= 97; d++)` pentru `n = 97`? Este 97 prim?

6. Ce afișează Euclid prin împărțiri pentru `a = 18`, `b = 0`? Dar varianta prin scăderi?

7. Simplifică fracția 45/60 folosind cmmdc.

8. Care dintre numerele 12 și 28 este perfect?

9. Scrie un program eficient care afișează câți divizori are `n ≤ 2 000 000 000` (indiciu: ce tip folosești pentru `d * d`?).

8. Mai departe (pentru cine vrea mai mult)

Ciurul lui Eratostene. Pentru **toate** numerele prime până la `n` nu verifici fiecare număr separat. Marchezi multiplii fiecărui prim:

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

erde = prim; roșu = tăiat (multiplu al unui prim mai mic); 1 nu este prin

Ciurul lui Eratostene

```
vector<bool> prim(n + 1, true);
prim[0] = prim[1] = false;
for (int i = 2; i * i <= n; i++)
    if (prim[i])
        for (int j = i * i; j <= n; j += i)
            prim[j] = false;
```

Lucrează în aproximativ `n · log log n` pași: numerele prime până la 10 milioane în sub o secundă. Îl vei folosi în clasa a X-a la vectori.

Criptografie. Securitatea cumpărăturilor online (RSA) se bazează pe un fapt simplu: **înmulțirea** a două numere prime uriașe e ușoară, dar **descompunerea** produsului în factori e imposibilă în timp rezonabil. Tot ce ai învățat în lecție (primalitate, cmmdc, Euclid extins) este fundația matematică a ei.

Euclid are 2300 de ani. Algoritmul apare în „Elementele" lui Euclid (aproximativ 300 î.Hr.) și este considerat cel mai vechi algoritm încă în folosință. Pentru n numere se aplică repetat: `g = cmmdc(g, x)`.

Teorema lui Euclid: există o infinitate de numere prime. Demonstrația are patru rânduri: presupune că ai o listă finită de prime, înmulțește-le și adaugă 1. Noul număr nu se divide cu niciunul din listă, deci are un factor prim care nu era în listă: lista nu putea fi completă.

Rezumat

- Divizori până la √n, în perechi `(d, n/d)`; atenție la pătratele perfecte.
- Prim: `n ≥ 2` și niciun divizor în `[2, √n]`.
- Euclid: `r = a % b; a = b; b = r;` până când `b == 0`.
- `cmmmc = a / cmmdc * b`.
- Numărul de divizori este impar exact pentru pătrate perfecte.

Probleme de lucru (pbinfo)

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

- [ ] **#45** **verifprim** — ușoară
  Verificare cu d · d ≤ n.
  https://www.pbinfo.ro/probleme/45/verifprim

- [ ] #376 **suma-divizori** — ușoară
  Divizorii în perechi (d, n/d).
  https://www.pbinfo.ro/probleme/376/suma-divizori
- [ ] #58 **cmmdc** — ușoară
  Euclid prin împărțiri.
  https://www.pbinfo.ro/probleme/58/cmmdc
- [ ] #59 **cmmmc** — ușoară
  a / cmmdc * b — în ordinea asta, ca să nu depășești.
  https://www.pbinfo.ro/probleme/59/cmmmc

Temă (pbinfo)

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

- [ ] #18 **numarul-de-divizori** — ușoară
  Citire din fișier (vezi lecția de fișiere sau cere-mi șablonul).
  https://www.pbinfo.ro/probleme/18/numarul-de-divizori
- [ ] #60 **primeintreele** — ușoară
  https://www.pbinfo.ro/probleme/60/primeintreele
- [ ] #61 **afisaredivizoricomuni** — ușoară
  https://www.pbinfo.ro/probleme/61/afisaredivizoricomuni
- [ ] #375 **urmatorul-prim** — ușoară
  https://www.pbinfo.ro/probleme/375/urmatorul-prim
- [ ] #64 **numarperfect** — ușoară
  https://www.pbinfo.ro/probleme/64/numarperfect
- [ ] #388 **numarul-divizorilor-pari** — ușoară
  https://www.pbinfo.ro/probleme/388/numarul-divizorilor-pari
- [ ] #391 **fractii** — ușoară
  Simplificarea fracțiilor cu cmmdc.
  https://www.pbinfo.ro/probleme/391/fractii
- [ ] #387 **suma-divizorilor-impari** — ușoară
  https://www.pbinfo.ro/probleme/387/suma-divizorilor-impari

Pentru cei care vor mai mult

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

- [ ] #1574 **prietene** — ușoară
  https://www.pbinfo.ro/probleme/1574/prietene

- [ ] #445 **pseudoperfect** — ușoară
  https://www.pbinfo.ro/probleme/445/pseudoperfect

Grile din subiectele de BAC (pbinfo)

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

- [ ] #661 **Exercițiul #661**
  pseudocod: numere naturale m, n, x (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=661
- [ ] #814 **Exercițiul #814**
  secvență cu while și i = 2 (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=814
- [ ] #821 **Exercițiul #821**
  x, y, d naturale strict pozitive (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=821
- [ ] #1264 **Exercițiul #1264**
  secvență pe numărul n (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1264
- [ ] #1620 **Exercițiul #1620**
  care secvențe afișează numărul tuturor divizorilor
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1620
- [ ] #1619 **Exercițiul #1619**
  câte cifre de 0 are la sfârșit n!
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1619

Citește și exersează în plus

- Divizibilitate https://www.pbinfo.ro/articole/66/divizibilitate
- Divizorii unui număr https://www.pbinfo.ro/articole/72/divizorii-unui-numar
- Verificarea primalității https://www.pbinfo.ro/articole/74/numere-prime
- Algoritmul lui Euclid https://www.pbinfo.ro/articole/73/algoritmul-lui-euclid
- Algopedia — divizibilitate și numere prime — wiki românesc de algoritmi, cu explicații pentru concursuri https://algopedia.ro/
- CSES — Introductory Problems — exersează gândirea eficientă (englez), fără teorie suplimentară https://cses.fi/problemset/
- VisuAlgo — Sieve / Number theory — animații interactive pentru algoritmi (caută secțiunea Math) https://visualgo.net/en

Răspunsuri

Exersează pe hârtie

1. `6`. Perechile (a, b): (12, 30) → (12, 18) → (12, 6) → (6, 6). Algoritmul calculează cmmdc(12, 30) = 6.

2. `4`: numerele 4, 9, 25, 49 (pătratele numerelor prime 2, 3, 5, 7). Un număr cu exact trei divizori este p^2, cu p prim: divizorii sunt 1, p, p^2.

3. 36 are 9 divizori: 1, 2, 3, 4, 6, 9, 12, 18, 36. Numărul e impar pentru că 36 este pătrat perfect: divizorul 6 nu are pereche diferită.

4. Euclid: (84, 36) → (36, 12) → (12, 0): cmmdc = **12**. cmmmc = 84 / 12 · 36 = 7 · 36 = **252**.

5. 8 execuții (d = 2, 3, 4, 5, 6, 7, 8, 9; la d = 10 avem 100 > 97). Niciunul nu divide pe 97, deci **97 este prim**.

6. Prin împărțiri: `18` (b e 0 din start, deci bucla nu se execută). Prin scăderi: **buclă infinită** (a ≠ b mereu, b – a se menține 0).

7. cmmdc(45, 60) = 15, deci 45/60 = 3/4.

8. 28: 1 + 2 + 4 + 7 + 14 = 28. Pentru 12: 1 + 2 + 3 + 4 + 6 = 16 ≠ 12.

9.

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

int main()
{
    long long n;
    int nr = 0;
    cin >> n;
    for (long long d = 1; d * d <= n; d++)
        if (n % d == 0)
        {
            nr++;
            if (d != n / d) nr++;
        }
    cout << nr;
    return 0;
}
```

`d * d` poate depăși `int` pentru n aproape de $2 \cdot 10^9$ (de exemplu d = 46 341 dă 2 147 488 281 > 2 147 483 647), deci `d` și `n` trebuie să fie `long long`.