INFORMATICĂ · PYTHON · CLASA A IX-A · PRELUCRĂRI ALE DATELOR NUMERICE · LECȚIA 6

cmmdc și algoritmul lui Euclid

Durată: 100 min Nivel: mate-info · Python Înainte: lecțiile py9-02

CE VEI ȘTI DUPĂ LECȚIE

- Calculezi cmmdc a două numere naturale cu algoritmul lui Euclid, în ambele variante (scăderi și împărțiri), și urmărești valorile pas cu pas.
- Compari cele două variante după numărul de pași și alegi varianta potrivită.
- Calculezi cmmmc pornind de la cmmdc și îl folosești în probleme practice.
- Simplifici fracții și faci operații cu două fracții date prin numărător și numitor.

1. Problema: cea mai mare placă de gresie

O sală are 360 cm × 270 cm. Vrem s-o acoperim cu plăci **pătrate**, **întregi**, toate la fel, cât mai mari. Latura plăcii trebuie să-l dividă și pe 360, și pe 270, deci căutăm **cel mai mare divizor comun**: `cmmdc(360, 270) = 90`. Ies 4 × 3 = 12 plăci de 90 cm.

Ai putea încerca toți divizorii de la cel mai mic dintre numere în jos, dar există o metodă mult mai rapidă, veche de peste 2000 de ani: **algoritmul lui Euclid**. Se bazează pe o observație simplă: dacă `d` divide și pe `a`, și pe `b`, atunci divide și pe `a - b` și pe restul `a % b`. Deci perechile `(a, b)`, `(a - b, b)` și `(b, a % b)` au **același** cmmdc, iar numerele devin tot mai mici.

2. Euclid prin scăderi repetate

Cât timp numerele sunt diferite, scazi pe cel mic din cel mare. Când devin egale, aceea este valoarea cmmdc.

```
a, b = map(int, input().split())    # numere naturale NENULE
while a != b:
    if a > b:
        a -= b
    else:
        b -= a
print(a)
```

Urmărire pentru `48 18`:

pas	a	b	ce se întâmplă
0	48	18	`a > b`, deci `a ← 48 - 18`
1	30	18	`a ← 30 - 18`
2	12	18	`b > a`, deci `b ← 18 - 12`
3	12	6	`a ← 12 - 6`
4	6	6	egale: se afișează **6**

ATENȚIE — GREȘELI FRECVENTE

Varianta cu scăderi **nu merge dacă un număr este 0**: pentru `a = 5`, `b = 0` scade mereu 0 din 5 și bucla nu se mai oprește. Testează întotdeauna și cazul cu 0.

3. Euclid prin împărțiri repetate

În loc să scazi de mai multe ori același `b`, iei direct restul. Cât timp `b` nu este 0, perechea `(a, b)` devine `(b, a % b)`. La final cmmdc este `a`.

```
a, b = map(int, input().split())
while b != 0:
    a, b = b, a % b      # atribuire simultană: totul din valorile vechi
print(a)
```

Urmărire pentru `48 18`:

pas	a	b	a % b
0	48	18	12
1	18	12	6
2	12	6	0
3	6	0	(bucla se oprește)

Se afișează **6**, după 3 pași în loc de 4. Varianta aceasta merge și când `b = 0` (`cmmdc(a, 0) = a`) și nici nu contează dacă `a < b`: primul pas le inversează singur (`18 48` devine `48 18`).

SFAT

În C++ aveai nevoie de o variabilă auxiliară: `r = a % b; a = b; b = r;`. În Python, `a, b = b, a % b` face același lucru într-un rând. Dacă scrii însă `a = b` și apoi `b = a % b`, ai pierdut vechiul `a`, iar `b` devine mereu 0.

4. Care variantă este mai rapidă?

Am numărat pașii (trecerile prin buclă) pentru câteva perechi:

a, b	cmmdc	pași, scăderi	pași, împărțiri
48, 18	6	4	3
123456, 7890	6	62	7
1000000, 1	1	999 999	1
1000000, 999999	1	999 999	2
89, 55	1	9	9

La scăderi, numărul de pași poate ajunge cât valoarea numerelor: pentru 10^9 și 1 ar fi aproape un miliard de pași, adică cam un minut de rulare în Python. La împărțiri, numărul de pași crește foarte

încet (proporțional cu numărul de **cifre**): pentru numere până la 10^9 sunt cel mult câteva zeci de pași. Cel mai rău caz pentru împărțiri apare la doi termeni consecutivi din șirul lui Fibonacci (ca `89, 55`), unde fiecare cât este 1 și cele două variante fac aceiași pași.

> **LA BAC**
>
> La Bac apar ambele variante în pseudocod, de obicei ca „ce afișează algoritmul pentru...”. Recunoaște-le după condiția buclei: $a \neq b$ înseamnă scăderi, $b \neq 0$ înseamnă împărțiri.

5. Cel mai mic multiplu comun

Între cmmdc și cmmmc există relația $\text{cmmdc}(a, b) \cdot \text{cmmmc}(a, b) = a \cdot b$. Deci:

```python
a, b = 60, 90
x, y = a, b
while y != 0:
    x, y = y, x % y
print(a // x * b)        # cmmmc = a / cmmdc · b; afișează 180
```

Exemplu. Un semafor are ciclul de 40 s roșu + 20 s verde = 60 s, altul 45 s + 45 s = 90 s. Pornesc împreună pe roșu. După $\text{cmmmc}(60, 90) = 180$ secunde sunt din nou amândouă la începutul ciclului.

Observă că am lucrat pe **copii** (`x`, `y`): Euclid „strică” valorile, iar pentru cmmmc avem nevoie de `a` și `b` inițiale.

> **DE ȘTIUT**
>
> Python are deja `math.gcd(a, b)` și `math.lcm(a, b)` (după `import math`). Le folosești ca **verificare** sau în proiecte, dar algoritmul trebuie să știi să-l scrii și să-l urmărești singur: asta se cere la evaluare și la Bac.

6. Aplicație: fracții

O fracție `a/b` este **ireductibilă** când $\text{cmmdc}(a, b) = 1$. Ca s-o simplifici, împarți numărătorul și numitorul la cmmdc. Programul de mai jos citește două fracții (`a/b` și `c/d`) și afișează suma și produsul, simplificate:

```python
a, b = map(int, input().split())     # prima fracție: a/b
c, d = map(int, input().split())     # a doua fracție: c/d

# suma: a/b + c/d = (a·d + b·c) / (b·d)
x, y = a * d + b * c, b * d
p, q = abs(x), abs(y)                # Euclid pe valori pozitive
while q != 0:
    p, q = q, p % q
print(f"suma = {x // p}/{y // p}")

# produsul: (a·c) / (b·d)
x, y = a * c, b * d
p, q = abs(x), abs(y)
while q != 0:
    p, q = q, p % q
print(f"produs = {x // p}/{y // p}")
```

intrare	ieșire
`3 4` și `5 6`	`suma = 19/12`, `produs = 5/8`
`2 3` și `9 4`	`suma = 35/12`, `produs = 3/2`
`1 2` și `-1 2`	`suma = 0/1`, `produs = -1/4`

Diferența și câtul se fac la fel: `(a·d − b·c)/(b·d)` și `(a·d)/(b·c)`, cu condiția ca `c ≠ 0`. Ai observat că bucla lui Euclid apare de două ori, identic? În lecția py9-08 o vei scrie **o singură dată**, ca funcție, și o vei apela de câte ori ai nevoie.

7. Cum testezi un program cu cmmdc

tip de test	exemplu	rezultat așteptat
caz obișnuit	`48 18`	`6`
numere egale	`7 7`	`7`
unul îl divide pe celălalt	`5 35`	`5`
prime între ele	`8 15`	`1`
ordinea inversă	`18 48`	`6`
unul este 1	`1 999`	`1`
unul este 0	`12 0`	`12` la împărțiri; buclă infinită la scăderi
numere mari	`1000000 999999`	`1`, instantaneu la împărțiri

8. Verifică-te

1. Urmărește Euclid prin împărțiri pentru `84 36`. Câți pași face și ce afișează?
2. Câte scăderi face varianta cu scăderi pentru `100 3`?
3. Știind că `cmmdc(a, b) = 4` și `a · b = 480`, cât este `cmmmc(a, b)`?

4. De ce este greșit codul următor? Ce afișează pentru `48 18`?

```
while b != 0:
    a = b
    b = a % b
print(a)
```

5. Când este fracția `n / (n + 1)` ireductibilă?
6. Câte plăci pătrate, cât mai mari, acoperă o terasă de 420 cm × 300 cm?

Răspunsuri

1. `(84, 36) → (36, 12) → (12, 0)`: 2 pași, afișează **12**.
2. **35** de scăderi: 33 de scăderi îl aduc pe 100 la 1, apoi `3 − 1 = 2` și `2 − 1 = 1`; cmmdc este 1.
3. `480 / 4 = 120`.
4. După `a = b`, vechiul `a` este pierdut, iar `b = a % b` devine `b % b = 0`. Bucla se oprește după primul pas și afișează `18` (greșit; corect este 6).
5. Mereu (pentru `n ≥ 1`): orice divizor comun al lui `n` și `n + 1` divide și diferența lor, care este 1.
6. `cmmdc(420, 300) = 60`, deci plăci de 60 cm: `7 · 5 = 35` de plăci.

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] **#58** **cmmdc** — ușoară
  https://www.pbinfo.ro/probleme/58/cmmdc
- [ ] **#59** **cmmmc** — ușoară
  https://www.pbinfo.ro/probleme/59/cmmmc
- [ ] **#391** **fractii** — ușoară
  https://www.pbinfo.ro/probleme/391/fractii

Temă pentru acasă (pbinfo)

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

- [ ] **#60** **primeintreele** — ușoară
  https://www.pbinfo.ro/probleme/60/primeintreele
- [ ] **#61** **afisaredivizoricomuni** — ușoară
  https://www.pbinfo.ro/probleme/61/afisaredivizoricomuni