INFORMATICĂ · PYTHON · CLASA A IX-A · PRELUCRĂRI ALE DATELOR NUMERICE · LECȚIA 5

Divizori, multipli, numere prime, factori primi

Durată: 100 min Nivel: mate-info · Python Înainte: lecțiile py9-03, py9-04

CE VEI ȘTI DUPĂ LECȚIE

- Verifici divizibilitatea și generezi multiplii unui număr cu %, // și range() cu pas.
- Parcurgi divizorii unui număr până la n și până la √n și explici, cu număr de pași, de ce a doua variantă este mai eficientă.
- Scrii un test de primalitate corect pentru toate cazurile limită (0, 1, 2, pătrate perfecte).
- Descompui un număr în factori primi și explici de ce algoritmul găsește doar factori primi.

1. Divizori și multipli

`d` este divizor al lui `n` exact când `n % d == 0`. Multiplii lui `k` sunt `k, 2k, 3k, …`, deci îi generezi direct cu `range` cu pas, fără să testezi fiecare număr.

```
n, k = 50, 7
print(n % 5 == 0)                  # True: 5 divide 50
print(list(range(k, n + 1, k)))    # [7, 14, 21, 28, 35, 42, 49]  (list(...) strânge valorile într-o listă; vezi py9-14)
a = 100
print((a + k - 1) // k * k)        # 105: primul multiplu al lui 7 ≥ 100
```

Context real. O bandă de 36 cm trebuie tăiată în bucăți egale, de lungimi naturale, fără să rămână rest. Lungimile posibile sunt exact **divizorii lui 36**: 1, 2, 3, 4, 6, 9, 12, 18, 36.

2. Parcurgerea divizorilor: varianta directă

Încerci fiecare `d` de la 1 la `n`:

```
n = int(input())
for d in range(1, n + 1):
    if n % d == 0:
        print(d, end=" ")          # 36 → 1 2 3 4 6 9 12 18 36
```

Corect, dar face `n` **pași**: O(n). O îmbunătățire simplă: niciun divizor (în afară de `n`) nu este mai mare decât `n / 2`, deci poți merge până la `n // 2` și să-l adaugi pe `n` separat. Sunt `n / 2` pași, dar ordinul rămâne O(n).

3. Parcurgerea până la √n

Divizorii vin **în perechi**: dacă `d` divide `n`, atunci și `n // d` divide `n`. Pentru 36: (1, 36), (2, 18), (3, 12), (4, 9), (6, 6). Din fiecare pereche, cel mai mic este cel mult `√n`. Deci e suficient să cauți `d` cu `d * d <= n` și să iei, de fiecare dată, ambii divizori.

```python
n = int(input())
d = 1
while d * d <= n:
    if n % d == 0:
        print(d, end=" ")
        if d != n // d:            # la pătrate perfecte, √n apare o singură dată
            print(n // d, end=" ")
    d += 1
# 36 → 1 36 2 18 3 12 4 9 6
```

ATENȚIE — GREȘELI FRECVENTE

Fără `if d != n // d`, la un pătrat perfect divizorul `√n` apare de două ori (pentru 36, `6` ar apărea dublat). Testează **întotdeauna** și un pătrat perfect. Observă și că divizorii nu mai ies în ordine crescătoare.

Condiția `d * d <= n` este mai sigură decât `d <= math.sqrt(n)`, care lucrează cu numere reale, iar acestea au precizie limitată.

4. Cât de mult contează? Comparația din programă

Am numărat pașii pentru cele trei variante și am măsurat timpul pentru `n = 10⁷` (pe calculatorul de test; la tine timpii vor fi alții, dar raportul va fi asemănător).

Variantă	Pași pentru n = 10^6	Pași pentru n = 10^7	Timp, n = 10^7	Ordin
`d` de la 1 la `n`	1 000 000	10 000 000	≈ 0,9 s	O(n)
`d` de la 1 la `n // 2`	500 000	5 000 000	≈ 0,6 s	O(n)
`d` cu `d * d <= n`	1 000	3 162	sub 0,01 s	O(√n)

Toate trei găsesc aceiași 64 de divizori pentru `10⁷`. Pentru `n = 10¹²`, prima variantă ar avea nevoie de peste o zi, iar a treia face 10^6 pași (≈ 0,2 s).

LA BAC

La Bac, „eficient din punctul de vedere al timpului de executare” la probleme cu divizori înseamnă, de obicei, parcurgerea până la √n. Explicația cerută: divizorii vin în perechi `(d, n / d)`, iar unul dintre ei este cel mult √n.

5. Test de primalitate

Un număr este **prim** dacă are exact doi divizori: 1 și el însuși. Deci 0 și 1 **nu** sunt prime, iar 2 este singurul prim par. Cauți un divizor între 2 și √n; dacă găsești unul, te oprești.

```python
n = int(input())
prim = n >= 2                       # 0 și 1 nu sunt prime
d = 2
while d * d <= n and prim:
    if n % d == 0:
        prim = False                # am găsit un divizor: ne oprim
    d += 1
if prim:
    print("prim")
else:
    print("nu e prim")
```

Test	0	1	2	4	9	97	1000000007
Ieșire	nu e prim	nu e prim	prim	nu e prim	nu e prim	prim	prim

`9` și `4` sunt pătrate de numere prime: dacă ai scrie `d * d < n`, le-ai declara greșit prime. Pentru `1000000007` bucla face aproximativ 31 600 de pași și se termină într-o fracțiune de secundă.

6. Descompunerea în factori primi

Împarți `n` la `d = 2` cât timp se poate și numeri de câte ori (exponentul), apoi treci la `d = 3`, `4`, ... Te oprești când `d * d > n`: dacă din `n` a rămas ceva mai mare decât 1, acel rest este și el un factor prim.

```python
n = int(input())
d = 2
while d * d <= n:
    e = 0
    while n % d == 0:
        n //= d
        e += 1
    if e > 0:
        print(d, e)                 # factorul și exponentul
    d += 1
if n > 1:
    print(n, 1)                     # ultimul factor prim, mai mare decât √n
```

Pentru `1800` afișează `2 3`, `3 2`, `5 2`, adică `1800 = 2³ · 3² · 5²`. Pentru `97` afișează `97 1`, iar pentru `1000000014` afișează pe rând `2 1`, `3 1`, `13 1`, `103 1`, `124471 1`.

De ce apar doar factori primi, deși `d` trece și prin 4, 6, 9? Când `d` ajunge la 4, toți factorii 2 au fost deja scoși din `n`, deci 4 nu mai poate divide `n`. La fel pentru orice `d` compus: factorii lui primi, mai mici, au fost eliminați înainte.

> **SFAT**
>
> Din descompunere afli direct **numărul de divizori**: dacă `n = p₁^e₁ · p₂^e₂ · …`, atunci are `(e₁ + 1)(e₂ + 1)…` divizori. Pentru `1800`: `4 · 3 · 3 = 36`.

7. Verifică-te

1. Care sunt toți multiplii lui 12 mai mici decât 70? Scrie instrucțiunea `range` care îi produce.
2. Câți pași face bucla `while d * d <= n` pentru `n = 50`? Ce divizori afișează?
3. De ce, la parcurgerea până la √n, este nevoie de `if d != n // d`? Dă un exemplu de test care prinde greșeala.
4. Un elev scrie testul de primalitate cu `d` de la 2 la `n - 1`. Este corect? Este eficient?
5. Ce afișează programul de descompunere pentru `n = 84`?
6. Câți divizori are $2^5 \cdot 7^2$?

Răspunsuri

1. 12, 24, 36, 48, 60: `range(12, 70, 12)`.
2. 7 pași (`d` de la 1 la 7, pentru că $8 \cdot 8 = 64 > 50$); afișează `1 50 2 25 5 10`.
3. Pentru pătratele perfecte, `d = n // d` la `d = √n`, deci același divizor ar fi afișat de două ori. Test: `n = 49` (ar apărea `7 7`).
4. Corect (cu grijă la 0 și 1), dar O(n): pentru `1000000007` ar face un miliard de pași în loc de ~31 600.
5. `2 2`, `3 1`, `7 1` ($84 = 2^2 \cdot 3 \cdot 7$).
6. $(5 + 1)(2 + 1) = 18$.

Probleme de lucru în clasă (pbinfo)

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

- [ ] **#45** **verifprim** — ușoară
  https://www.pbinfo.ro/probleme/45/verifprim
- [ ] **#376** **suma-divizori** — ușoară
  https://www.pbinfo.ro/probleme/376/suma-divizori
- [ ] **#62** **factorizare** — ușoară
  https://www.pbinfo.ro/probleme/62/factorizare

Temă pentru acasă (pbinfo)

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

- [ ] **#64** **numarperfect** — ușoară
  https://www.pbinfo.ro/probleme/64/numarperfect
- [ ] **#375** **urmatorul-prim** — ușoară
  https://www.pbinfo.ro/probleme/375/urmatorul-prim
- [ ] **#1319** **descompunere-factori** — ușoară
  https://www.pbinfo.ro/probleme/1319/descompunere-factori

☐ #388 **numarul-divizorilor-pari** ușoară

https://www.pbinfo.ro/probleme/388/numarul-divizorilor-pari
