FIȘĂ DE LUCRU · PYTHON · CLASA A IX-A · FIȘA 5 · LECȚIA PY9-05

Divizori, multipli, numere prime, factori primi

Durată: 100 min Nivel: mate-info · Python

CE VEI ȘTI DUPĂ LECȚIE

- Urmărești algoritmi de divizibilitate și numeri pașii pe care îi fac.
- Repari greșelile tipice: divizorul √n numărat de două ori, ultimul factor prim uitat, primalitatea suprascrisă, range() care se oprește prea devreme.
- Scrii programe O(√n) cu divizori, numere prime și factori primi, testate și pe cazurile limită.

Ce trebuie să știi (o pagină)

Ce vrei	Cum	Ordin
`d` divide `n`?	`n % d == 0`	O(1)
multiplii lui `k` până la `n`	`range(k, n + 1, k)`	
câți multipli ai lui `k` în `[a, b]`	`b // k - (a - 1) // k`	O(1)
cel mai mic multiplu al lui `k` ≥ `a`	`(a + k - 1) // k * k`	O(1)
cel mai mare multiplu al lui `k` ≤ `a`	`a // k * k`	O(1)
toți divizorii, simplu	`for d in range(1, n + 1)`	O(n)
toți divizorii, eficient	`while d * d <= n`: iei `d` și `n // d`, iar dacă `d == n // d`, doar o dată	O(√n)
`n` este prim?	`n >= 2` și niciun `d` cu `2 ≤ d`, `d * d ≤ n`, care îl divide	O(√n)
factori primi	`d = 2, 3, …` cât timp `d * d <= n`; împarți cât se poate; la final, dacă `n > 1`, e un factor prim	O(√n)
numărul de divizori	`n = p₁^e₁ · p₂^e₂ · …` are `(e₁ + 1)(e₂ + 1)…` divizori	
partea întreagă a lui √n	`math.isqrt(n)` (după `import math`)	

SFAT

Teste obligatorii la orice problemă cu divizori: `1`, un număr prim, un **pătrat perfect** (de exemplu 49) și o putere a unui prim (de exemplu 32).

Nivelul 1 · Prezice

★ citește și prezice

P1. Ce afișează?

```
print(91 % 7 == 0, 91 % 13, list(range(15, 61, 15)))
print((73 + 9) // 10 * 10, 120 // 7 * 7)
```

P2. Ce afișează programul de parcurgere până la √n (lecția, secțiunea 3) pentru `n = 100`? Câți pași face bucla?

P3. Ce afișează pentru `n = 28`? Ce verifică programul?

```
s = 0
for d in range(1, n // 2 + 1):
    if n % d == 0:
        s += d
print(s, s == n)
```

P4. Ce afișează pentru `n = 90`?

```
c = 0
for d in range(1, n + 1):
    if n % d == 0 and d % 2 == 1:
        c += 1
print(c)
```

P5. Fă trasarea programului de descompunere în factori primi (lecția, secțiunea 6) pentru `n = 1372`. Ce afișează? De câte ori se execută bucla exterioară? Ce valori au `d` și `n` la final?

P6. Testul de primalitate de mai jos are o greșeală. Pentru ce numere de la 0 la 100 dă un răspuns greșit?

```
prim = n >= 2
for d in range(2, n // 2):
    if n % d == 0:
        prim = False
```

P7. Câți pași face testul de primalitate din lecție (cu oprire la primul divizor) pentru `n = 1000001`? Dar pentru `n = 1000003`, care este prim? *Indiciu:* `1000001 = 101 · 9901`.

P8. Câți divizori are $600 = 2^3 \cdot 3 \cdot 5^2$? Dar 2^{10}? Ce numere au **exact 3** divizori? Dă primele patru.

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Suma divizorilor, în O(√n). Pentru `12` dă corect `28`, dar pentru `16` dă `35`. De ce? Repară.

```
s = 0
d = 1
while d * d <= n:
    if n % d == 0:
        s += d + n // d
    d += 1
print(s)
```

P10. Programul spune că `9`, `10` și `1` sunt prime. Găsește cele **două** greșeli.

```
n = int(input())
prim = True
for d in range(2, n):
    if n % d == 0:
        prim = False
    else:
        prim = True
print(prim)
```

P11. Descompunerea afișează pentru `8` corect `2 3`, dar pentru `194` doar `2 1`, iar pentru `97` nimic. Ce lipsește?

```
d = 2
while d * d <= n:
    e = 0
    while n % d == 0:
        n //= d
        e += 1
    if e > 0:
        print(d, e)
    d += 1
```

P12. *Rânduri amestecate.* După `n = int(input())`, ordonează rândurile ca programul să afișeze **numărul de divizori** în O(√n). Un rând este în plus. Ce afișează pentru `36`, `13` și `1`?

```
A)  d = 1
B)  while d * d <= n:
C)  if n % d == 0:
D)  c += 2
E)  if d * d == n:
F)  c -= 1
G)  d += 1
H)  c = 0
I)  while d <= n // 2:
J)  print(c)
```

P13. *Traducere din C++.* Bucla `for (d = 2; d * d <= n; d++)` a fost tradusă ca mai jos. Programul spune că `49`, `4` și `15` sunt prime. De ce? Repară-l folosind `math.isqrt`.

```
prim = n >= 2
for d in range(2, int(n ** 0.5)):
    if n % d == 0:
        prim = False
```

P14. Câte numere din `[a, b]` se divid cu 3 **sau** cu 5? Pentru `1…15` programul dă `8`, nu `7`. Ce a numărat de două ori? Repară formula.

```
print(b // 3 - (a - 1) // 3 + b // 5 - (a - 1) // 5)
```

Nivelul 3 · Scrie cod

★★ scrie cod

Toate programele trebuie să fie O(√n) (P18 și P19 fac un test O(√n) pentru fiecare număr verificat).

P15. Ambalarea. `n` produse se pun în **cel puțin două** pachete, toate cu același număr de produse. Care este numărul **minim** de pachete? Afișează `imposibil` dacă nu se poate.

intrare	12	35	13	49	2	1
ieșire	2	5	13	7	2	`imposibil`

P16. Perfect, abundent, deficient. Fie `s` suma divizorilor lui `n`, fără `n`. Numărul este *perfect* dacă `s = n`, *abundent* dacă `s > n` și *deficient* dacă `s < n`.

intrare	6	28	12	8	1	496
ieșire	`perfect`	`perfect`	`abundent`	`deficient`	`deficient`	`perfect`

P17. Cel mai mare factor prim.

intrare	13195	97	1024	600851475143
ieșire	29	97	2	6857

P18. Prime într-un interval. Citește `a` și `b`. Câte numere prime sunt în `[a, b]`?

intrare (`a`, `b`)	`1`, `10`	`10`, `20`	`1`, `100`	`90`, `96`	`2`, `2`
ieșire	4	4	25	0	1

P19. Prime gemene. Afișează prima pereche de numere prime `(p, p + 2)` cu `p ≥ n`.

intrare	1	3	6	12	100
ieșire	`3 5`	`3 5`	`11 13`	`17 19`	`101 103`

P20. Numărul de divizori din factori primi. Descompune `n` și folosește formula `(e₁ + 1)(e₂ + 1)…`. Compară, pe aceleași teste, cu programul de la P12.

intrare	1	12	97	1000000	600851475143
ieșire	1	6	2	49	16

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Problema inversă. Care este cel mai mic număr natural cu exact 6 divizori? Dar cu exact 5? Cu exact 7? Rațonează cu formula numărului de divizori, apoi verifică printr-un program.

P22. Doar impari. După ce verifici dacă `n` este par, poți testa doar `d = 3, 5, 7, …`. Câți pași economisești pentru `n = 1000003`? Se schimbă ordinul de complexitate? De ce?

P23. De ce funcționează? În descompunere, bucla se oprește când `d * d > n`. Explică de ce ce a rămas din `n`, dacă este mai mare decât 1, este sigur un număr **prim**.

P24. Cât costă multe teste? Numără câte numere prime sunt până la `N`, testând fiecare număr cu metoda din lecție. Rulează pentru `N = 10^5` și `N = 10^6` și notează timpii. De câte ori crește timpul când `N` crește de 10 ori? Ce ordin de complexitate are programul întreg?

P25. Mini-proiect: echipele. Cei `n` elevi ai unei clase se împart în echipe egale, fiecare cu **între 3 și 8** membri. Afișează toate variantele, câte una pe rând, ca `număr_echipe membri`, crescător după numărul de membri; dacă nu există, afișează `imposibil`. Construiește tu tabelul de teste: cel puțin un caz cu mai multe variante, unul fără nicio variantă și unul cu o singură variantă. Teste de pornire: `24` → `8 3`, `6 4`, `4 6`, `3 8`; `13` → `imposibil`.

P26. Numere speciale. *a)* Un număr este *emirp* dacă este prim, oglinditul lui este tot prim, dar **diferit** de el (`13` și `31`; `11` nu este emirp). Citește `N` și afișează câte numere emirp sunt în `[1, N]`. *b)* Două numere diferite sunt *amiabile* dacă suma divizorilor proprii (fără numărul însuși) ai fiecăruia este egală cu celălalt număr. Citește `n` și afișează prima pereche `a b` de numere amiabile cu `a ≥ n` și `a < b`.

intrare `N`	**12**	**13**	**100**	**200**	**1000**
ieșire a)	0	1	8	15	36

intrare `n`	**1**	**221**	**1211**	**3000**
ieșire b)	`220 284`	`1184 1210`	`2620 2924`	`5020 5564`

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

---