FIȘĂ DE LUCRU · PYTHON · CLASA A IX-A · FIȘA I1 (INTENSIV) · LECȚIA PY9-I1

Ciurul lui Eratostene și exponențierea rapidă

Durată: 150 min Nivel: intensiv · Python

CE VEI ȘTI DUPĂ LECȚIE

- Urmărești evoluția listei din ciur și a variabilelor din exponențierea rapidă, pas cu pas.
- Repari ciururi și funcții de putere cu greșeli realiste (limite, ordinea instrucțiunilor, împărțire reală).
- Scrii programe care folosesc ciurul construit o singură dată și puterile modulo m în contexte de identificare, criptare și autentificare.
- Argumentezi, cu număr de pași și memorie, când alegi ciurul și când testul de primalitate.

Ce trebuie să știi (o pagină)

Ce	Cum	De reținut
ciurul: reprezentare	`e = [True] * (n + 1)`, `e[0] = e[1] = False`	indicele = numărul, valoarea = „e prim?”
ciurul: eliminare	`while i * i <= n:` · `if e[i]:` · `for j in range(i * i, n + 1, i): e[j] = False`	multiplii sub `i · i` sunt deja tăiați
lista primelor	`[k for k in range(n + 1) if e[k]]`	`sum(e)` = câte prime sunt ≤ `n`
ciur vs. test	ciur: puțin peste `n` pași, `n + 1` valori în memorie	test pentru fiecare `x`: cam `n · √n` pași
exponent par	`a^b = (a²)^(b/2)`	o înmulțire înjumătățește exponentul
varianta iterativă	`if b % 2 == 1: r = r * a` · `a = a * a` · `b = b // 2`	pași = numărul de biți ai lui `b`
înmulțiri în `r`	câți biți de 1 are `b`	`b = 255` → 8, `b = 256` → 1
puteri modulo	`% m` după **fiecare** înmulțire, sau `pow(a, b, m)`	`a ** b % m` construiește întâi numărul uriaș
RSA simplificat	`n = p · q`, `φ = (p − 1)(q − 1)`, `gcd(e, φ) = 1`, `d · e % φ == 1`	criptare `pow(x, e, n)`, decriptare `pow(c, d, n)`, `x < n`
codificare prin prime	fiecărui element un prim; mulțimea = produsul	decodificare: `c % p == 0`; comune = `gcd`

SFAT

Când urmărești exponențierea rapidă, scrie un tabel cu coloanele `b`, `b` în baza 2, `a`, `r`. Ultima cifră binară a lui `b` îți spune dacă `a` intră în rezultat.

Nivelul 1 · Prezice

★ citește și prezice

P1. Urmărește ciurul din lecție pentru `n = 20`. Ce numere se taie **pentru prima dată** la `i = 2` și la `i = 3`? Ce se întâmplă la `i = 4`? Cu ce valoare a lui `i` se termină bucla `while`? Care este lista finală?

P2. Folosind funcția `ciur` din lecție, ce afișează?

```
e = ciur(30)
print(sum(e), e[25], e.index(True), len(e))
```

P3. Varianta de mai jos numără **de câte ori** este tăiat fiecare număr. Ce afișează? De ce `c[10]` nu este 2, deși 10 are doi factori primi?

```
n = 40
e = [True] * (n + 1)
c = [0] * (n + 1)
i = 2
while i * i <= n:
    if e[i]:
        for j in range(i * i, n + 1, i):
            e[j] = False
            c[j] += 1
    i += 1
print(c[10], c[30], c[36])
```

P4. Completează tabelul pentru `putere(2, 10)` (funcția din lecție), cu valorile **de la sfârșitul** fiecărui pas.

pas	b	a	r
1			
2			
3			
4			

P5. Ce afișează? Atenție la cazurile limită.

```
print(pow(2, 10, 1000), pow(3, 0, 5), pow(7, 5, 1), pow(-2, 3, 5))
```

P6. Pentru fiecare `b` din lista `1, 2, 7, 8, 255, 256`, câte pași face bucla din `putere` și de câte ori se execută `r = r * a`? Care dintre 255 și 256 e „mai ieftin” și de ce?

P7. Cu paleta din lecție (roșu 2, galben 3, albastru 5, verde 7, alb 11, negru 13) și funcția `decodifica`, ce afișează `decodifica(1001)`, `decodifica(12)`, `decodifica(1)`? Este 12 un cod corect pentru un tablou?

P8. RSA de jucărie: `p = 5`, `q = 11`, `e = 3`. Cât sunt `n`, `φ` și `d` (cel mai mic)? Ce dă `pow(2, e, n)` și ce dă decriptarea rezultatului? Dar pentru mesajul `x = 54`?

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Ciurul de mai jos se oprește cu eroare pentru `n = 30`. Ce eroare și de ce? După ce o repari, mai rămâne o greșeală, vizibilă la `sum(e)`. Care?

```
def ciur(n):
    e = [True] * n
    i = 2
    while i * i <= n:
        if e[i]:
            for j in range(i * i, n + 1, i):
                e[j] = False
        i += 1
    return e
```

P10. Un elev a scris `while i * i < n:`. Pentru `n = 24` și `n = 26` rezultatul e corect, dar pentru `n = 25` nu. Ce număr apare greșit ca prim? Dă încă un `n ≤ 50` care prinde greșeala și spune regula.

P11. Funcția nu se mai termină pentru `putere(3, 13)`. Găsește cauza (un singur caracter) și explică ce se întâmplă cu `b` și cu `a`.

```
def putere(a, b):
    r = 1
    while b > 0:
        if b % 2 == 1:
            r = r * a
        a = a * a
        b = b / 2
    return r
```

P12. Aici `a = a * a` a fost mutat **înaintea** lui `if`. Ce calculează de fapt funcția? Verifică pe `(3, 13)` și `(2, 10)`.

```
def putere(a, b):
    r = 1
    while b > 0:
        a = a * a
        if b % 2 == 1:
            r = r * a
        b = b // 2
    return r
```

P13. *Rânduri amestecate.* Pune rândurile în ordine (stabilește tu indentarea) ca să obții funcția `putere_mod(a, b, m)`. Un rând este în plus. Ce dă `putere_mod(3, 200, 13)`?

```
A)  a = a * a % m
B)  return r
C)  def putere_mod(a, b, m):
D)  b = b // 2
E)  r = r * a % m
F)  while b > 0:
G)  b = b - 2
H)  r = 1
I)  if b % 2 == 1:
J)  a = a % m
```

P14. Un elev vrea cheia publică `e = 15` pentru `p = 61`, `q = 53`, apoi caută `d` cu `while d * e % phi != 1: d += 1`. Programul nu se oprește. De ce? Cum alegi corect `e` și ce protecție adaugi buclei?

Nivelul 3 · Scrie cod

★★ scrie cod

Fiecare program se scrie cu **funcții** (py9-08), iar ciurul se construiește **o singură dată**.

P15. Maratonul numerelor prime. Fiecare participant primește ca identificator un număr prim: primul înscris 2, al doilea 3 și așa mai departe. Scrie `primele_k(k)` care întoarce lista primelor `k` prime. Nu știi dinainte până unde să cerni: pornește cu o limită mică și dubleaz-o până ajung numerele.

k	ieșire
`1`	`[2]`
`5`	`[2, 3, 5, 7, 11]`
`10`	ultimul identificator `29`
`1000`	ultimul identificator `7919`

P16. Multe întrebări. Se citesc `q` intervale `[a, b]` cu $1 \le a \le b \le 10^6$. Pentru fiecare, afișează câte prime sunt în interval. Construiește o dată ciurul și o listă `c` cu `c[x]` = câte prime sunt $\le$ `x`; răspunsul este `c[b] − c[a − 1]`.

interval	ieșire
`1 10`	`4`
`10 20`	`4`
`14 16`	`0`
`2 2`	`1`
`1 1000000`	`78498`

P17. Ciurul celui mai mic factor. Modifică ciurul: în loc de `True/False`, `f[x]` reține **cel mai mic factor prim** al lui `x` (inițial `f[x] = x`). Apoi `factori(x)` descompune `x` doar prin împărțiri repetate la `f[x]`.

x	factori(x)
`360`	`[2, 2, 2, 3, 3, 5]`

`x`	`factori(x)`
`97`	`[97]`
`1`	`[]`
`999983`	`[999983]`
`1000000`	șase de 2 și șase de 5

P18. Ultimele trei cifre. Scrie `putere_mod` **fără** `pow` și afișează ultimele trei cifre ale lui a^b, mereu pe 3 caractere (`f"{r:03d}"`). Verifică fiecare rezultat și cu `pow(a, b, 1000)`.

`a b`	ieșire
`2 10`	`024`
`7 2026`	`649`
`3 0`	`001`
`12345 6789`	`625`
`10 5`	`000`

P19. Codul de autentificare. Banca și aplicația au secretul `K = 271828` și modulul `M = 1000003` (prim). La fiecare logare banca trimite o provocare `c`, iar aplicația răspunde cu `c` la puterea `K`, modulo `M`, afișat pe 6 cifre. Scrie funcția `cod(c)` și programul care citește `c`. Ce observi la ultimele două teste? Ce valori ale lui `c` nu ar trebui trimise niciodată?

`c`	ieșire
`12345`	`376299`
`1`	`000001`
`999999`	`549097`
`0`	`000000`
`1000003`	`000000`

P20. Expertul de la muzeu. Cu paleta din lecție, scrie `valid(c)`: un cod este corect dacă se obține ca produs de prime **distincte** din paletă (fiecare culoare cel mult o dată). Apoi, pentru două coduri corecte, afișează culorile comune folosind `math.gcd`.

intrare	ieșire
`valid(110)`, `valid(30030)`	`True`, `True`
`valid(12)`, `valid(17)`	`False`, `False`
comune `110 2145`	`['albastru', 'alb']`
comune `6 35`	`[]`
comune `30030 1001`	`['verde', 'alb', 'negru']`

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Problemă inversă. Pentru ce valori ale lui `b` face bucla din `putere` **exact 4 pași**? Dintre ele, pentru care se execută `r = r * a` o singură dată? Dar exact de două ori? Justifică cu scrierea în baza 2.

P22. Cât economisim? Pentru `b = 10⁹`, câte înmulțiri face metoda clasică? Câți pași și câte înmulțiri face `putere_mod`? Măsoară cât durează la tine `10⁷` înmulțiri clasice `r = r * 7 % M` și estimează cât ar dura metoda clasică pentru `10⁹`.

P23. Ciur sau test? Alege și justifică (pași și memorie) pentru: (a) verifici dacă un singur număr de 12 cifre este prim; (b) primești 100 000 de numere ≤ 10^6 și trebuie să spui pentru fiecare dacă e prim; (c) vrei toate primele până la 10^9. *Indiciu:* `sys.getsizeof([True] * 10**6)` arată câți octeți ocupă lista.

P24. RSA pe hârtie. `p = 7`, `q = 11`. Calculează `n` și `φ`, alege **cel mai mic** `e > 1` cu `gcd(e, φ) = 1` și găsește `d`. Criptează `x = 9` și decriptează. Apoi încearcă `x = 80`: ce obții la decriptare și de ce?

P25. Mini-proiect: seiful cu chei. Scrie un program cu trei funcții: `alege_chei(p, q, e)` (verifică cu ciurul că `p` și `q` sunt prime, că `p ≠ q` și că `gcd(e, φ) = 1`, apoi întoarce `(e, d, n)`), `cripteaza(text, e, n)` și `decripteaza(lista, d, n)`. Testează pe un mesaj al tău. Întrebare: de ce cerem `p ≠ q`? Încearcă `p = q = 61`, `e = 17` cu formula `φ = (p − 1)(q − 1)` și vezi ce iese la decriptarea lui `"SALUT"`.

Probleme de lucru în clasă (pbinfo)

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

- [ ] **#303** eratostene — medie
  https://www.pbinfo.ro/probleme/303/eratostene
- [ ] **#3977** expon — ușoară
  https://www.pbinfo.ro/probleme/3977/expon

Temă pentru acasă (pbinfo)

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

- [ ] **#302** cautaprim — medie
  https://www.pbinfo.ro/probleme/302/cautaprim
- [ ] **#510** sortprime — ușoară
  https://www.pbinfo.ro/probleme/510/sortprime