FIȘĂ DE LUCRU · PYTHON · CLASA A IX-A · FIȘA 3 · LECȚIA PY9-03

Testare, depanare și eficiența algoritmilor

Durată: 100 min Nivel: mate-info · Python

CE VEI ȘTI DUPĂ LECȚIE

- Construiești teste care ating fiecare ramură și fiecare graniță a unui program.
- Găsești erori prin trasare și le încadrezi corect: sintaxă, execuție, logică.
- Compari doi algoritmi după numărul de operații și după ordinul de complexitate.

Ce trebuie să știi (o pagină)

Idee	Pe scurt
set de teste	caz obișnuit + fiecare ramură + granițe (și vecinii lor) + extreme + date nevalide
validarea datelor	programul verifică datele înainte să le folosească (`if x < 1 or x > 10: …`)
trasare	tabel cu o coloană pentru fiecare variabilă și pentru condiție, câte un rând pe pas
`print("DEBUG", …)`	afișare temporară a valorilor; o ștergi după depanare
eroare de **sintaxă**	programul nu pornește: `SyntaxError`, `IndentationError`
eroare de **execuție**	se oprește la un rând: `ZeroDivisionError`, `ValueError`, `TypeError`, `NameError`
eroare de **logică**	rulează, dar rezultatul e greșit; fără mesaj
eficiență	timp (număr de operații) și memorie (câte valori ții minte deodată)
O(1)	fără bucle care depind de `n`: o formulă
O(√n)	buclă cu `k * k <= n`
O(n)	o buclă cu `n` pași; bucle **una după alta** se adună
O(n^2)	două bucle **una în alta**, fiecare cu ~`n` pași
estimare	Python face, orientativ, câteva milioane de pași simpli pe secundă

SFAT

Când un test pică, caută **cel mai mic** test care pică și fă trasarea pe el. Un tabel de 4 rânduri se urmărește ușor, unul de 400 nu.

Nivelul 1 · Prezice

★ citește și prezice

P1. Fă tabelul de trasare (`n` , `pasi`) pentru `n = 6` . Ce afișează?

```
n = 6
pasi = 0
while n != 1:
    if n % 2 == 0:
        n //= 2
    else:
        n = 3 * n + 1
    pasi += 1
print(pasi)
```

P2. Ce tip de eroare are fiecare fragment și ce mesaj (sau rezultat) apare?

```
x = int("12")                # a)
y = x / (x - 12)
```

```
print("Media: " + 9.5)       # b)
```

```
for i in range(5)            # c)
    print(i)
```

```
a, b = 8, 6                  # d) media lui a și b
print(a + b / 2)
```

P3. De câte ori se execută `c += 1` în fiecare fragment, pentru `n = 5` ? Ce ordin de complexitate are fiecare?

```
for i in range(n):           # A
    for j in range(n):
        c += 1
```

```
for i in range(n):           # B
    for j in range(i):
        c += 1
```

```
for i in range(n):           # C
    c += 1
for j in range(n):
    c += 1
```

P4. Ce ordin de complexitate (O(1), O(√n), O(n), O(n²)) are fiecare fragment? a) `for i in range(1, n + 1, 2): s += i` · b) `k = 1` și `while k * k <= n: k += 1` · c) `for i in range(100):` cu `for j in range(n): c += 1` în interior · d) `x = n * n * n` · e) `for i in range(n):` cu `for j in range(n, 0, -1): c += 1` în interior.

P5. Ce afișează programul, cu tot cu rândurile de depanare?

```
n = 365
c = 0
while n > 0:
    print("DEBUG", n, c)
    n //= 10
    c += 1
print(c)
```

P6. Sunt echivalente condițiile `10 <= x < 20` și `x // 10 == 1` pentru numere întregi? Verifică pe `-5`, `9`, `10`, `19`, `20`. De ce sunt acestea testele potrivite?

P7. Un program O(n²) rulează 2 secunde pentru `n = 1000`. Cât estimezi pentru `n = 2000`? Dar pentru `n = 10 000`? Același calcul pentru un program O(n) care rulează 2 secunde pentru `n = 10⁶`, dacă `n` se dublează.

P8. Pentru care dintre probleme trebuie să **memorezi** toate cele `n` numere citite? a) suma lor; b) maximul; c) câte sunt mai mari decât media lor; d) afișarea lor în ordine inversă.

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Programul numără câte numere pare sunt în intervalul `[a, b]`. Găsește un test pe care greșește și repară-l.

```
a = int(input())
b = int(input())
c = 0
for x in range(a, b):
    if x % 2 == 0:
        c += 1
print(c)
```

P10. Programul citește note până la `0` și afișează media lor. Pentru `8 10 0` afișează `5.0`, iar pentru `0` se oprește cu eroare. Fă trasarea, găsește **ambele** probleme și repară-le (pentru `0`, afișează `nicio notă`).

```
s = 0
c = 0
x = int(input())
while x != 0:
    x = int(input())
    s += x
    c += 1
print(s / c)
```

P11. Programul are o eroare de **fiecare** tip. În ce ordine le descoperi și cum? Repară-le. Pentru `3`, mesajul corect este `Suma pătratelor este 14`.

```python
n = int(input())
if n < 0
    print("n invalid")
s = 0
for i in range(1, n):
    s += i * i
print("Suma pătratelor este " + s)
```

Indiciu: după validare, calculul ar trebui să se facă doar pentru `n` valid.

P12. Un an este bisect dacă se divide cu 4 și nu cu 100, **sau** dacă se divide cu 400. Programul a trecut testele `2024 → DA`, `2023 → NU`, `1900 → NU`. Ce test lipsește? Repară programul.

```python
an = int(input())
if an % 4 == 0 and an % 100 != 0:
    print("DA")
else:
    print("NU")
```

P13. *Corect, dar lent.* Programul numără multiplii lui 7 din `1…n`. Ce ordin are? Rescrie-l în O(1). Pe care variantă o poți rula pentru `n = 10¹²`?

```python
n = int(input())
c = 0
for i in range(1, n + 1):
    if i % 7 == 0:
        c += 1
print(c)
```

P14. *Rânduri amestecate.* După `n = int(input())`, ordonează rândurile ca programul să afișeze cel mai mic `k` cu `1 + 2 + … + k ≥ n`. Un rând este în plus. Ce afișează pentru `10` și pentru `11`? Ce ordin de complexitate are?

```
A)  k += 1
B)  s += k
C)  while s < n:
D)  s = 0
E)  print(k)
F)  k = 0
G)  while s <= n:
```

Nivelul 3 · Scrie cod

★★ scrie cod

Înainte de cod, adaugă **cel puțin două** teste proprii la fiecare tabel și explică ce verifică.

P15. Nota. Citește o notă. Dacă nu este între 1 și 10, afișează `notă invalidă`; altfel, `promovat` (cel puțin 5) sau `corigent`.

intrare	5	4	10	1	0	11
ieșire	`promovat`	`corigent`	`promovat`	`corigent`	`notă invalidă`	`notă invalidă`

P16. Pașii lui Collatz. Pentru `n ≥ 1`: dacă `n` e par, îl înjumătățești, altfel devine `3n + 1`; te oprești la 1. Afișează numărul de pași. Variantă: afișează și fiecare valoare (trasarea făcută de program).

intrare	1	6	7	27
ieșire	0	8	16	111

P17. Perechi cu sumă pară. Câte perechi `(i, j)` cu `1 ≤ i < j ≤ n` au `i + j` par? Scrie întâi varianta O(n²), cu două bucle, apoi una O(1). Compară-le pe toate testele.

intrare	1	4	5	10	1000000
ieșire	0	2	4	20	249999500000

P18. Multipli într-un interval. Citește `a`, `b`, `k` (`1 ≤ a ≤ b`) și afișează câți multipli ai lui `k` sunt în `[a, b]`, **fără buclă**.

intrare (`a`, `b`, `k`)	ieșire
`1`, `10`, `3`	`3`
`3`, `9`, `3`	`3`
`4`, `5`, `3`	`0`
`7`, `7`, `7`	`1`
`1`, `1000000000`, `7`	`142857142`

P19. Cronometrul. Măsoară cât durează suma `1 + … + n` cu o buclă, pentru `n = 10⁵`, `10⁶`, `10⁷`. Folosește `import time`, apoi `t = time.perf_counter()` înainte de buclă și `time.perf_counter() - t` după ea. Verifică de fiecare dată rezultatul cu formula lui Gauss. Ce observi când `n` crește de 10 ori? Apoi reprezintă cei trei timpi **grafic**, cu bare din `#` în consolă: cea mai lungă bară are 40 de `#`, iar celelalte sunt proporționale cu timpul lor (cel puțin un `#`). Se potrivește graficul cu ce ai spus despre ritmul de creștere?

P20. Pătrate perfecte. Câte pătrate perfecte nenule sunt cel mult egale cu `n`? Folosește o buclă O(√n).

intrare	0	1	15	16	1000000000000
ieșire	0	1	3	4	1000000

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Testul care prinde. Un elev calculează suma lui Gauss cu `int(n * (n + 1) / 2)` în loc de `n * (n + 1) // 2`. Pentru `n = 10⁸` și chiar `n = 10⁹` rezultatele coincid. Găsește cel mai mic `n` pentru care diferă și explică de ce. *Indiciu:* numerele reale (`float`) au 53 de biți pentru cifre.

P22. Doi algoritmi. Vrem să știm dacă `n` este pătrat perfect. A: `for k in range(n + 1)`, cu oprire la `k * k == n`. B: `k = 0`, `while k * k < n: k += 1`, apoi verifici `k * k == n`. Câți pași face fiecare pentru `n = 10⁶ + 1`? Dar pentru `n = 10⁶`? Ce ordin are fiecare?

P23. Jumătate tot pătratic. Arată că în fragmentul de mai jos `c += 1` se execută `n(n + 1) / 2` ori. Dacă e cam jumătate din n^2, de ce spunem tot $O(n^2)$? Ce se întâmplă cu timpul când `n` se dublează?

```
for i in range(1, n + 1):
    for j in range(1, i + 1):
        c += 1
```

P24. Alege algoritmul după limite. Cu aproximativ 10^7 pași pe secundă și o limită de 1 secundă, ce ordine de complexitate (O(1), O(√n), O(n), O(n²)) sunt acceptabile pentru: a) $n \le 1000$; b) $n \le 10^6$; c) $n \le 10^{12}$; d) $n \le 10^{18}$?

P25. Jurnal de depanare. Programul de mai jos numără numerele de 3 cifre care au suma cifrelor egală cu `s`. Completează un jurnal cu rubricile: *simptom*, *cel mai mic test care pică*, *trasare*, *cauză*, *reparație*, *retestare*. Ce ordin de complexitate are programul? (Atenție: ce este „`n`" aici?)

```
s = int(input())
c = 0
for n in range(100, 999):
    if n // 100 + n // 10 % 10 + n % 10 == s:
        c += 1
print(c)
```

Probleme de lucru în clasă (pbinfo)

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

- [ ] `#1360` **suma-gauss** — ușoară
  https://www.pbinfo.ro/probleme/1360/suma-gauss
- [ ] `#1255` **lipsa** — ușoară
  https://www.pbinfo.ro/probleme/1255/lipsa

Temă pentru acasă (pbinfo)

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

- [ ] `#3609` **countmod10** — ușoară
  https://www.pbinfo.ro/probleme/3609/countmod10
- [ ] `#3978` **sabc** — medie
  https://www.pbinfo.ro/probleme/3978/sabc