test-e.ro

CAIET ELEV

Lecția 25

Sortare cu lista de frecvențe. Condiții de aplicabilitate

Când valorile de sortat sunt numere întregi dintr-un interval mic și cunoscut dinainte, nu mai e nevoie să le comparăm între ele: numărăm de câte ori apare fiecare valoare posibilă, într-o listă de frecvențe în care indicele este valoarea, apoi le scriem în ordine, fiecare de câte ori a apărut.

DISCIPLINA Informatică **CLASA** a IX-a · mate-info **LIMBAJ** Python 3

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Rezolvă exercițiile pe hârtie: urmărești execuția pas cu pas, completezi, asociezi și scrii singur soluțiile, apoi le verifici pe platformă. Ce ai voie să folosești: liste, [0] * k, indexare, len, for cu range, while, if/else și metoda append(). Fără def/return, fără min/max/sum, fără .sort()/sorted(), fără atribuiri simultane, fără break sau continue — pentru ca cele două metode de sortare să arate la fel puse alături. Lista de frecvențe se indexează pe intervalul dat în enunț: indicele este chiar valoarea. Fără diacritice în codul Python.

CUPRINS

1 Ce știm deja .....
2 Numărăm, în loc să comparăm .....
3 De la frecvențe la șirul sortat .....
4 Când se poate și când nu .....
5 Cât costă fiecare metodă .....
6 Ce mai putem citi din lista de frecvențe .....
7 Cod ghidat · cinci probleme .....

SECȚIUNEA 1 · CE ȘTIM DEJA

1 Ce știm *deja*

Lecția trecută am sortat comparând: căutam de fiecare dată cel mai mic element rămas și îl aduceam la locul lui. Astăzi schimbăm complet unghiul — dar avem nevoie de trei unelte pe care le știm deja: o listă de contoare pornită de la zero, accesul direct la o poziție și numărarea aparițiilor. Le reluăm aici, pe scurt.

O listă se poate construi de la zero cu o valoare repetată: `[0] * 11` creează o listă cu **unsprezece** elemente, toate egale cu 0. Numerotarea pozițiilor începe de la 0, deci pozițiile sunt `0, 1, ..., 10`. Accesul la o poziție este direct: `f[7]` ajunge la al optulea element fără să treacă prin celelalte, oricât de lungă ar fi lista.

Numărarea aparițiilor unei valori o știm de la L15: parcurgem lista, comparăm fiecare element cu valoarea căutată și creștem un contor.

Aceeași idee în Python (cu sintaxă reală):

```
c = 0
for i in range(n):
    if v[i] == x:
        c = c + 1
print(c)
```

De la lecția trecută păstrăm și **prețul** sortării prin selecția minimului: pentru o listă cu `n` elemente se fac `n - 1` comparări la primul pas, `n - 2` la al doilea și așa mai departe, în total `n(n-1)/2`. Pentru `n = 5` sunt 10 comparări; pentru `n = 100`, deja 4950. Numărul nu depinde de datele din listă: și o listă deja sortată costă la fel.

> **DE REȚINUT**
> Într-o listă obișnuită, indicele e o *poziție*, iar conținutul e o *valoare*: `v[2] = 9` înseamnă „al treilea element este 9”. Reține exact această citire — lecția de astăzi o va răsturna.

A EXERCIȚIUL A · GRILĂ

Uneltele de care avem nevoie

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Ce construiește instrucțiunea `f = [0] * 11`?

- O listă cu 11 elemente, toate egale cu 0
- O listă cu numerele de la 0 la 11, așezate în ordine crescătoare
- O listă cu 10 elemente egale cu 0, plus un ultim element egal cu 11
- Un singur număr, 0, repetat de 11 ori în interiorul unui text

2. Câte comparări între elemente face sortarea prin selecția minimului pe o listă cu 5 elemente?

- 25 de comparări, adică 5 × 5, câte una pentru fiecare pereche posibilă
- 5 comparări, câte una pentru fiecare element al listei
- 4 comparări, cu una mai puțin decât numărul de elemente
- 10 comparări, adică 4 + 3 + 2 + 1

3. Ce se afișează?

- 4, pentru că bucla parcurge toate cele patru poziții ale listei
- 2, pentru că valoarea 3 apare pe pozițiile 0 și 2
- 0, pentru că variabila c nu se modifică niciodată în interiorul buclei
- 3, pentru că variabila c ajunge la valoarea celui mai mare element

4. Ce înseamnă că o listă este sortată crescător?

- Primul element este cel mai mic, iar restul pot fi în orice ordine
- Niciun element nu este mai mare decât cel care îl urmează
- Toate elementele sunt diferite între ele și așezate de la cel mai mic la cel mai mare
- Suma elementelor de pe pozițiile pare este mai mică decât suma celor de pe pozițiile impare

B

EXERCIȚIUL B · DRY-RUN

Unde se ascunde minimul

DRY-RUN

Urmărește căutarea poziției minimului, așa cum am făcut-o lecția trecută. Coloanele i și v[i] sunt date; tu completezi coloana cu răspunsul la comparație (scrie DA sau NU) și valoarea lui p după fiecare pas.

```
v = [7, 2, 9, 2, 5]
p = 0
for i in range(1, 5):
    if v[i] < v[p]:
        p = i
```

```
6  print(p, v[p])
```

i (dat)	v[i] (dat)	v[i] < v[p] ? (completează, text)	p după pas (completează, numeric)
–	–	–	0
1	2		
2	9		
3	2		
4	5		
5 – bucla se oprește	–	–	1

HINT *La fiecare pas compari v[i] cu v[p], adică cu minimul găsit până acum, și schimbi p doar dacă v[i] este STRICT mai mic.*

SPRIJIN · SCRIE CODUL

Câte apariții are valoarea x?

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând, iar la final un număr întreg x. Afișează câte dintre cele n numere citite sunt egale cu x. Ex.: pentru n = 5, numerele 3 7 3 1 3 și x = 3: valoarea 3 apare pe pozițiile 0, 2 și 4, deci se afișează 3.

HINT *Dacă valoarea x nu apare deloc, răspunsul este 0 — contorul rămâne neatins.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Valoarea repetată

SCRIERE COD

Se citesc două numere naturale, v și c, cu c >= 1. Afișează pe un singur rând valoarea v repetată de c ori, cu câte un spațiu între repetări. Ex.: pentru v = 7 și c = 3 se afișează 7 7 7.

HINT Atenție. *Nu construi un text pe care să-l afișezi la final; folosește print(v, end=" ") în buclă și un print() gol după buclă, care încheie rândul.*

HINT Curiozitate. *Exact asta face un program care desenează o histogramă: pentru fiecare valoare afișează un simbol de atâtea ori cât arată numărul ei de apariții.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · NUMĂRĂM, ÎN LOC SĂ COMPARĂM

2 Numărăm, în loc să *comparăm*

Dacă valorile care pot apărea sunt puține și le cunoaștem dinainte — notele de la 1 la 10, codurile de triaj de la 1 la 5, numărul de frați al fiecărui elev — nu mai avem nevoie să comparăm datele între ele. E de ajuns să ținem, pentru fiecare valoare posibilă, un contor cu numărul ei de apariții. Lista acestor contoare se numește listă de frecvențe.

Pornim de la notele unei clase: `8 4 8 10 4 8`. Vrem să știm câte note de fiecare fel sunt. Am putea scrie zece instrucțiuni `if`, câte una pentru fiecare notă posibilă — dar există o soluție mult mai scurtă: construim o listă `f` cu 11 poziții, toate 0, și pentru fiecare notă citită creștem cu 1 contorul de la poziția cu indicele egal cu nota.

Aceeași idee în Python (cu sintaxă reală):

```
1  f = [0] * 11
2  for i in range(n):
3      x = int(input())
4      f[x] = f[x] + 1
```

După toate cele șase note, lista arată așa: `f[4] = 2`, `f[8] = 3`, `f[10] = 1`, iar restul pozițiilor au rămas 0. Poziția `f[0]` nu se folosește niciodată, pentru că nu există nota 0 — o lăsăm acolo tocmai ca indicele să fie **chiar** nota, fără nicio corecție de calcul.

LISTA DE FRECVENȚE

Este o listă în care **indicele reprezintă valoarea**, iar **conținutul de pe acea poziție arată de câte ori apare valoarea** în șirul de date. Este exact inversul citirii obișnuite: la o listă de date, `v[2] = 9` înseamnă „al treilea element este 9”; la o listă de frecvențe, `f[2] = 9` înseamnă „valoarea 2 apare de nouă ori”.

ATENȚIE

Lungimea listei de frecvențe se stabilește după **valoarea maximă posibilă**, nu după câte date sunt. Pentru note de la 1 la 10 avem nevoie de `[0] * 11`, pentru că trebuie să existe și poziția 10 — iar numerotarea începe de la 0. Cu `[0] * 10` ultima poziție ar fi 9, iar prima notă de 10 ar opri programul cu eroare.

Lista de frecvențe **pierde** ordinea în care au venit datele: din `f` nu mai poți afla că prima notă a fost 8. Păstrează însă tot ce contează pentru sortare — care valori apar și de câte ori fiecare.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

> Diagrama cu bare din raportul dirigintelui („câți elevi au luat fiecare notă”) este chiar o listă de frecvențe desenată. Aplicația care numără câte mesaje ai primit în fiecare oră a zilei ține un contor pentru fiecare dintre cele 24 de ore. Iar un aparat de vot electronic nu compară buletinele între ele: ține câte un contor pentru fiecare candidat și adună 1 la fiecare vot.

A

EXERCIȚIUL A · DRY-RUN

Cum se umple lista de frecvențe

DRY-RUN

Urmărește construirea listei de frecvențe pentru șase note. Coloanele i și x sunt date; tu completezi, cu numere, valoarea contorului f[x] înainte și după creșterea de la pasul respectiv. Primul rând este completat ca exemplu.

```
f = [0] * 11
note = [8, 4, 8, 10, 4, 8]
for i in range(6):
    x = note[i]
    f[x] = f[x] + 1
print(f[4], f[8], f[10])
```

i (dat)	x = note[i] (dat)	f[x] înainte (completează, numeric)	f[x] după (completează, numeric)
0	8	0	1
1	4		
2	8		
3	10		
4	4		
5	8		
6 — bucla se oprește	—	—	—

HINT *La fiecare pas te uiți DOAR la contorul de la poziția x: valoarea lui de dinainte și aceeași valoare plus 1.*

B

EXERCIȚIUL B · SCRIE CODUL

Câte note de fiecare fel

SCRIERE COD

Se citește un număr natural n, apoi n note, numere întregi din intervalul 1..10, fiecare de pe câte un rând. Afișează pe un singur rând, separate prin câte un spațiu, zece numere: câte note de 1 sunt, câte de 2, și așa mai departe până la câte note de 10 sunt. Ex.: pentru n = 6 și notele 8 4 8 10 4 8 se afișează 0 0 0 2 0 0 0 3 0 1: două note de 4, trei de 8, una de 10, restul zero.

HINT *Se afișează zece numere, chiar dacă multe sunt 0 — un contor rămas la zero este o informație, nu o lipsă.*

HINT **Atenție.** *Lista de frecvențe are 11 poziții ([0] * 11), dar afișarea începe de la poziția 1: poziția 0 nu corespunde niciunei note.*

SOLUȚIA TA · PYTHON

C

EXERCIȚIUL C · COMPLETARE

Definiția listei de frecvențe

COMPLETARE

Completează definiția folosind cuvintele din rezervă (două cuvinte rămân nefolosite).

`conținutul` `media` `ordonate` `apare` `indicele` `întregi`

Lista de frecvențe este o listă în care _____ reprezintă valoarea, iar _____ de pe acea poziție arată de câte ori _____ valoarea în șirul de date. Metoda se poate aplica doar dacă valorile sunt _____ și se află într-un interval mic.

SECȚIUNEA 3 · DE LA FRECVENȚE LA ȘIRUL SORTAT

3 De la frecvențe la șirul *sortat*

Lista de frecvențe conține deja toată informația de care avem nevoie: care valori apar și de câte ori fiecare. Ca să obținem șirul sortat, nu ne mai rămâne decât s-o citim în ordine, de la cea mai mică valoare posibilă către cea mai mare, și să scriem fiecare valoare de câte ori arată contorul ei. Nicio comparare între date.

După ce lista de frecvențe e construită, sortarea devine o simplă parcurgere. Mergem cu `nota` de la cea mai mică valoare posibilă la cea mai mare și, pentru fiecare, afișăm valoarea de `f[nota]` ori. Dacă `f[nota]` este 0, bucla interioară nu rulează deloc și valoarea e sărită — exact ce vrem.

Aceeași idee în Python (cu sintaxă reală):

```
for nota in range(1, 11):
    for j in range(f[nota]):
        print(nota, end=" ")
print()
```

SORTAREA CU LISTĂ DE FRECVENȚE

Are exact doi pași. **Pasul 1 — contorizarea:** parcurgi datele o singură dată și, pentru fiecare valoare `x`, faci `f[x] = f[x] + 1`. **Pasul 2 — reconstrucția:** parcurgi lista de frecvențe de la prima valoare posibilă la ultima și afișezi fiecare valoare de atâtea ori cât arată contorul ei. Datele inițiale nu se mai compară niciodată între ele, nu se mută și nu se interschimbă.

Pentru `8 4 8 10 4 8`: la pasul 2, `nota = 4` are `f[4] = 2` și afișează `4 4`; `nota = 8` are `f[8] = 3` și afișează `8 8 8`; `nota = 10` afișează `10`. Rezultat: `4 4 8 8 8 10`.

Cât costă? Pasul 1 face `n` operații, câte una pentru fiecare dată citită. Pasul 2 se uită la fiecare dintre cele `k + 1` poziții ale listei de frecvențe și afișează, în total, tot `n` valori. Numărul de pași este deci aproximativ `n + k + 1`, adică **O(n + k)** — unde `n` este numărul de date, iar `k` cea mai mare valoare posibilă. Afișările nu se numără: la fiecare metodă numărăm doar operația ei esențială, exact cum la selecția minimului am numărat doar comparările, nu și interschimbările. Selecția minimului, în schimb, făcea `n(n-1)/2` comparări, adică $O(n^2)$.

ATENȚIE

Parcurgerea de la pasul 2 trebuie să ajungă la **ultima** valoare posibilă, inclusiv. `range(1, 11)` merge de la 1 la 10; `range(1, 10)` s-ar opri la 9 și toate notele de 10 ar dispărea din rezultat, fără niciun mesaj de eroare. Programul ar părea că merge — pe testele fără note de 10.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Așa își sortează un sistem de triaj pacienții din camera de gardă: codurile de urgență sunt doar cinci, iar pacienții pot fi sute. Așa afișează o aplicație de sondaje rezultatele: numără voturile pentru fiecare opțiune, apoi le scrie în ordine. Și tot așa își aranjează un magazin online produsele după numărul de stele: notele posibile sunt cinci, produsele — mii.

A

EXERCIȚIUL A · ORDONARE

Pașii metodei

ORDONARE

Numerotează pașii de la 1 la 5, în ordinea în care se execută sortarea cu listă de frecvențe. Pașii îți apar amestecați.

- [ ] Parcurgem lista de frecvențe de la cea mai mică valoare posibilă către cea mai mare.
- [ ] Stabilim intervalul valorilor posibile și creăm lista de frecvențe, cu toate pozițiile egale cu 0.
- [ ] Afișăm fiecare valoare de atâtea ori cât arată contorul ei, sărind peste cele cu contorul 0.
- [ ] Citim pe rând, o singură dată, valorile din șirul de date.
- [ ] Pentru fiecare valoare citită, adunăm 1 la poziția al cărei indice este chiar valoarea.

B

EXERCIȚIUL B · SCRIE CODUL

Notele clasei, sortate

SCRIERE COD

Se citește un număr natural n, apoi n note, numere întregi din intervalul 1..10, fiecare de pe câte un rând. Afișează notele sortate crescător, pe un singur rând, separate prin câte un spațiu, folosind metoda cu listă de frecvențe. Ex.: pentru n = 6 și notele 8 4 8 10 4 8, lista de frecvențe are f[4] = 2, f[8] = 3, f[10] = 1, iar afișarea dă 4 4 8 8 8 10.

HINT *Notele care apar de mai multe ori trebuie afișate de tot atâtea ori — sortarea nu elimină duplicatele.*

HINT **Atenție.** *Parcurgerea finală merge până la 10 inclusiv, deci range(1, 11). Testul cu o singură notă de 10 verifică exact asta.*

SOLUȚIA TA · PYTHON

PERFORMANȚĂ · SCRIE CODUL

A k-a notă din clasament

SCRIERE COD

Se citește un număr natural n, apoi n note, numere întregi din intervalul 1..10, fiecare de pe câte un rând, iar la final un număr natural k, cu 1 <= k <= n. Afișează a k-a notă din șirul sortat crescător, fără a construi și fără a afișa șirul sortat: coboară prin lista de frecvențe scăzând din k numărul de apariții al fiecărei note, până când k nu mai depășește contorul notei curente. Ex.: pentru n = 6, notele 8 4 8 10 4 8 și k = 3: f[4] = 2, iar 3 > 2, deci scădem și rămâne k = 1, trecem la nota 5; notele 5, 6, 7 au contorul 0 și 1 > 0, deci le sărim fără să scădem nimic; la nota 8, f[8] = 3 și 1 <= 3, deci răspunsul este 8.

HINT *Aici inversăm algoritmul: în loc să scriem tot șirul sortat și să ne uităm la poziția k, aflăm direct valoarea de pe acea poziție. Numărul de pași nu mai depinde deloc de n.*

HINT **Atenție.** *Condiția buclei este k > f[nota], cu > strict. Cu >= s-ar sări peste ultima notă a fiecărei grupe și răspunsul ar fi cu o valoare prea mare.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 4 · CÂND SE POATE ȘI CÂND NU

4 Când se poate și când *nu*

Sortarea cu listă de frecvențe nu este „metoda cea mai bună”. Este o metodă cu condiții de intrare foarte precise: dacă sunt îndeplinite, câștigă detașat; dacă nu, e inaplicabilă sau risipitoare. Aici învățăm să recunoaștem, înainte de a scrie o linie de cod, dacă datele noastre se pretează la ea.

Metoda cere trei lucruri, toate obligatorii.

Prima condiție: valorile trebuie să fie numere întregi. Indicele unei liste este obligatoriu întreg — nu există `f[12.50]`. Un șir de prețuri ca `12.50, 8.75, 30.20` nu poate fi contorizat direct: nu există poziție cu numărul 12,50.

A doua condiție: intervalul valorilor trebuie cunoscut dinainte. Trebuie să știm, înainte de prima citire, cât de lungă facem lista de frecvențe. Pentru note știm: `1..10`. Pentru codurile de triaj știm: `1..5`. Pentru „un număr întreg oarecare” nu știm nimic, deci nu putem porni.

A treia condiție: intervalul trebuie să fie mic. Lista de frecvențe ocupă `k + 1` poziții **indiferent câte date avem**. Pentru 8 valori dintr-un interval `0..1 000 000` metoda dă rezultatul corect, dar rezervă un milion de poziții pentru opt numere și le parcurge pe toate — un cost absurd față de cele 28 de comparări ale selecției minimului.

> **ATENȚIE**
>
> Costul metodei nu depinde doar de câte date ai, ci și de **cât de mare e intervalul**. `n + k + 1` are doi termeni: dacă `k` e uriaș, îl domină pe `n` și metoda pierde, chiar dacă datele sunt puține și frumoase.

Există și o situație intermediară: valorile sunt întregi și apropiate între ele, dar nu încep de la 0 — de exemplu înălțimile elevilor, între 100 și 200 cm. Aici avem două ieșiri. Cea simplă, pe care o folosim în lecție: facem lista cu `[0] * 201` și indexăm direct cu înălțimea, lăsând nefolosite primele 100 de poziții. Cea economică, pe care doar o menționăm: păstrăm doar cele 101 poziții utile și indexăm cu `x - 100`, adică deplasat. A doua variantă economisește memorie, dar adaugă o scădere la fiecare acces — o vom relua când vom avea nevoie de ea.

DE REȚINUT

Întrebarea corectă nu este „care metodă e mai rapidă?”, ci „ce fel de date am?”. Puține valori distincte și multe date ⇒ lista de frecvențe. Valori neîntregi sau interval necunoscut/uriaș ⇒ o metodă bazată pe comparații, cum e selecția minimului.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Un magazin cu 50 000 de comenzi și cinci niveluri de prioritate le sortează instantaneu cu listă de frecvențe. Aceleași 50 000 de comenzi sortate după valoarea în lei, cu bani, nu se pot sorta așa — valorile nu sunt întregi și pot fi oricât de multe. Iar un clasament de olimpiadă cu punctaje întregi de la 0 la 100 e chiar cazul ideal: mulți concurenți, puține punctaje posibile.

A

EXERCIȚIUL A · ASOCIERE

Ce metodă se potrivește?

ASOCIERE

Fiecărui set de date din stânga îi corespunde metoda potrivită și motivul ei. Scrie în căsuță litera potrivită. Fiecare literă se folosește o singură dată.

☐	Notele unei clase: 30 de valori întregi, între 1 și 10
☐	Prețurile a 30 de produse, exprimate în lei și bani
☐	Codurile numerice personale a 20 de elevi, de câte 13 cifre
☐	Codurile de triaj a 500 de pacienți, între 1 și 5

A `Selecția minimului — intervalul de valori este uriaș`

B `Lista de frecvențe — interval mic, cunoscut dinainte (1..10)`

C `Lista de frecvențe — foarte multe date (500) față de doar 5 valori posibile`

D `Selecția minimului — valorile nu sunt numere întregi`

B

EXERCIȚIUL B · SCRIE CODUL

Se poate aplica metoda?

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Afișează pe același rând numărul n, un spațiu, apoi DA dacă toate cele n valori se află în intervalul 0..100 (deci se poate construi direct o listă de frecvențe cu 101 poziții) sau NU în caz contrar. Ex.: pentru n = 4 și numerele 3 -1 5 8 se afișează 4 NU, pentru că valoarea -1 nu poate fi folosită ca indice într-o listă de frecvențe care începe de la 0.

HINT *Limitele intervalului fac parte din el: 0 și 100 sunt acceptate, 101 nu.*

HINT **Atenție.** *Citește toate cele n valori, chiar dacă ai găsit deja una în afara intervalului: dacă te oprești mai devreme, rămân numere necitite, iar programul nu mai poate spune nimic despre ele. Folosește o variabilă-martor (de exemplu* `ok = 1`*), pe care o pui pe 0 la prima valoare nepotrivită și o verifici abia după ce bucla s-a încheiat.*

SOLUȚIA TA · PYTHON

C

EXERCIȚIUL C · GRILĂ

Condițiile de aplicabilitate

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Avem 1000 de note întregi, între 1 și 10. Care metodă face mai puțini pași și de ce?

- Amândouă fac același număr de pași, pentru că prelucrează exact aceleași o mie de valori
- Selecția minimului, pentru că numărul de comparări scade pe măsură ce lista se ordonează
- Lista de frecvențe, pentru că numărul de pași crește proporțional cu n + k, nu cu n²
- Selecția minimului, pentru că nu are nevoie de o listă suplimentară în memorie

2. De ce nu putem sorta cu listă de frecvențe un șir de prețuri ca `12.50`, `8.75`, `30.20`?

- Pentru că lista de frecvențe funcționează doar pentru valori mai mici decât 100
- Pentru că prețurile se compară altfel decât numerele obișnuite, prin rotunjire la lei
- Pentru că numărul de prețuri diferite este întotdeauna prea mare pentru memorie
- Pentru că valorile nu sunt întregi, iar indicele unei liste este obligatoriu întreg

3. Avem 8 valori întregi din intervalul `0..1 000 000`. Ce se întâmplă dacă aplicăm metoda?

- Metoda dă rezultat greșit, pentru că lista de frecvențe nu poate depăși o mie de poziții
- Metoda nu se poate aplica deloc: valorile atât de mari nu pot fi folosite ca indici
- Metoda devine mai rapidă decât selecția minimului, pentru că intervalul este foarte mare
- Metoda dă rezultatul corect, dar risipește: peste un milion de poziții pentru opt valori

SECȚIUNEA 5 · CÂT COSTĂ FIECARE METODĂ

5 Cât costă fiecare *metodă*

„Mai rapid” nu este o impresie, ci un număr care se poate calcula înainte de a rula programul. Numărăm operațiile fiecărei metode pe același set de date și vedem exact de la ce dimensiune încolo se schimbă câștigătorul.

Selecția minimului face, indiferent de date, `n(n-1)/2` comparări: `n - 1` la primul pas, `n - 2` la al doilea, până la 1. Sortarea cu listă de frecvențe face `n` contorizări plus o parcurgere a celor `k + 1` poziții, deci aproximativ `n + k + 1` pași.

Punem cifrele alături, pentru note între 1 și 10 (deci `k = 10`, listă cu 11 poziții):

N (CÂTE DATE)	SELECȚIA MINIMULUI: N(N-1)/2 COMPARĂRI	LISTA DE FRECVENȚE: N + K + 1 PAȘI
10	45	21
100	4 950	111
1 000	499 500	1 011

Diferența nu e „puțin mai bine”: la o mie de note, o metodă face aproape jumătate de milion de operații, iar cealaltă o mie. Motivul e forma formulelor: una crește **pătratic** cu `n`, cealaltă **liniar**.

DE REȚINUT

Un algoritm al cărui număr de pași crește ca n² are următoarea proprietate: dacă dublezi datele, numărul de pași se face de aproximativ patru ori mai mare. La un algoritm de ordin n + k, dacă dublezi datele păstrând același interval, numărul de pași doar se dublează. De aici vine toată diferența din tabel.

Dar formula are doi termeni, iar al doilea nu depinde de `n`. Pentru `n = 5` valori dintr-un interval `0..1000`, selecția minimului face 10 comparări, iar lista de frecvențe 1006 pași — de o sută de ori mai mult. **Câștigătorul se schimbă atunci când `k` devine mare față de `n`.**

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Exact acest calcul îl face un programator înainte de a alege cum sortează datele într-o aplicație: nu încearcă amândouă variantele și cronometrează, ci se uită la câte date sunt și câte valori distincte pot lua. Un joc care afișează clasamentul a 100 000 de jucători după nivel (1–50) alege lista de frecvențe fără să stea pe gânduri.

A

EXERCIȚIUL A · SCRIE CODUL

Care metodă face mai puțini pași

SCRIERE COD

Se citesc două numere naturale: n, numărul de date (n >= 2), și k, cea mai mare valoare posibilă. Considerăm că selecția minimului face n * (n - 1) / 2 comparări, iar metoda cu listă de frecvențe face n + k + 1 pași. Afișează pe același rând numărul de comparări al selecției minimului, un spațiu, numărul de pași al metodei cu listă de frecvențe, un spațiu, apoi cuvântul FRECVENTE dacă a doua metodă face strict mai puțini pași sau SELECTIE în caz contrar. Ex.: pentru n = 10 și k = 10: selecția minimului face 10 * 9 / 2 = 45 comparări, iar lista de frecvențe 10 + 10 + 1 = 21 pași; cum 21 < 45, se afișează 45 21 FRECVENTE.

HINT *La egalitate se afișează SELECTIE, pentru că a doua metodă trebuie să fie STRICT mai bună ca să merite lista suplimentară din memorie.*

HINT **Atenție.** *Împărțirea la 2 se face cu //, nu cu /: rezultatul trebuie să fie un număr întreg, iar / ar afișa 45.0.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 6 · EXTINDERE · CE MAI PUTEM CITI DIN LISTA DE FRECVENȚE

6 Ce mai putem citi din lista de *frecvențe*

Această secțiune depășește conținuturile obligatorii ale clasei a IX-a: o parcurgi ca să mergi mai departe cu aceeași unealtă. Odată construită, lista de frecvențe răspunde la mult mai multe întrebări decât „care e ordinea”. Care valoare apare cel mai des? Câte valori diferite sunt? Care e valoarea din mijloc? Toate se citesc dintr-o singură parcurgere a contoarelor, fără să ne mai atingem de datele inițiale.

Valoarea care apare de cele mai multe ori într-un șir de date se numește **moda** șirului. Într-o listă de frecvențe se găsește imediat: e indicele contorului celui mai mare. Parcurgem contoarele reținând, la fiecare pas, indicele celui mai mare de până acolo — exact ca la căutarea minimului de la L24, doar că acum căutăm un maxim și îl căutăm printre contoare, nu printre date.

Aceeași idee în Python (cu sintaxă reală):

```
best = 1
for nota in range(2, 11):
    if f[nota] > f[best]:
        best = nota
```

ATENȚIE

Comparația se scrie cu `>` strict. Cu `>=`, la două valori cu același număr de apariții s-ar reține ultima întâlnită, nu prima — iar răspunsul s-ar muta pe cea mai mare notă în loc de cea mai mică. Verifică pe `5 5 7 7`: cu `>` iese 5, cu `>=` iese 7.

Tot din lista de frecvențe se citesc și alte valori folosite des în statistică. **Mediana** este valoarea aflată exact la mijlocul șirului sortat: pentru `n` impar, a `(n // 2 + 1)`-a valoare — și o aflăm cu exact tehnica de la exercițiul de performanță, coborând prin contoare. **Numărul de valori distincte** e pur și simplu câte contoare sunt nenule.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**
>
> Raportul „nota cea mai frecventă la simulare” al unei școli e o modă calculată exact așa. Aplicația de sănătate care spune „dormi cel mai des 7 ore” ține un contor pentru fiecare număr întreg de ore. Iar un magazin care afișează „mărimea cea mai cerută: 42” nu compară comenzile între ele — le numără pe mărimi.

A

EXERCIȚIUL A · SCRIE CODUL

Nota care apare cel mai des

SCRIERE COD

Se citește un număr natural n, apoi n note, numere întregi din intervalul 1..10, fiecare de pe câte un rând. Afișează nota care apare de cele mai multe ori. Dacă mai multe note au același număr maxim de apariții, afișează cea mai mică dintre ele. Ex.: pentru n = 6 și notele 8 4 8 10 4 8: nota 8 apare de trei ori, nota 4 de două ori, nota 10 o dată, deci se afișează 8.

HINT *Regula „cea mai mică dintre ele” se obține gratuit dacă parcurgi contoarele în ordine crescătoare și compari cu > strict.*

HINT **Atenție.** *Compari contoare (f[nota] > f[best]), nu note. Confuzia dintre indice și conținut e greșeala numărul unu a acestei lecții.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 7 · COD GHIDAT

7 Scrie cod care *trece testele*

Cinci probleme. Scrie soluția pe hârtie, urmărește-o mental pe exemplul dat și abia apoi tastează-o pe platformă, unde e verificată automat pe 5 cazuri de test.

1

PROBLEMA 7.1 · COD GHIDAT

Triajul pacienților

SCRIERE COD

Se citește un număr natural n, apoi n coduri de urgență, numere întregi din intervalul 1..5, fiecare de pe câte un rând. Afișează codurile sortate descrescător, pe un singur rând, separate prin câte un spațiu. Ex.: pentru n = 6 și codurile 3 1 5 3 2 3 se afișează 5 3 3 3 2 1: mai întâi urgențele de cod 5, apoi cele de cod 3, și așa mai departe.

HINT *Singura schimbare față de sortarea crescătoare este sensul parcurgerii listei de frecvențe: range(5, 0, -1) în loc de range(1, 6). Contorizarea rămâne identică.*

HINT **Curiozitate.** *În camerele de gardă, codul 1 înseamnă urgență maximă, nu minimă — de aceea ordinea de tratare e crescătoare acolo. Aici sortăm descrescător tocmai ca să exersăm celălalt sens.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · COD GHIDAT

Câte valori distincte

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi din intervalul 0..100, fiecare de pe câte un rând. Afișează câte valori distincte apar printre ele. Ex.: pentru n = 6 și numerele 3 7 3 100 7 3 valorile distincte sunt 3, 7 și 100, deci se afișează 3.

HINT *Nu compara numerele două câte două: după contorizare, o valoare este prezentă dacă și numai dacă are contorul strict mai mare decât 0.*

HINT **Atenție.** *Intervalul este 0..100, deci lista de frecvențe are 101 poziții și parcurgerea începe de la 0, nu de la 1. Testul cu cinci zerouri verifică exact asta.*

SOLUȚIA TA · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

Mediana înălțimilor

SCRIERE COD

Se citește un număr natural impar n, apoi n înălțimi exprimate în centimetri, numere întregi din intervalul 100..200, fiecare de pe câte un rând. Afișează înălțimea aflată exact la mijlocul șirului sortat crescător (mediana), folosind lista de frecvențe și fără a construi șirul sortat. Ex.: pentru n = 5 și înălțimile 170 165 180 165 175, șirul sortat ar fi 165 165 170 175 180, iar valoarea din mijloc este 170 — a treia, adică a 5 // 2 + 1-a.

HINT *Lista de frecvențe are 201 poziții, iar primele 100 rămân nefolosite: indicele este chiar înălțimea, fără nicio scădere. E varianta simplă, dar risipitoare — cea economică ar fi să indexăm cu x - 100.*

HINT **Atenție.** *Parcurgerea contoarelor pornește de la 100, nu de la 0: sub 100 cm nu există nicio înălțime posibilă, iar contoarele acelea sunt oricum zero.*

SOLUȚIA TA · PYTHON

4

PROBLEMA 7.4 · REPARĂ CODUL

Indicele sau conținutul?

SCRIERE COD

Programul de mai jos ar trebui să afișeze notele sortate crescător, dar pentru n = 5 și notele 10 7 10 3 7 afișează 1 2 2 2 2 în loc de 3 7 7 10 10. El încalcă afirmația centrală din teoria lecției: într-o listă de frecvențe, indicele este valoarea, iar conținutul este numărul ei de apariții. Găsește greșeala,

repar-o și afișează notele sortate crescător, pe un singur rând, separate prin câte un spațiu.

Programul stricat:

```
n = int(input())
f = [0] * 11
for i in range(n):
    x = int(input())
    f[x] = f[x] + 1
for nota in range(1, 11):
    for j in range(f[nota]):
        print(f[nota], end=" ")
print()
```

Ex.: pentru n = 5 și notele 10 7 10 3 7, contoarele sunt f[3] = 1, f[7] = 2, f[10] = 2. Programul stricat afișează de câte ori apare fiecare notă (1, apoi 2 2, apoi 2 2) în loc să afișeze notele înseși (3, apoi 7 7, apoi 10 10).

HINT *Numărul de valori afișate este corect — sunt tot cinci. Greșit e CE se afișează: bucla interioară rulează de câte ori trebuie, dar tipărește conținutul contorului în loc de indicele lui.*

HINT **Atenție.** *Toate liniile programului sunt corecte, cu o singură excepție, aflată în bucla interioară. Apasă „Verifică soluția” înainte de a modifica ceva, ca să vezi cu ochii tăi ieșirea greșită, apoi compară-o cu cea așteptată.*

PROGRAMUL CORECTAT · PYTHON

PROBLEMA 7.5 · PROVOCARE

Perechi cu aceeași notă

SCRIERE COD

Se citește un număr natural n, apoi n note, numere întregi din intervalul 1..10, fiecare de pe câte un rând. Afișează câte perechi de elevi diferiți au aceeași notă. O pereche se numără o singură dată: perechea (Ana, Bogdan) și perechea (Bogdan, Ana) sunt aceeași pereche. Ex.: pentru n = 6 și notele 8 4 8 10 4 8: cei trei elevi cu nota 8 formează 3 perechi, cei doi cu nota 4 formează o pereche, iar elevul cu 10 nu are pereche — în total 4.

HINT *Dacă o valoare apare de f ori, dintre ei se pot forma f * (f - 1) // 2 perechi (împărțire întreagă, ca la exercițiul din secțiunea 5). Nu e nevoie să compari elevii doi câte doi: numărul se calculează direct din contor.*

HINT **Atenție.** *Formula funcționează și pentru f = 0 și f = 1, unde dă 0 — deci o poți aplica la toate cele zece note, fără nicio verificare suplimentară.*

HINT **Curiozitate.** *Aceeași formulă spune de ce, într-o clasă de 30 de elevi, sunt 435 de perechi posibile de colegi de bancă.*

SOLUȚIA TA · PYTHON