test-e.ro

CAIET ELEV

Lecția 24

Metode de sortare. Sortare prin selecția minimului

A sorta o listă înseamnă a-i rearanja elementele după un criteriu de ordine, fără să adaugi, să pierzi sau să modifici vreo valoare.

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: secvențial, if/elif/else, while, for cu range, %, //, liste (literal, indexare, len, for pe listă, metode de listă), slicing, import math, def/return. Nicio soluție nu folosește .sort(), sorted(), .reverse(), min() sau max() — ar anula exact ce învățăm. Interschimbarea se scrie cu variabila temporară aux, în trei linii; fără atribuiri simultane. Fără dicționare, fără fișiere, fără clase. Fără break sau continue. Fără diacritice în codul Python.

CUPRINS

1 Ce știm deja despre liste .....
2 Ce înseamnă „a sorta” .....
3 Prima metodă: interschimbări repetate .....
4 Sortarea prin selecția minimului .....
5 Cât costă fiecare metodă .....
6 Când sortezi un clasament .....
7 Cod ghidat · cinci probleme .....

SECȚIUNEA 1 · CE ȘTIM DEJA: LISTE, POZIȚII ȘI INTERSCHIMBARE

1 Ce știm deja despre *liste*

Înainte de a rearanja o listă, trebuie să știm sigur trei lucruri: cum ajungem la elementul de pe o poziție dată, cum găsim poziția celui mai mic element și cum mutăm două valori una în locul celeilalte fără să pierdem niciuna. Le reluăm pe toate trei aici, pe scurt.

O listă este o succesiune de valori numerotate de la zero: în `v = [7, 2, 9, 2, 5]`, `v[0]` este `7`, iar `v[4]` este `5`. Numărul de elemente îl dă `len(v)`, iar ultimul indice valid este `len(v) - 1` — indicele `5` într-o listă cu cinci elemente nu există. Când vrem să lucrăm cu pozițiile, parcurgem lista cu `for i in range(len(v))` și folosim `v[i]`; când ne interesează doar valorile, putem scrie direct `for x in v`.

A doua unealtă este căutarea poziției celei mai mici valori. Ideea e simplă: presupunem că cea mai mică se află pe prima poziție și reținem indicele ei în `p`; apoi comparăm fiecare element următor cu `v[p]` și, ori de câte ori găsim ceva strict mai mic, actualizăm `p`.

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

DE REȚINUT

Comparăm cu `<` strict, nu cu `<=`. Diferența se vede când valoarea minimă apare de mai multe ori: cu `<` reținem **prima** ei apariție, cu `<=` am reține ultima. Pentru lista de mai sus, `p` rămâne `1`, nu devine `3`.

A treia unealtă este interschimbarea. Ea pare banală, dar are o capcană: dacă scriem direct `v[i] = v[j]`, valoarea veche din `v[i]` se pierde definitiv și nu mai avem ce pune în `v[j]`. De aceea o salvăm întâi într-o variabilă temporară.

```
aux = v[i]
v[i] = v[j]
v[j] = aux
```

ATENȚIE

> Ordinea celor trei linii nu e negociabilă. Dacă începi cu `v[i] = v[j]`, ambele poziții ajung să conțină aceeași valoare, iar cealaltă dispare din listă — o greșeală pe care o vom întâlni la propriu în exercițiul de remediere din secțiunea 7.

A

EXERCIȚIUL A · GRILĂ

Liste, poziții, minim

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Pentru `v = [4, 9, 2, 7]`, ce afișează `print(v[2])`?

- `Nimic, pentru că indicele 2 depășește lungimea unei liste cu 4 elemente`
- `2, pentru că indicii încep de la zero, deci v[2] este al treilea element`
- `7, pentru că indicele 2 se numără dinspre sfârșitul listei către început`
- `9, pentru că numărătoarea elementelor unei liste începe de la unu`

2. Pentru `v = [3, 8, 1]`, ce afișează `for i in range(1, len(v)): print(v[i], end=" ")`?

- `3 8 — parcurgerea sare peste ultimul element, pentru că indicii merg până la len(v)`
- `8 1 — parcurgerea pornește de la indicele 1 și merge până la ultimul element`
- `8 1 3 — parcurgerea începe din mijlocul listei și se închide cu primul element`
- `3 8 1 — range(1, len(v)) acoperă toată lista, începând cu primul element`

3. Care secvență interschimbă corect valorile de pe pozițiile `0` și `2`?

- `aux = v[0], apoi v[2] = aux, apoi v[0] = v[2] — copia se folosește la sfârșit`
- `v[0] = v[2], apoi v[2] = v[0] — cele două valori își iau locul una alteia direct`
- `v[0] = aux, apoi aux = v[2], apoi v[2] = v[0] — variabila se pregătește la final`
- `aux = v[0], apoi v[0] = v[2], apoi v[2] = aux`

4.

Pentru `v = [7, 2, 9, 2, 5]`, ce valoare are `p` după secvența `p = 0` urmată de `for i in range(1, len(v)): if v[i] < v[p]: p = i`?

- `1 — poziția primei apariții a celei mai mici valori din listă`
- `3 — poziția ultimei apariții a celei mai mici valori din listă`
- `0 — valoarea inițială nu se schimbă niciodată în interiorul acestei bucle`
- `2 — poziția celei mai mari valori, pentru că bucla reține maximul`

B

EXERCIȚIUL B · DRY-RUN

Unde se ascunde cel mai mic element

DRY-RUN

Urmărește căutarea poziției minimului în lista `v = [7, 2, 9, 2, 5]`. Coloanele i și v[i] sunt date; tu completezi valoarea lui v[p] înainte de comparație (numerică), verdictul comparației (DA / NU) și valoarea lui p după pas (numerică). Ultimul rând arată starea în care bucla se oprește.

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

i (dat)	v[i] (dat)	v[p] înainte	v[i] < v[p]?	p după pas
1	2			
2	9			
3	2			
4	5			
5	–	–	–	1

HINT *La fiecare pas compari v[i] cu v[p], adică cu valoarea de pe poziția reținută în p. Actualizezi p doar când v[i] este STRICT mai mic decât v[p].*

SPRIJIN · SCRIE CODUL

Poziția celui mai mic element

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Afișează pe același rând poziția primului element minim din listă, un spațiu, apoi valoarea lui. Ex.: pentru n = 5 și numerele 7 2 9 2 5, cea mai mică valoare este 2 și apare prima dată pe poziția 1, deci se afișează 1 2.

HINT *Pozițiile se numără de la zero, deci pentru o listă cu un singur element răspunsul începe cu 0.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Schimbă două valori între ele

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând, apoi încă două numere i și j, care sunt poziții valide în listă. Interschimbă elementele de pe pozițiile i și j folosind o variabilă temporară, apoi afișează lista rezultată pe un singur rând, cu elementele separate prin câte un spațiu. Ex.: pentru n = 5, numerele 3 1 4 1 5 și pozițiile 0 și 4, lista devine 5 1 4 1 3.

HINT **Atenție.** *Dacă i și j sunt egale, lista trebuie să rămână neschimbată — verifică asta pe testul cu i = j = 2.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 2 · CE ÎNSEAMNĂ „A SORTA” O LISTĂ

2 Ce înseamnă „a *sorta*”

A sorta nu înseamnă a schimba valorile, ci a le rearanja. După sortare lista conține exact aceleași numere ca înainte — doar că așezate în ordinea cerută de criteriu. Aici stabilim ce înseamnă „sortat”, cum verificăm dacă o listă chiar este sortată și ce înseamnă că o sortăm „în loc”.

O listă este **sortată crescător** dacă niciun element nu este mai mare decât cel care îl urmează: `v[0] <= v[1] <= ... <= v[n-1]`. Dacă lista are valori egale, condiția rămâne îndeplinită — de aceea folosim `<=`, nu `<`. Simetric, lista este **sortată descrescător** dacă niciun element nu este mai mic decât următorul. Regula după care comparăm două elemente se numește **criteriu de ordonare**; pentru numere el este evident, dar pentru un clasament poate fi „scor mai mare înseamnă loc mai bun”, iar pentru o agendă „ordinea alfabetică a numelor”.

Sortarea este o **rearanjare** — în limbaj matematic, o **permutare** a elementelor: aceleași valori, altă așezare. Nimic nu se pierde, nimic nu se adaugă, nicio valoare nu se modifică. Lista `[5, 2, 8, 1]` sortată crescător devine `[1, 2, 5, 8]` — verifică: aceleași patru numere.

Cele două metode din această lecție sortează **în loc**: lucrează chiar în lista dată, mutând valorile între pozițiile ei, fără să construiască o listă nouă. Avantajul e că nu consumă memorie în plus; „prețul” e că lista inițială se pierde — dacă mai ai nevoie de ea, îți faci întâi o copie cu `v[:]`.

Verificarea faptului că o listă este deja sortată se face dintr-o singură parcurgere: compari fiecare element cu vecinul din stânga și, dacă găsești o pereche în ordine greșită, răspunsul este „nu”. Codul de mai jos parcurge lista până la capăt — am putea opri căutarea la prima pereche greșită, dar pentru asta ne-ar trebui `break`, pe care nu îl folosim în această lecție.

```
sortata = True
for i in range(1, n):
    if v[i] < v[i - 1]:
        sortata = False
```

DE REȚINUT

Python are deja `v.sort()` și `sorted(v)`, învățate la lecția 17, și în programele reale exact ele se folosesc. În această lecție **nu le folosim**: scopul e să înțelegem ce se întâmplă înăuntrul lor. Un instrument pe care îl poți construi singur nu-ți mai este o cutie neagră.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Catalogul electronic afișează elevii în ordine alfabetică, deși profesorul îi adaugă în ordinea în care se înscriu. Aplicația de muzică îți sortează melodiile după numărul de ascultări ca să-ți construiască topul personal. Magazinul online îți arată produsele „de la cel mai ieftin la cel mai scump” — aceeași listă, alt criteriu de ordonare.

A

EXERCIȚIUL A · GRILĂ

Sortat sau doar amestecat?

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Care dintre liste este sortată crescător?

   - `[9, 5, 2, 2]` — valorile sunt așezate de la cea mai mare la cea mai mică
   - `[5, 2, 9, 2]` — lista conține aceleași valori, doar așezate în altă ordine
   - `[2, 5, 9, 2]` — lista începe crescător, deci se consideră sortată crescător
   - `[2, 2, 5, 9]` — valorile nu scad niciodată de la stânga la dreapta

2. Ce înseamnă că o sortare se face „în loc”?

   - `Sortarea are voie să schimbe și valorile elementelor, nu doar pozițiile lor`
   - `Elementele se rearanjează chiar în lista dată, fără a se construi o listă nouă`
   - `Lista se copiază mai întâi, iar rezultatul se scrie în copia proaspăt creată`
   - `Lista sortată apare doar pe ecran, iar în memorie lista rămâne neschimbată`

3. O listă cu 6 elemente a fost sortată crescător. Ce s-a schimbat față de lista inițială?

   - `Doar valorile care se repetă dispar, iar ordinea celorlalte rămâne neatinsă`
   - `Nimic în listă, pentru că sortarea schimbă doar felul în care se afișează`
   - `Doar ordinea elementelor; valorile din listă sunt exact aceleași ca înainte`
   - `Și ordinea, și valorile, pentru că unele elemente se pierd în timpul sortării`

B

EXERCIȚIUL B · SCRIE CODUL

Este lista deja sortată?

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Afișează pe același rând primul element al listei, un spațiu, ultimul element, un spațiu, apoi DA dacă lista este sortată crescător sau NU în caz contrar. Ex.: pentru n = 5 și numerele 1 3 3 8 9 se afișează 1 9 DA, pentru că niciun element nu este mai mare decât următorul.

HINT *Valorile egale nu strică sortarea: 1 3 3 8 9 rămâne sortată crescător.*

HINT **Atenție.** *O listă cu un singur element este întotdeauna sortată, iar primul și ultimul element sunt același număr.*

SOLUȚIA TA · PYTHON

3 SECȚIUNEA 3 · PRIMA METODĂ: INTERSCHIMBĂRI REPETATE

Prima metodă: interschimbări *repetate*

Cea mai directă idee de sortare este și cea mai grăbită: fixează o poziție, compar-o cu toate cele din dreapta ei și, de fiecare dată când găsești ceva mai mic, mută-l imediat acolo. Metoda funcționează — dar plătește un preț pe care îl vom măsura în secțiunea următoare.

METODA NAIVĂ — INTERSCHIMBĂRI REPETATE

Pentru fiecare poziție `i`, de la `0` la `n - 2`, compară `v[i]` cu fiecare element `v[j]` aflat la dreapta lui. Dacă `v[j] < v[i]`, interschimbă-le **pe loc**. La sfârșitul pasului, pe poziția `i` se află cel mai mic element din porțiunea `i .. n-1`.

De ce funcționează? Pe poziția `i` ține mereu „cel mai mic văzut până acum” din porțiunea din dreapta. Fiecare valoare mai mică descoperită o alungă pe cea de acolo, care pleacă în locul ei. La finalul pasului, nimic din partea dreaptă nu mai este mai mic decât `v[i]` — altfel s-ar fi făcut încă o interschimbare. Poziția `i` este deci definitiv rezolvată, iar pasul următor se ocupă doar de porțiunea rămasă.

```
for i in range(n - 1):
    for j in range(i + 1, n):
        if v[j] < v[i]:
            aux = v[i]
            v[i] = v[j]
            v[j] = aux
```

Urmărește primul pas pe lista `[5, 2, 8, 1]`: `2 < 5`, deci interschimbăm și lista devine `[2, 5, 8, 1]`; `8 < 2` este fals, nu se întâmplă nimic; `1 < 2`, deci interschimbăm din nou și lista devine `[1, 5, 8, 2]`. Un singur pas, **două** interschimbări — deși în listă cel mai mic element era unul singur.

ATENȚIE

Bucla interioară pornește de la `i + 1`, nu de la `0`. Dacă ar porni de la `0`, ar recompara elemente deja așezate definitiv și ar strica ordinea din partea stângă a listei.

DE REȚINUT

Această metodă se numește, în literatură, **sortare prin interschimbare** (*exchange sort*). Ea nu trebuie confundată cu **metoda bulelor**, pe care o vei studia mai târziu: acolo se compară doar elemente **vecine**, aici se compară `v[i]` cu **toate** elementele din dreapta lui.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Așa procedezi instinctiv când aranjezi în mână cărțile de joc și le muți una câte una, de fiecare dată când vezi una mai mică. La fel se aranjează borcanele pe raftul din cămară după data expirării, mutându-l în față pe fiecare care expiră mai devreme. Este metoda „la vedere”: mult efort de mutat, puțin efort de gândit.

A

EXERCIȚIUL A · DRY-RUN

Primul pas al metodei naive

DRY-RUN

Urmărește doar primul pas (i = 0) al metodei naive, pe lista `v = [5, 2, 8, 1]`. Coloanele j și v[j] sunt date; tu completezi valoarea lui v[0] înainte de comparație (numerică) și verdictul (DA dacă are loc interschimbarea, NU altfel). Ultimul rând arată starea în care bucla se oprește.

```
v = [5, 2, 8, 1]
i = 0
for j in range(i + 1, 4):
    if v[j] < v[i]:
        aux = v[i]
        v[i] = v[j]
        v[j] = aux
print(v)
```

j (dat)	v[j] (dat)	v[0] înainte	Se interschimbă?
1	2		
2	8		
3	1		
4	–	1	–

HINT *Compari mereu cu valoarea aflată ACUM pe poziția 0 — ea se schimbă după fiecare interschimbare. Interschimbarea are loc doar când v[j] este strict mai mic decât v[0].*

B

EXERCIȚIUL B · SCRIE CODUL

Sortare prin interschimbări repetate

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Sortează lista crescător prin metoda interschimbărilor repetate și afișează elementele sortate pe un singur rând, separate prin câte un spațiu. Ex.: pentru n = 5 și numerele 5 2 8 1 9 se afișează 1 2 5 8 9.

HINT **Atenție.** *Interschimbarea se face cu variabilă temporară. Dacă scrii direct v[i] = v[j] urmat de v[j] = v[i], ambele poziții ajung să conțină aceeași valoare.*

HINT *O listă cu un singur element este deja sortată — bucla exterioară nu rulează deloc, iar programul trebuie să o afișeze corect.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 4 · SORTAREA PRIN SELECȚIA MINIMULUI

4 Sortarea prin selecția *minimului*

A doua metodă schimbă un singur lucru — dar acel lucru face toată diferența: nu mai mută nimic în timp ce caută. Întâi află unde se ascunde cel mai mic element din partea nesortată, ține minte doar poziția lui, și abia la final face o singură interschimbare.

SELECȚIA MINIMULUI

Pentru fiecare poziție `i`, de la `0` la `n - 2`: presupune că minimul porțiunii nesortate este chiar pe poziția `i` (`p = i`); parcurge apoi toate pozițiile `j` din dreapta și, ori de câte ori `v[j] < v[p]`, reține noul indice (`p = j`); la sfârșitul căutării, dacă `p` este diferit de `i`, interschimbă `v[i]` cu `v[p]`.

Diferența față de metoda naivă este că `p` reține o **poziție**, nu o valoare, iar interschimbarea are loc **o singură dată pe pas**, la sfârșit. Pe lista `[5, 2, 8, 1]`, primul pas caută și găsește minimul `1` pe poziția `3`, apoi face o singură interschimbare: lista devine `[1, 2, 8, 5]`. Metoda naivă avea nevoie de două mutări ca să ajungă la un rezultat asemănător.

```python
for i in range(n - 1):
    p = i
    for j in range(i + 1, n):
        if v[j] < v[p]:
            p = j
    if p != i:
        aux = v[i]
        v[i] = v[p]
        v[p] = aux
```

După pasul `i`, porțiunea `v[0] .. v[i]` este sortată **definitiv**: conține cele mai mici `i + 1` valori, în ordine. De aceea bucla exterioară se oprește la `n - 2`: când au rămas de așezat toate elementele în afară de ultimul, ultimul e obligatoriu cel mai mare, deci e deja la locul lui.

ATENȚIE

`p = i` trebuie rescris **la fiecare pas**, în interiorul buclei exterioare. Dacă îl scrii o singură dată, înaintea buclelor, la pașii următori vei compara cu un element care a fost deja mutat, iar rezultatul va fi greșit fără ca programul să dea vreo eroare.

CURIOZITATE DE SINTAXĂ

Python știe să interschimbe două valori dintr-o singură instrucțiune: `v[i], v[p] = v[p], v[i]`. Scrierea e elegantă, dar ascunde exact ce vrem să vedem aici — de aceea în lecție folosim peste tot varianta cu `aux`, în trei pași.

UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ

Așa se stabilește ordinea de start la un concurs de schi: comisia caută întâi cel mai bun timp din calificări, îl trece în capul listei, apoi reia căutarea printre cei rămași. La fel alegi cel mai copt fruct din coș înainte de a-l pune deoparte, sau cel mai apropiat termen dintr-o listă de sarcini pe care vrei să o rescrii în ordinea urgenței.

A

EXERCIȚIUL A · ORDONARE

Pașii selecției minimului

ORDONARE

Numerotează pașii de la 1 la 6, în ordinea în care se execută pentru un singur pas al buclei exterioare. Pașii îți apar amestecați.

- [ ] Treci la poziția următoare şi repetă, până când rămâne un singur element neașezat.
- [ ] La sfârșitul căutării, dacă p este diferit de i, interschimbă v[i] cu v[p] folosind aux.

- [ ] Ori de câte ori v[j] este strict mai mic decât v[p], reține noul indice: p = j.
- [ ] Consideră prima poziție nesortată, i, ca punct de plecare al pasului curent.
- [ ] Presupune că minimul se află chiar pe poziția i și reține p = i.
- [ ] Parcurge toate pozițiile j din dreapta lui i și compară v[j] cu v[p].

B

EXERCIȚIUL B · SCRIE CODUL

Sortare prin selecția minimului

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Sortează lista crescător prin metoda selecției minimului — cu reținerea poziției minimului în variabila p și cel mult o interschimbare pe pas — și afișează elementele sortate pe un singur rând, separate prin câte un spațiu. Ex.: pentru n = 5 și numerele 9 3 7 1 8 se afișează 1 3 7 8 9.

HINT *Verificarea if p != i nu schimbă rezultatul, dar evită o interschimbare inutilă atunci când minimul este deja la locul lui.*

HINT **Atenție.** *Variabila p se inițializează cu i la fiecare pas al buclei exterioare, nu o singură dată la începutul programului.*

SOLUȚIA TA · PYTHON

PERFORMANȚĂ · SCRIE CODUL

Cele mai mici k valori

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând, apoi un număr natural k, cu 1 <= k <= n. Aplică doar primii k pași ai selecției minimului și afișează primele k elemente ale listei, pe un singur rând, separate prin câte un spațiu. Ex.: pentru n = 6, numerele 9 3 7 1 8 2 și k = 3, primii trei pași aduc în față valorile 1, 2 și 3, deci se afișează 1 2 3.

HINT *Restul listei rămâne nesortat — și nici nu ne interesează. Aici e câștigul: pentru k = 3 faci trei pași, nu n - 1.*

HINT **Curiozitate.** *Exact așa funcționează un clasament „Top 10” pe un site cu milioane de intrări: nimeni nu sortează tot, ci doar extrage primele zece.*

SOLUȚIA TA · PYTHON

5 SECȚIUNEA 5 · CÂT COSTĂ? COMPARĂRI, INTERSCHIMBĂRI ȘI O(N²)

Cât costă fiecare *metodă*

Cele două metode dau exact același rezultat. Ca să alegem între ele, trebuie să le măsurăm: câte comparări face fiecare și câte mutări. Prima cifră iese identică — a doua le desparte net.

Comparările. La primul pas, elementul din poziția `0` se compară cu toate cele `n - 1` de la dreapta lui. La al doilea pas rămân `n - 2` comparări, apoi `n - 3`, și așa mai departe, până la una singură. Totalul este suma primelor `n - 1` numere naturale:

```
(n - 1) + (n - 2) + ... + 2 + 1 = n(n - 1) / 2
```

Formula nu depinde de valorile din listă: **ambele metode fac exact același număr de comparări**, chiar dacă lista e deja sortată.

N	COMPARĂRI = N(N − 1)/2
5	10
10	45
100	4 950
1000	499 500

Când `n` se dublează, numărul de comparări crește de aproape patru ori: de la 10 la 45, de la 4 950 la 19 900. Spunem că metodele au complexitate **O(n²)** — costul crește ca pătratul numărului de elemente. Pentru 1000 de elemente sunt aproape jumătate de milion de comparări; pentru un milion de elemente ar fi de neatins într-un timp rezonabil, motiv pentru care în practică se folosesc metode mai bune, pe care le vei studia mai târziu.

Interschimbările. Aici cele două metode se despart. Metoda naivă mută de fiecare dată când găsește ceva mai mic, deci în cel mai rău caz face tot `n(n - 1) / 2` interschimbări. Selecția minimului face cel mult **una pe pas**, deci cel mult `n - 1` în total. Pentru o listă de 100 de elemente aranjată invers: 4 950 de mutări față de 99.

N	INTERSCHIMBĂRI (NAIVĂ, CEL MAI RĂU CAZ)	INTERSCHIMBĂRI (SELECȚIE, CEL MULT)
5	10	4
10	45	9
100	4 950	99

> **DE REȚINUT**
>
> Numărul de comparări nu ne ajută să alegem între cele două metode — este identic. Ne ajută numărul de **mutări**. De aceea selecția minimului este metoda pe care o reținem: face aceeași căutare, dar plătește mult mai puțin pentru rearanjare.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**

> Un depozit compară pe hârtie toate coletele ca să afle care pleacă primul, dar le ridică fizic o singură dată — mutarea e cea care costă. Într-o bază de date, compararea a două înregistrări e ieftină, dar mutarea uneia poate însemna rescrierea a mii de octeți pe disc. Iar când muți mobila în cameră, măsori tot, apoi cari o singură dată.

A

EXERCIȚIUL A · SCRIE CODUL

Câte mutări face fiecare metodă?

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Sortează crescător o copie a listei cu metoda naivă și altă copie cu selecția minimului, numărând interschimbările făcute de fiecare. Afișează pe primul rând numărul de interschimbări făcute de metoda naivă, iar pe al doilea rând numărul făcut de selecția minimului. Ex.: pentru n = 5 și numerele 5 4 3 2 1, metoda naivă face 10 interschimbări, iar selecția minimului doar 2, deci se afișează 10 pe primul rând și 2 pe al doilea.

HINT *Copiile se fac cu a = v[:] și b = v[:]. Dacă ai scrie a = v, cele două nume ar desemna aceeași listă, iar a doua metodă ar primi lista deja sortată de prima.*

HINT **Atenție.** *La selecția minimului, interschimbarea se numără doar când are loc efectiv, adică în interiorul lui if p != i.*

SOLUȚIA TA · PYTHON

B

EXERCIȚIUL B · ASOCIERE

Câte comparări pentru fiecare listă?

ASOCIERE

Fiecărei dimensiuni de listă din stânga îi corespunde numărul de comparări făcute de metodă. Scrie în căsuță litera potrivită. Fiecare literă se folosește o singură dată.

- ☐ O listă cu 4 elemente
- ☐ O listă cu 5 elemente
- ☐ O listă cu 10 elemente
- ☐ O listă cu 20 de elemente

A 190 de comparări | B 6 comparări | C 10 comparări | D 45 de comparări

6 Când sortezi un *clasament*

SECȚIUNEA 6 · CLASAMENTUL UNUI CONCURS: DOUĂ LISTE CARE SE MIȘCĂ ÎMPREUNĂ

În realitate nu sortezi niciodată numere singure. Sortezi concurenți după scor, produse după preț, elevi după medie — adică perechi. Aici învățăm regula care ține perechile legate în timpul sortării.

Un clasament are două informații pentru fiecare concurent: **numărul de concurs** și **scorul**. Le păstrăm în două liste **paralele**: `nr[i]` și `sc[i]` se referă la același concurent — pozițiile din cele două liste sunt legate una de alta prin indice.

Sortăm descrescător după scor, adică prin **selecția maximului**: la fiecare pas căutăm poziția celui mai mare scor rămas și o aducem în față. Regula nouă e simplă și absolut obligatorie: **orice interschimbare se aplică la ambele liste, la aceiași doi indici**.

```
if p != i:
    aux = sc[i]
    sc[i] = sc[p]
    sc[p] = aux
    aux = nr[i]
    nr[i] = nr[p]
    nr[p] = aux
```

Ordinea celor două blocuri nu contează, dar **prezența amândurora** contează enorm: dacă muți doar scorurile, obții o listă de scoruri sortată perfect, lipită de numere de concurs care nu mai au nicio legătură cu ele. Rezultatul e un clasament în care premiul întâi merge la altcineva — o eroare care nu produce niciun mesaj de eroare.

> **ATENȚIE**
>
> Comparația se face **numai** pe lista după care sortăm (`sc`), iar interschimbarea pe **amândouă**. Dacă ai compara și după `nr`, ai amesteca două criterii diferite.

> **UNDE SE FOLOSEȘTE ÎN VIAȚA REALĂ**
>
> Așa se așază finaliștii unei olimpiade pe scenă: comisia sortează după punctaj, dar mută pe scenă persoana, nu cifra. Tot așa, aplicația de sport îți ordonează echipele după numărul de puncte, cărând după fiecare punctaj numele, emblema și golaverajul. Iar când îți sortezi pozele după dată, sistemul mută fișierul întreg, nu doar data lui.

A

EXERCIȚIUL A · SCRIE CODUL

Clasamentul concursului

SCRIERE COD

Se citește un număr natural n, numărul de concurenți, apoi pentru fiecare concurent două numere, fiecare de pe câte un rând: numărul lui de concurs și scorul obținut. Afișează clasamentul, câte un concurent pe rând, în ordinea descrescătoare a scorului: numărul de concurs, un spațiu, apoi scorul. Ex.: pentru n = 3 și perechile 101 40, 102 95, 103 70, clasamentul afișat este 102 95, apoi 103 70, apoi 101 40.

HINT *Scorurile sunt distincte, deci ordinea finală este unică.*

HINT **Atenție.** *La fiecare interschimbare mută elementele din ambele liste, la aceiași doi indici. Dacă muți doar scorurile, numerele de concurs rămân pe loc și clasamentul devine greșit, fără niciun mesaj de eroare.*

SOLUȚIA TA · PYTHON

B

EXERCIȚIUL B · COMPLETARE

Regula listelor paralele

COMPLETARE

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

`pierde` `ambele` `sortată` `aceiași` `paralelă` `diferiți`

Când sortăm un clasament, scorurile stau într-o listă, iar numerele de concurs stau într-o listă _____. La fiecare interschimbare mutăm elemente din _____ liste, folosind _____ doi indici. Dacă am muta doar scorurile, legătura dintre concurent și rezultatul lui s-ar _____.

7 SECȚIUNEA 7 · COD GHIDAT

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

Clasament invers

SCRIERE COD

Se citește un număr natural n, apoi n numere întregi, fiecare de pe câte un rând. Sortează lista descrescător prin selecția maximului și afișează elementele pe un singur rând, separate prin câte un spațiu. Ex.: pentru n = 5 și numerele 3 9 1 7 5 se afișează 9 7 5 3 1.

HINT *Față de selecția minimului se schimbă un singur caracter: comparația devine v[j] > v[p].*

HINT **Atenție.** *Valorile egale nu strică nimic — trebuie doar să apară toate în rezultat, testul cu trei valori de 6 verifică exact asta.*

SOLUȚIA TA · PYTHON

2

PROBLEMA 7.2 · COD GHIDAT

Elementul din mijloc

SCRIERE COD

Se citește un număr natural impar n, apoi n numere întregi, fiecare de pe câte un rând. Sortează lista crescător prin selecția minimului și afișează elementul aflat exact în mijlocul listei sortate. Ex.: pentru n = 5 și numerele 7 1 9 3 5, lista sortată este 1 3 5 7 9, iar elementul din mijloc este 5.

HINT *Pentru n impar, poziția din mijloc este n // 2: la 5 elemente, poziția 2; la 7 elemente, poziția 3.*

HINT **Curiozitate.** *Valoarea din mijlocul listei sortate se numește mediană și e folosită în statistică pentru că, spre deosebire de medie, nu e influențată de câteva valori extreme.*

SOLUȚIA TA · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

Cele mai apropiate două valori

SCRIERE COD

Se citește un număr natural n, cu n >= 2, apoi n numere întregi, fiecare de pe câte un rând. Afișează cea mai mică diferență, în modul, dintre două elemente aflate pe poziții diferite ale listei. Cu alte cuvinte: cât de apropiate sunt cele mai apropiate două valori din listă. Ex.: pentru n = 5 și numerele 10 3 25 7 40, lista sortată este 3 7 10 25 40, iar cea mai mică diferență dintre două valori vecine este 10 - 7 = 3.

HINT *După sortare, cele două valori cele mai apropiate sunt obligatoriu vecine — de aceea e suficient să compari perechile alăturate, nu toate perechile posibile.*

HINT **Atenție.** *Dacă o valoare apare de două ori, diferența minimă este 0; testul cu două valori de 8 verifică exact acest caz.*

SOLUȚIA TA · PYTHON

4

PROBLEMA 7.4 · REPARĂ CODUL

Interschimbarea care înghite valori

SCRIERE COD

Programul de mai jos ar trebui să sorteze crescător lista prin selecția minimului, dar pentru n = 5 și numerele 9 3 7 1 8 afișează 1 1 1 1 8 în loc de 1 3 7 8 9. Găsește greșeala, repar-o și afișează lista corect sortată, pe un singur rând, cu elementele separate prin câte un spațiu.

Programul stricat:

```
n = int(input())
v = []
for i in range(n):
    x = int(input())
    v.append(x)
for i in range(n - 1):
    p = i
    for j in range(i + 1, n):
        if v[j] < v[p]:
            p = j
    v[i] = v[p]
    v[p] = v[i]
for i in range(n):
    print(v[i], end=" ")
print()
```

Ex.: După reparare, pentru n = 5 și numerele 9 3 7 1 8 programul trebuie să afișeze 1 3 7 8 9.

HINT *Căutarea minimului este corectă; problema este exclusiv în cele două linii care ar trebui să facă interschimbarea.*

HINT **Atenție.** *Prima linie suprascrie v[i] înainte ca vechea lui valoare să fie salvată undeva, iar a doua linie o copiază înapoi pe cea nouă — cele două poziții ajung să conțină aceeași valoare, iar valoarea veche dispare din listă.*

PROGRAMUL CORECTAT · PYTHON

PROBLEMA 7.5 · PROVOCARE

Clasament cu departajare

SCRIERE COD

Se citește un număr natural n, apoi pentru fiecare concurent două numere, fiecare de pe câte un rând: numărul de concurs și scorul. Afișează clasamentul, câte un concurent pe rând — numărul de concurs, un spațiu, scorul — în ordinea descrescătoare a scorului, iar la scoruri egale în ordinea crescătoare a numărului de concurs. Ex.: pentru n = 3 și perechile 12 80, 7 80, 9 95, clasamentul este 9 95, apoi 7 80 (număr de concurs mai mic), apoi 12 80.

HINT *Criteriul are acum două trepte: prima decide după scor, a doua intră în joc doar când scorurile sunt egale.*

HINT **Atenție.** *Cele două condiții se scriu ca două ramuri ale aceleiași căutări (if pentru scor mai mare, elif pentru scor egal și număr mai mic), iar interschimbarea rămâne una singură, pe ambele liste.*

SOLUȚIA TA · PYTHON