INFORMATICĂ · PYTHON · CLASA A IX-A · STRATEGII DE REZOLVARE A PROBLEMELOR · LECȚIA 17

Metode de sortare: selecție, frecvențe, bule

Durată: 150 min Nivel: mate-info · Python Înainte: lecțiile py9-14, py9-15, py9-16

CE VEI ȘTI DUPĂ LECȚIE

- Aplici pas cu pas sortarea prin selecția minimului, metoda bulelor și sortarea cu listă de frecvențe și le implementezi în Python.
- Numeri comparațiile și interschimbările fiecărei metode în cazul favorabil și nefavorabil.
- Alegi metoda potrivită după datele problemei (câte valori, ce interval, cât de ordonate sunt deja).
- Compari metodele scrise de tine cu list.sort() și sorted().

1. De ce sortăm „de mână", dacă există sort()?

`a.sort()` (lecția py9-15) rezolvă orice sortare într-un rând. Totuși, studiem trei metode clasice, din trei motive: înțelegi **ce costă** o sortare (comparații, interschimbări), poți **adapta** metoda (doar primele 10 valori, doar o parte din listă, alt criteriu) și le întâlnești la BAC, în pseudocod, unde nu există `sort()`.

Toate metodele de mai jos sortează **crescător** și modifică lista primită. Interschimbarea a două elemente se scrie în Python fără variabilă auxiliară: `a[i], a[j] = a[j], a[i]`.

2. Sortarea prin selecția minimului

Ideea. Juriul unui concurs de talente așază finaliștii în ordinea scorului: caută scorul **cel mai mic** și îl pune pe primul loc, apoi caută cel mai mic dintre cei rămași și îl pune pe locul 2, și tot așa.

```
def selectie(a):
    n = len(a)
    for i in range(n - 1):              # locul pe care îl completăm
        p = i                           # poziția minimului din a[i:]
        for j in range(i + 1, n):
            if a[j] < a[p]:
                p = j
        a[i], a[p] = a[p], a[i]         # minimul ajunge pe locul i
```

Urmărire pentru `scoruri = [72, 95, 64, 88, 70]` (partea deja sortată e la stânga):

Pasul	Minimul din partea nesortată	Lista după pas
1	64 (poziția 2) ↔ 72	`[64, 95, 72, 88, 70]`
2	70 (poziția 4) ↔ 95	`[64, 70, 72, 88, 95]`
3	72 e deja pe loc	`[64, 70, 72, 88, 95]`
4	88 e deja pe loc	`[64, 70, 72, 88, 95]`

Cost. Pasul `i` face `n - 1 - i` comparații, deci în total `(n - 1) + (n - 2) + … + 1 = n(n - 1)/2`, **oricum ar arăta lista** (aici 10, și la fel pentru o listă deja sortată). Interschimbări: cel mult `n - 1` (aici 2 utile). Ordin de complexitate: $O(n^2)$.

3. Metoda bulelor

Ideea. Elevii se așază la rând după înălțime. Parcurgi rândul de la stânga la dreapta și, **ori de câte ori doi vecini sunt în ordine greșită, îi schimbi între ei**. După o trecere, cel mai înalt a ajuns la capăt, ca o bulă care urcă. Repeți până când o trecere nu mai face nicio schimbare.

```
def bule(a):
    n = len(a)
    for k in range(n - 1):
        schimbat = False
        for j in range(n - 1 - k):       # ultimele k elemente sunt deja la locul lor
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
                schimbat = True
        if not schimbat:                 # nicio schimbare: lista e sortată
            break
```

Urmărire pentru `inaltimi = [165, 158, 172, 150, 168]`:

Trecerea	Lista după trecere	Fixat la final
1	`[158, 165, 150, 168, 172]`	172
2	`[158, 150, 165, 168, 172]`	168
3	`[150, 158, 165, 168, 172]`	165
4	`[150, 158, 165, 168, 172]` (nicio schimbare)	

Aici: 10 comparații și 5 interschimbări.

Lista de 5 elemente	Comparații	Interschimbări
deja sortată `[1, 2, 3, 4, 5]`	4 (o trecere, apoi `break`)	0
inversă `[5, 4, 3, 2, 1]`	10	10

DE ȘTIUT

Fiecare interschimbare de vecini repară exact o pereche „inversată” (un element mai mare înaintea unuia mai mic). De aceea numărul de interschimbări al metodei bulelor este egal cu numărul de astfel de perechi din lista inițială. În cazul cel mai rău, `n(n - 1)/2`.

ATENȚIE — GREȘELI FRECVENTE

Fără variabila `schimbat`, metoda face mereu toate cele `n - 1` treceri, chiar dacă lista este deja sortată. Iar dacă bucla interioară merge până la `range(n)`, ultimul pas cere `a[n]`: `IndexError`.

4. Sortarea cu listă de frecvențe

Ideea. Notele unei clase la informatică sunt întregi de la 1 la 10. Nu le comparăm deloc: **numărăm** câte note de fiecare fel există (lista de frecvențe din lecția py9-14), apoi scriem fiecare valoare de atâtea ori câte apariții are, în ordinea valorilor.

```
def sort_frecvente(note):
    fr = [0] * 11
    for x in note:                  # 1. numărare
        fr[x] += 1
    rez = []
    for v in range(1, 11):          # 2. reconstruire, în ordinea valorilor
        for _ in range(fr[v]):
            rez.append(v)
    return rez

print(sort_frecvente([7, 10, 4, 7, 9, 5, 10, 7, 8]))
# [4, 5, 7, 7, 7, 8, 9, 10, 10]
```

Descrescător parcurgi valorile invers. Triajul unei camere de gardă ordonează pacienții după gravitate (coduri 1–5, 5 = cel mai grav):

```
grav = [3, 5, 1, 5, 2, 4, 3]
fr = [0] * 6
for g in grav:
    fr[g] += 1
rez = []
for v in range(5, 0, -1):
    rez.extend([v] * fr[v])
print(rez)                          # [5, 5, 4, 3, 3, 2, 1]
```

Cost. O parcurgere pentru numărare (`n` pași) și una pentru reconstruire (`M` valori posibile, plus cele `n` elemente scrise): O(n + M). Pentru mii de pacienți cu coduri 1–5, este mult mai rapidă decât orice metodă cu comparații.

> **LA BAC**
>
> Metoda funcționează **doar pentru numere întregi dintr-un interval mic și cunoscut**. Pentru prețuri reale sau pentru numere de până la 10^9, lista de frecvențe ar fi imposibil de mare sau imposibil de indexat.

5. Care metodă și când?

	Selecția minimului	Bulele (cu `schimbat`)	Listă de frecvențe
comparații	mereu `n(n − 1)/2`	între `n − 1` și `n(n − 1)/2`	niciuna
interschimbări	cel mult `n − 1`	câte perechi inversate există	niciuna (reconstruiești lista)
memorie suplimentară	nu	nu	lista `fr` (și lista rezultat)

	Selecția minimului	Bulele (cu `schimbat`)	Listă de frecvențe
potrivită când	interschimbările sunt costisitoare; vrei doar primele `k` valori	lista e **aproape sortată**	valori întregi într-un interval mic
complexitate	$O(n^2)$	$O(n^2)$, $O(n)$ pe listă sortată	$O(n + M)$

Exemplu. Din 200 de punctaje la olimpiadă vrei doar **primele 10**, descrescător. Nu e nevoie de sortarea completă: oprești selecția (a **maximului**, acum) după 10 pași, sau bulele după 10 treceri. Ambele fac `199 + 198 + … + 190 = 1945` comparații, dar selecția face cel mult 10 interschimbări, pe când bulele, pe date aleatoare, au făcut în testele noastre în jur de 1750–1800.

6. Și sort()?

```
v = [4, 1, 3]
print(sorted(v, reverse=True), v)                    # [4, 3, 1] [4, 1, 3]
print(sorted(["Mara", "Al", "Cristina", "Ion"], key=len))
# ['Al', 'Ion', 'Mara', 'Cristina']
```

`sort()` și `sorted()` folosesc un algoritm mult mai rapid ($O(n \log n)$), scris și optimizat de autorii limbajului. **În programele obișnuite îl folosești pe el.** Metodele tale le scrii când problema cere explicit, când vrei doar o parte din rezultat (primele `k`) sau când datele permit ceva mai rapid (frecvențele). Indiferent de metodă, testezi pe: listă vidă, un element, listă deja sortată, listă inversă, valori egale.

> **DE ȘTIUT**
>
> **Opțional, în afara programei:** a patra metodă clasică, *sortarea prin inserție*, se poate vedea ca aranjarea cărților de joc din mână. Păstrezi prefixul din stânga sortat și iei pe rând următorul element `x`: muți spre dreapta elementele mai mari decât el și îl pui în golul rămas.
>
> ```
> v = [5, 2, 4, 1]
> for i in range(1, len(v)):
>     x = v[i]
>     j = i - 1
>     while j >= 0 and v[j] > x:
>         v[j + 1] = v[j]
>         j -= 1
>     v[j + 1] = x          # după pasul i = 1, 2, 3: [2, 5, 4, 1], [2, 4, 5, 1], [1, 2, 4, 5]
> ```
>
> Este tot $O(n^2)$ în cazul cel mai rău, dar pe o listă deja sortată face doar `n - 1` comparații, ca bulele cu `schimbat`. La Bac nu se cere; o menționăm doar pentru comparație.

7. Verifică-te

1. Cum arată lista `[9, 4, 7, 1]` după **primul** pas al selecției minimului? Dar după prima trecere a metodei bulelor?

2. Câte comparații face selecția minimului pentru o listă de 8 elemente deja sortată?
3. Câte interschimbări face metoda bulelor pe `[2, 1, 4, 3]`? Câte treceri, cu `schimbat`?
4. Poți sorta cu listă de frecvențe temperaturile `[-3, 5, 0, -1, 2]`? Ce trebuie să modifici?
5. Care metodă e cea mai potrivită pentru 50 000 de cereri de mentenanță codificate 1–8? Dar pentru o listă de 30 de prețuri reale, aproape ordonată?
6. După ce trecere a metodei bulelor ești sigur că maximul este pe ultima poziție? De ce?

Răspunsuri

1. Selecție: `[1, 4, 7, 9]` (1 ↔ 9). Bule: `[4, 7, 1, 9]`.
2. `7 + 6 + … + 1 = 28`.
3. 2 interschimbări (`2 ↔ 1`, `4 ↔ 3`); 2 treceri (a doua nu mai schimbă nimic și oprește algoritmul).
4. Da, dacă **decalezi** valorile: `fr[x + 3] += 1` (cea mai mică valoare, −3, ajunge pe poziția 0), iar la reconstruire scrii `v − 3`.
5. Cererile: listă de frecvențe (multe valori, doar 8 coduri). Prețurile: metoda bulelor cu `schimbat` (puține treceri pe o listă aproape sortată), sau pur și simplu `sort()`.
6. După prima: maximul este „împins" de fiecare comparație cu vecinul din dreapta, până la capăt.

Probleme de lucru în clasă (pbinfo)

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

- [ ] `#509` **ordonare** — ușoară
  https://www.pbinfo.ro/probleme/509/ordonare
- [ ] `#244` **cifreord** — ușoară
  https://www.pbinfo.ro/probleme/244/cifreord
- [ ] `#129` **sortare** — ușoară
  https://www.pbinfo.ro/probleme/129/sortare

Temă pentru acasă (pbinfo)

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

- [ ] `#130` **sortcif** — ușoară
  https://www.pbinfo.ro/probleme/130/sortcif
- [ ] `#511` **ksort** — ușoară
  https://www.pbinfo.ro/probleme/511/ksort
- [ ] `#183` **sortpp** — medie
  https://www.pbinfo.ro/probleme/183/sortpp