INFORMATICĂ · PYTHON · CLASA A VIII-A · TABLOURI UNIDIMENSIONALE · LECȚIA 11

Sortarea unui șir: selecție și bule

Durată: 50 min Nivel: gimnaziu · Python Înainte: lecțiile py8g-10

CE VEI ȘTI DUPĂ LECȚIE

- Înțelegi ideea sortării prin selecție și a sortării cu bule.
- Urmărești, pas cu pas, cum se schimbă lista.
- Sortezi un șir crescător și descrescător.

1. A sorta înseamnă a pune în ordine

Ai cinci cărți de joc și vrei să le ordonezi de la cea mai mică la cea mai mare. Există mai multe metode. Două dintre ele sunt ușor de înțeles: **sortarea prin selecție** și **sortarea cu bule**. În ambele folosim ce știm deja: *minimul*, *compararea vecinilor* și *interschimbarea*.

Lucrăm cu șirul `5 2 9 1 6`.

2. Sortarea prin selecție

Ideea: caută cel mai mic element din tot șirul și pune-l pe prima poziție. Apoi caută cel mai mic din **restul** șirului și pune-l pe a doua poziție. Și tot așa.

```
v = [5, 2, 9, 1, 6]
n = len(v)
for i in range(n - 1):
    poz = i
    for j in range(i + 1, n):
        if v[j] < v[poz]:
            poz = j
    v[i], v[poz] = v[poz], v[i]
print(*v)
```

```
1 2 5 6 9
```

Bucla de la `j` găsește poziția minimului din restul listei. Apoi îl interschimbăm cu elementul de pe poziția `i`. Iată ce se întâmplă:

Pasul `i`	Minimul din restul listei	Lista după interschimbare
0	`1` (poziția 3)	`1 2 9 5 6`
1	`2` (poziția 1)	`1 2 9 5 6` (era deja bine)
2	`5` (poziția 3)	`1 2 5 9 6`
3	`6` (poziția 4)	`1 2 5 6 9`

După pasul `i`, primele `i + 1` poziții sunt **gata** și nu se mai ating.

3. Sortarea cu bule

Ideea: compară vecinii, doi câte doi. Dacă sunt în ordine greșită, îi schimbi. Cel mai mare element „iese la suprafață" ca o bulă și ajunge la capăt. Repeți pentru restul șirului.

```
v = [5, 2, 9, 1, 6]
n = len(v)
for pas in range(n - 1):
    for i in range(n - 1 - pas):
        if v[i] > v[i + 1]:
            v[i], v[i + 1] = v[i + 1], v[i]
print(*v)
```

```
1 2 5 6 9
```

După fiecare trecere, încă un element mare își găsește locul definitiv la sfârșit:

Trecerea	Lista la sfârșitul trecerii
0	`2 5 1 6 9` (9 a ajuns la capăt)
1	`2 1 5 6 9` (6 a ajuns pe loc)
2	`1 2 5 6 9`
3	`1 2 5 6 9` (nicio schimbare)

În trecerea 0, bula `9` a „urcat": `5` cu `2` se schimbă, `5` cu `9` rămân, `9` cu `1` se schimbă, `9` cu `6` se schimbă.

4. Sortare descrescătoare

Vrei de la cel mai mare la cel mai mic? Schimbi **un singur semn**: la bule, `v[i] < v[i + 1]`, iar la selecție cauți maximul (`v[j] > v[poz]`).

```
v = [5, 2, 9, 1, 6]
n = len(v)
for pas in range(n - 1):
    for i in range(n - 1 - pas):
        if v[i] < v[i + 1]:
            v[i], v[i + 1] = v[i + 1], v[i]
print(*v)
```

```
9 6 5 2 1
```

SFAT

Scurtătură (opțional). Python are `v.sort()` (crescător) și `v.sort(reverse=True)` (descrescător). Au același rezultat. Ca să înțelegi ce face calculatorul, trasează pe hârtie unul dintre algoritmii de mai sus.

5. Care e mai bun?

Pentru liste mici, oricare merge. Pentru liste foarte mari, amândoi sunt lenți: fac aproximativ `n · n` comparații. Mai târziu (în liceu) vei întâlni metode mult mai rapide.

> **ATENȚIE — GREȘELI FRECVENTE**
>
> Nu confunda: bucla exterioară ține **cât de mult s-a sortat**, iar cea interioară face comparațiile. Dacă dai alte limite la `range`, apare `IndexError` sau rămân elemente nesortate.

6. Verifică-te

1. Cum arată `3 1 2` după prima trecere a sortării cu bule?
2. După primul pas al selecției, ce se află pe poziția 0?
3. Ce schimbi ca să sortezi descrescător?
4. Trasează selecția pentru `4 3 1`. (Scrie lista după fiecare pas.)
5. De ce trebuie interschimbarea de la selecție să folosească `v[i], v[poz] = v[poz], v[i]`?

Răspunsuri

1. `1 2 3`: (3,1) se schimbă → `1 3 2`; (3,2) se schimbă → `1 2 3`.
2. Cel mai mic element din tot șirul.
3. Sensul comparației (`<` în loc de `>` la bule, maximul în loc de minim la selecție).
4. `i = 0`: minimul este 1 → `1 3 4`; `i = 1`: minimul din `3 4` este 3 → `1 3 4`.
5. Ca să nu pierdem valoarea de pe poziția `i`: schimbăm două valori, nu le copiem pe una peste alta (lecția py8g-08).
