INFORMATICĂ · CLASA A X-A · TABLOURI UNIDIMENSIONALE · LECȚIA 5

Căutarea secvențială, căutarea binară și interclasarea

Durată: 60 min Nivel: toate profilurile Înainte: lecțiile x-04

CE VEI ȘTI DUPĂ LECȚIE

- Cauți o valoare într-un vector oarecare (secvențial) și într-un vector sortat (binar).
- Înțelegi de ce căutarea binară face doar ~$\log_2 n$ pași.
- Găsești prima poziție cu a[i] ≥ x (varianta de căutare binară cea mai utilă).
- Interclasezi doi vectori sortați într-un al treilea, în timp liniar.
- Calculezi reuniunea și intersecția a două mulțimi sortate.

1. Căutarea secvențială

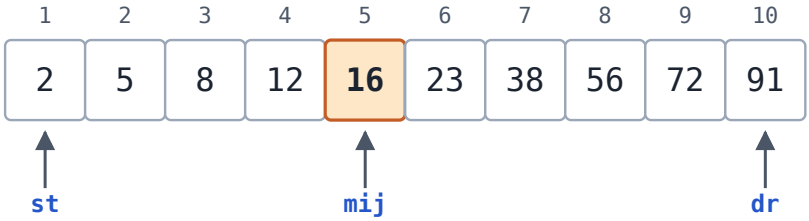
Parcurgem vectorul până găsim valoarea:

```
int poz = -1;
for (int i = 1; i <= n && poz == -1; i++)
    if (a[i] == x) poz = i;
```

În cel mai rău caz verificăm toate cele n elemente: **O(n)**.

2. Căutarea binară (doar pe vector sortat!)

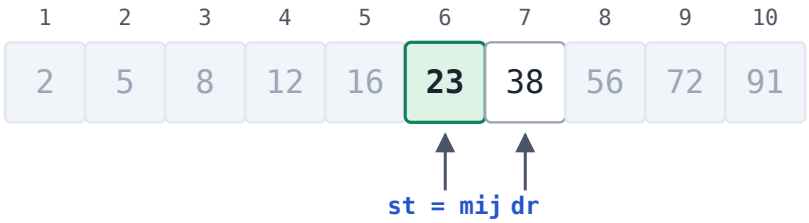
Ca la ghicitul unui număr între 1 și 100: „e mai mare sau mai mic?”. La fiecare întrebare înjumătățim intervalul. Cauți un cuvânt în dicționar: deschizi la mijloc, vezi dacă ești înainte sau după, și continui pe jumătatea potrivită.



Pasul 1: st = 1, dr = 10, mij = 5 → a[5] = 16 < 23, deci st = 6



Pasul 2: st = 6, dr = 10, mij = 8 → a[8] = 56 > 23, deci dr = 7



Pasul 3: st = 6, dr = 7, mij = 6 → a[6] = 23: găsit!

Căutarea binară

```cpp
#include <iostream>
using namespace std;

int a[100001], n, x;

int main()
{
    cin >> n >> x;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    int st = 1, dr = n, poz = -1;
    while (st <= dr && poz == -1)
    {
        int mij = (st + dr) / 2;
        if (a[mij] == x) poz = mij;
        else if (a[mij] < x) st = mij + 1;   // x e în dreapta
        else dr = mij - 1;                   // x e în stânga
    }
    cout << poz;
    return 0;
}
```

Pentru n = 1 000 000 face cel mult **20** de pași ($2^{20} \approx 10^6$). Complexitate: **O(log n)**.

ATENȚIE — GREȘELI FRECVENTE

Căutarea binară funcționează **doar** pe vectori sortați. Pe un vector nesortat dă rezultate greșite fără niciun avertisment.

DE CE AȘA?

De ce `st = mij + 1` și `dr = mij - 1`, nu `st = mij`? Elementul `mij` a fost deja verificat și nu e `x`; îl excludem. Cu `st = mij` intervalul ar putea să nu se micșoreze (când `st + 1 == dr`, `mij == st`), iar bucla ar fi infinită.

Urmărire

`a = (2, 5, 8, 12, 16, 23, 38, 56, 72, 91)`, căutăm `x = 23`:

st	dr	mij	a[mij]	decizie
1	10	5	16	16 < 23 → st = 6
6	10	8	56	56 > 23 → dr = 7
6	7	6	23	găsit la poziția 6

Pentru `x = 40` (nu există):

st	dr	mij	a[mij]	decizie
1	10	5	16	st = 6
6	10	8	56	dr = 7
6	7	6	23	st = 7
7	7	7	38	st = 8
8	7	—	`st > dr` → nu există	

LA BAC

La grile se întreabă des: „Ce valori ia `mij`?" sau „Câte comparații se fac până se găsește/nu se găsește x?". Fă tabelul ca mai sus.

Varianta „prima poziție cu a[i] ≥ x"

Mai utilă decât căutarea exactă: răspunde la „câte elemente sunt mai mici decât x?", „unde aș insera x?".

```cpp
#include <iostream>
using namespace std;

int a[100001], n, x;

int main()
{
    cin >> n >> x;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    int st = 1, dr = n, rez = n + 1;     // n + 1 = „nu există”
    while (st <= dr)
    {
        int mij = (st + dr) / 2;
        if (a[mij] >= x) { rez = mij; dr = mij - 1; }   // e candidat, căutăm mai la stânga
        else st = mij + 1;
    }
    cout << rez;       // elementele mai mici decât x sunt exact a[1..rez-1]
    return 0;
}
```

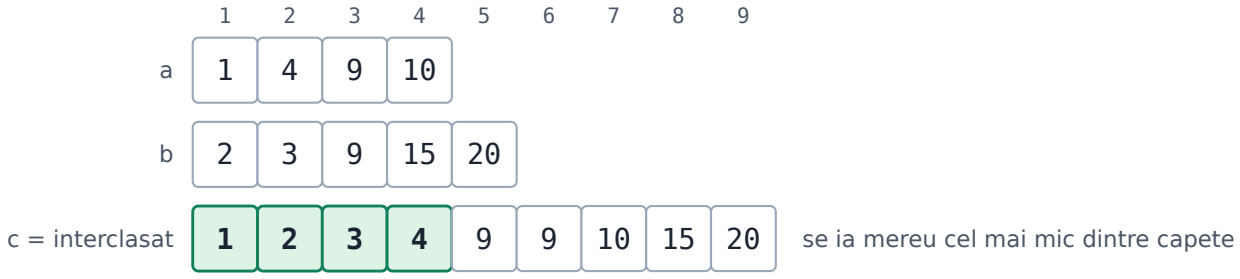
Câte elemente sunt în intervalul `[x, y]` ? → (prima poziție cu `a[i] > y`) − (prima poziție cu `a[i] ≥ x`).

DE CE AȘA?

De ce nu ne oprim când găsim `x` ? Pot exista mai multe copii (de exemplu `5 5`); noi vrem prima. Reținem `rez = mij` și continuăm în stânga, căutând o apariție și mai de la început. Ideea „păstrez un candidat bun și caut unul mai bun” se numește căutare binară pe răspuns.

3. Interclasarea

Avem doi vectori **sortați** `a` (n elemente) și `b` (m elemente). Vrem un vector `c` sortat cu toate elementele. Ideea: comparăm „capetele” și îl luăm pe cel mai mic, ca la două cozi care se îmbină într-una singură.



Interclasare

```cpp
#include <iostream>
using namespace std;

int a[1001], b[1001], c[2002], n, m;

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int j = 1; j <= m; j++) cin >> b[j];
    int i = 1, j = 1, k = 0;
    while (i <= n && j <= m)
        if (a[i] < b[j]) c[++k] = a[i++];
        else c[++k] = b[j++];
    while (i <= n) c[++k] = a[i++];      // ce a rămas din a
    while (j <= m) c[++k] = b[j++];      // ce a rămas din b
    for (int t = 1; t <= k; t++)
        cout << c[t] << ' ';
    return 0;
}
```

a[i]	b[j]	se ia	c
1	2	1 (din a)	1
4	2	2 (din b)	1 2
4	3	3 (din b)	1 2 3
4	9	4 (din a)	1 2 3 4
9	9	9 (din b, egal)	1 2 3 4 9
9	15	9 (din a)	1 2 3 4 9 9
10	15	10 (din a), a se termină	1 2 3 4 9 9 10
—	15, 20	copiez restul din b	1 2 3 4 9 9 10 15 20

Complexitate: **O(n + m)**: mult mai bine decât „le pun pe toate și sortez” (O((n+m)²) cu bule).

SFAT

`c[++k] = a[i++];` face trei lucruri: crește k, copiază `a[i]` în `c[k]`, apoi crește i. E o scriere compactă și foarte des folosită: asigură-te că o înțelegi.

DE CE AȘA?

De ce cele două bucle de la final? Când unul dintre vectori se termină, celălalt mai poate avea elemente, toate mai mari decât tot ce am scris. Doar unul din cele două `while` se execută efectiv; celălalt are condiția falsă din start.

Reuniune și intersecție (mulțimi sortate, fără duplicate)

```
// intersecția: doar elementele comune
int i = 1, j = 1, k = 0;
while (i <= n && j <= m)
    if (a[i] == b[j]) { c[++k] = a[i]; i++; j++; }
    else if (a[i] < b[j]) i++;
    else j++;
```

Pentru reuniune, la egalitate pui elementul o singură dată și avansezi în ambii vectori.

4. Interclasare fără memorie (Bac III.3)

Dacă cele două șiruri sortate sunt în fișiere și sunt foarte mari, poți interclasa **citind câte un element din fiecare** și scriind direct rezultatul, fără vectori. Același algoritm, dar `a[i]` devine „valoarea citită curent din primul fișier".

5. Greșeli frecvente

Greșeala	Efect	Corect
căutare binară pe vector nesortat	rezultate greșite	sortează întâi
`while (st < dr)` în loc de `st <= dr`	pierzi cazul cu un element	`st <= dr`
`st = mij`	buclă infinită	`st = mij + 1`
uiți cele două bucle de la interclasare	pierzi elementele rămase	copiază restul
`(st + dr) / 2` cu valori foarte mari	depășire de `int`	`st + (dr - st) / 2`

6. La BAC

> **LA BAC**
>
> Grilele de căutare binară cer **valorile lui `mij`** sau **numărul de pași** (tabelul din lecție). Cele de interclasare cer vectorul rezultat sau al câtelea element e un anumit număr. La II.3 poți fi pus să scrii interclasarea; la III.3, interclasarea din fișiere fără vectori.

7. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Câți pași (valori ale lui `mij`) face căutarea binară ca să constate că 40 **nu** apare în vectorul din secțiunea 2?

2. Interclasează (pe hârtie) `(1, 4, 9, 10)` cu `(2, 3, 9, 15, 20)`.

3. Se dau doi vectori sortați. Afișează elementele din `a` care **nu** apar în `b` (diferența), în O(n + m).

4. Care este numărul maxim de pași ai căutării binare pentru n = 1000? Dar pentru n = 10^6?

5. Ce rezultat obții dacă aplici căutarea binară pentru `x = 5` pe vectorul nesortat `(5, 1, 3, 9)`?

6. În vectorul sortat `(1, 3, 5, 5, 7, 9)`, câte elemente sunt în intervalul `[3, 7]`? Folosește „prima poziție ≥ x" de două ori.

7. De ce se scrie `st + (dr - st) / 2` în loc de `(st + dr) / 2` pentru indici mari?

8. Mai departe (pentru cine vrea mai mult)

`lower_bound` și `upper_bound`. În C++ standard, `lower_bound(a+1, a+n+1, x) - a` dă exact „prima poziție cu `a[i] ≥ x`", iar `upper_bound` pe „prima poziție cu `a[i] > x`". Acum știi ce fac înăuntru.

Căutare binară pe răspuns. Nu doar în vectori: dacă răspunsul unei probleme are proprietatea „dacă x merge, atunci și x+1 merge", îl cauți binar printre numere. Exemplu: „care e cea mai mică viteză cu care ajungi la timp?". Este o tehnică des folosită la concursuri.

Căutare binară la depanare. `git bisect` caută binar versiunea de cod care a introdus un bug: dintre 1000 de versiuni, 10 teste. Aceeași idee, aplicată la istoricul unui proiect.

Merge sort. Interclasarea este cheia sortării prin interclasare: împarți vectorul în două, sortezi fiecare jumătate și le interclasezi. Timp O(n log n), garantat; o implementezi la recursivitate (clasa a XI-a).

Rezumat

- Secvențial: O(n), merge pe orice vector.
- Binar: O(log n), doar pe vector sortat; `st`, `dr`, `mij`.
- Prima poziție ≥ x: reții `rez = mij` și cauți mai la stânga.
- Interclasare: compari capetele, iei minimul, copiezi restul. O(n + m).

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] **#508** **cautare-binara** — medie
  Pentru fiecare y[j], există în x?
  https://www.pbinfo.ro/probleme/508/cautare-binara
- [ ] **#4856** **cb-0** — ușoară
  https://www.pbinfo.ro/probleme/4856/cb-0
- [ ] **#241** **interclasare** — ușoară
  https://www.pbinfo.ro/probleme/241/interclasare

Temă (pbinfo)

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

- [ ] #2276 **cb** — ușoară
  Câte elemente sunt într-un interval: două căutări binare.
  https://www.pbinfo.ro/probleme/2276/cb
- [ ] #250 **interclasare1** — medie
  https://www.pbinfo.ro/probleme/250/interclasare1
- [ ] #251 **interclasare2** — medie
  https://www.pbinfo.ro/probleme/251/interclasare2
- [ ] #530 **multimi1** — medie
  Reuniune și intersecție.
  https://www.pbinfo.ro/probleme/530/multimi1
- [ ] #284 **interclasare3** — medie
  https://www.pbinfo.ro/probleme/284/interclasare3
- [ ] #507 **cautare** — ușoară
  https://www.pbinfo.ro/probleme/507/cautare

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #3974 **irdds** — medie
  https://www.pbinfo.ro/probleme/3974/irdds
- [ ] #2668 **comun** — medie
  https://www.pbinfo.ro/probleme/2668/comun

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #1051 **Exercițiul #1051**
  căutare binară în tablou unidimensional (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1051
- [ ] #1193 **Exercițiul #1193**
  căutare binară pentru x = 48 (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1193
- [ ] #1204 **Exercițiul #1204**
  căutare binară în (3,7,12,15,17,22,30) (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1204

- [ ] #1217 **Exercițiul #1217**
  căutare binară în (2,8,13,19,20,38,47) (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1217
- [ ] #1364 **Exercițiul #1364**
  căutare binară în (7,9,11,14,16,18,20) (Bac 2022)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1364
- [ ] #1415 **Exercițiul #1415**
  căutare binară în tablou de ani (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1415
- [ ] #1690 **Exercițiul #1690**
  căutare binară, valori succesive ale lui mij (Bac 2025)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1690
- [ ] #816 **Exercițiul #816**
  interclasarea lui A=(1,7,9) și B=(5,7,8,12) (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=816
- [ ] #822 **Exercițiul #822**
  interclasare A=(20,17,12,10,3) cu B (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=822
- [ ] #1050 **Exercițiul #1050**
  interclasare A și B (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1050
- [ ] #1228 **Exercițiul #1228**
  interclasare A=(5,7,15,25,33) cu B (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1228
- [ ] #1270 **Exercițiul #1270**
  interclasare A=(1,7,10,18,32) cu B=(2,5,12,16,49) (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1270

Citește și exersează în plus

- Căutarea binară https://www.pbinfo.ro/articole/3633/cautarea-binara
- Interclasarea tablourilor https://www.pbinfo.ro/articole/5588/interclasarea-tablourilor
- VisuAlgo — Binary Search — animație interactivă; alege „Search” și apoi introdu un vector sortat https://visualgo.net/en/list
- Python Tutor pentru C++ — urmărește st, dr și mij pas cu pas https://pythontutor.com/cpp.html
- cppreference — lower_bound / upper_bound — versiunile gata scrise ale căutării binare din lecție https://en.cppreference.com/w/cpp/algorithm/lower_bound

Răspunsuri

Exersează pe hârtie

1. 4 pași: `mij = 5, 8, 6, 7` (valorile 16, 56, 23, 38), apoi `st > dr`.

2. `(1, 2, 3, 4, 9, 9, 10, 15, 20)`.

3.

```cpp
#include <iostream>
using namespace std;

int a[1001], b[1001], n, m;

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int j = 1; j <= m; j++) cin >> b[j];
    int i = 1, j = 1;
    while (i <= n)
    {
        while (j <= m && b[j] < a[i]) j++;     // sar peste elementele mai mici din b
        if (j > m || b[j] != a[i])
            cout << a[i] << ' ';               // a[i] nu apare în b
        i++;
    }
    return 0;
}
```

Fiecare indice doar crește: **O(n + m)**.

4. Pentru n = 1000: **10** pași ($2^{10} = 1024 \geq 1001$). Pentru n = 10^6: **20** de pași ($2^{20} \approx 1{,}05 \cdot 10^6$).

5. Se obține „nu există", deși 5 este în vector. Pași: `mij = 2`, `a[2] = 1 < 5` → `st = 3`; `mij = 3`, `a[3] = 3 < 5` → `st = 4`; `mij = 4`, `a[4] = 9 > 5` → `dr = 3`, apoi `st > dr`. Căutarea binară presupune vector sortat; pe unul nesortat poate rata elemente.

6. `4`: prima poziție ≥ 3 este 2, prima poziție > 7 este 6 (elementul 9), deci 6 − 2 = 4 (3, 5, 5, 7).

7. `st + dr` poate depăși valoarea maximă a lui `int` când `st` și `dr` sunt mari (de exemplu $1{,}5 \cdot 10^9 + 1{,}5 \cdot 10^9$), iar rezultatul devine negativ. `st + (dr - st) / 2` nu depășește niciodată `dr`. Bug celebru, rămas ani în biblioteca standard Java.