Organizatorice

- Re-Explicare cerinte laborator:
  - Tema 1: 10 p (Sortari)
  - Tema 3: 10 p (Structura de date complexa)
  - Tema 2: 30 p (Pentru 30p trebuie obtinute 100p din problemele impartite in 5 sectiuni)

  - Usurare un pic a laboratorului. Nota 5 cu 20p, nota 10 tot cu 50p. Practic primele 20p valoreaza 0.25 si urmeazaorele 30p valoreaza 0.1(6). Daca ai 26 puncte vei avea in loc de 5.2 -> **6** (20 *0.25 + 6 * 0.1*(6))
  - **Dar... vreau sa nu aveti probleme copiate...**
  - **Scopul e sa treaca mai usor cei care fac cu chiu cu vai problemele si mai greu cei care le copieaza.!**

B-Arbori

Ștergerea din B-Arbore

La ștergere, numărul de chei dintr-un nod scade cu 1, deci e posibil să ajungem să avem mai puțin de *t* chei. În acest caz, este nevoie de un proces de **fuziune** (invers divizării).

Trebuie să ne asigurăm că toate nodurile rămân cu cel puțin *t* chei.

În procesul de fuziune, o cheie coboară într-un fiu înainte de a i se aplica acestuia (recursiv) procesul de ștergere.

Ștergerea din B-Arbore

Dacă, în urma ștergerii, rădăcina rămâne fără nicio cheie (deci, cu un singur nod fiu), atunci acest nod rădăcină gol este eliminat, iar noua rădăcină devine unicul fiu al rădăcinei vechi. Astfel, înălțimea arborelui scade cu 1.

Ștergerea din B-Arbore

Avem trei cazuri:

1. Cheia de șters **k** este într-un nod **frunză x**. Avem 2 subcazuri:
   a) Dacă, după ștergere, nodul rămâne cu suficiente chei, atunci se șterge cheia **k**, fără nicio altă modificare.

   b) Dacă, după ștergere, nu rămân suficiente chei, atunci:
      - Încercăm să împrumutăm o cheie de la fratele din stânga.
      - Dacă nu putem (rămâne și stânga cu prea puține chei), atunci încercăm împrumutul la dreapta.
      - Dacă nu putem împrumuta nici de la stânga, nici de la dreapta, atunci aplicăm fuziunea pe unul din frați și cheia părinte corespunzătoare.

Ștergerea din B-Arbore

Avem trei cazuri:

2. Cheia de șters **k** este într-un nod **intern x**. Se disting 3 subcazuri:
   a) Dacă fiul **y** din stânga lui **k** are cel puțin *t* chei, atunci se caută predecesorul **k'** al lui **k** în subarborele de rădăcină **y**. Se șterge **k'** și se înlocuiește **k** din **x** cu **k'**. Se aplică mai departe procesul, recursiv.

   b) Analog pentru fiul **z** din dreapta lui **k**, căutându-se succesorul **k'** al lui **k**.

   c) Dacă și **y**, și **z** au *t-1* chei, se aplică fuziunea pe cele două noduri (adăugându-se și cheia **k** în **y**). În urma procesului, nodul **y** va avea *2t-1* chei. Se aplică mai departe procesul de ștergere recursivă a lui **k** din **y**.

Ștergerea din B-Arbore

Avem trei cazuri:

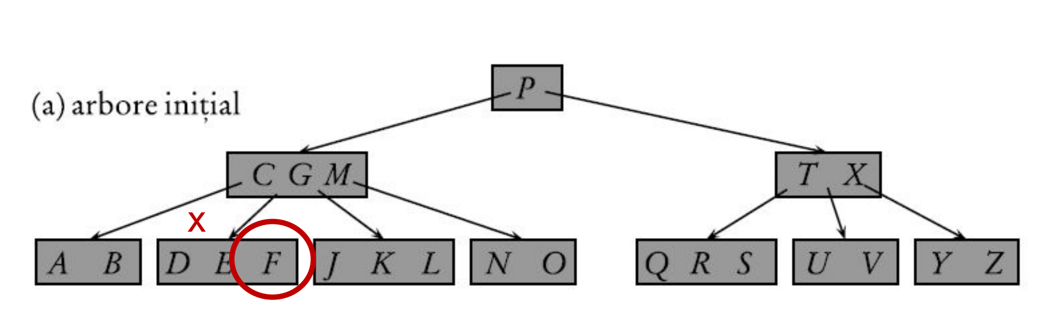
3. Cheia de șters **k** nu se găsește în nodul intern **x**
   - Determinăm rădăcina **r'** (fiu al lui **x**) care indică subarborele în care se găsește **k**.

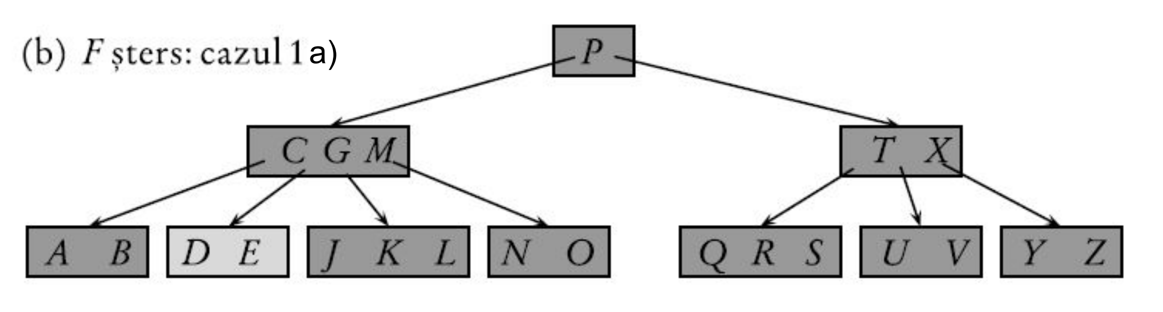
a) Dacă **r'** are *t-1* chei, dar are un frate în stânga sau în dreapta care are *t* chei, atunci mutăm o cheie din **x** în **r'**, apoi mutăm o cheie din unul din cei 2 frați înapoi în **x**.

b) Dacă **r'** și cei 2 frați din stânga și din dreapta au câte *t-1* chei, aplicăm procesul de fuziune pe **r'** și unul din cei 2 frați. Se mută, de asemenea, o cheie din **x** în noul nod rezultat, ca și cheie mediană.

Ștergerea din B-Arbore

Exemplu (t = 3):

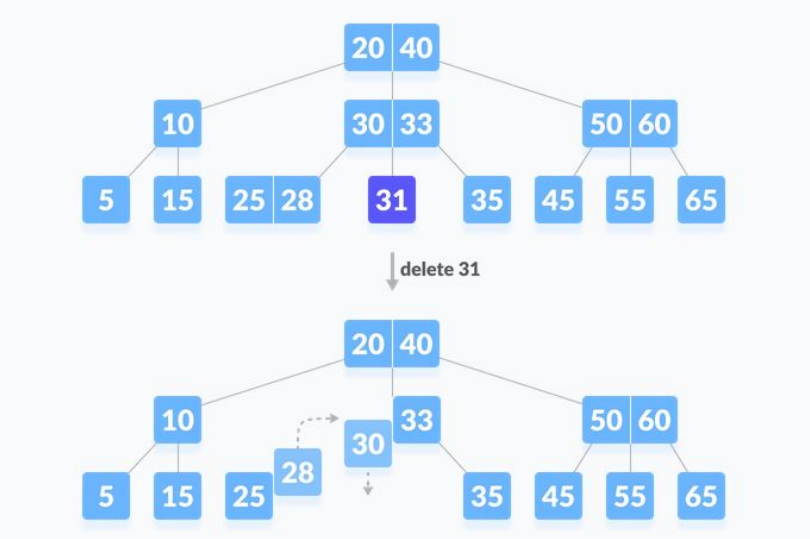


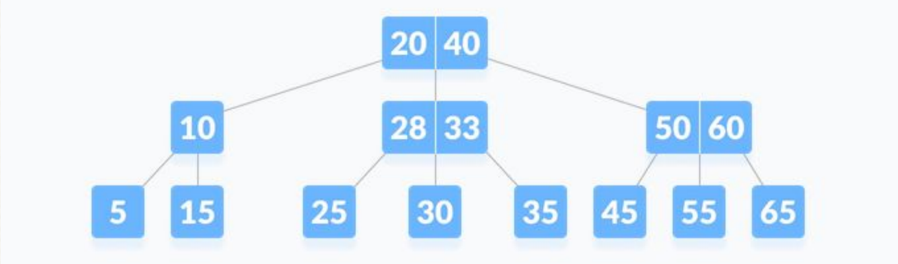


Cheia de șters se află în frunza **x** și putem șterge ușor

Ștergerea din B-Arbore

Exemplu (t = 2):

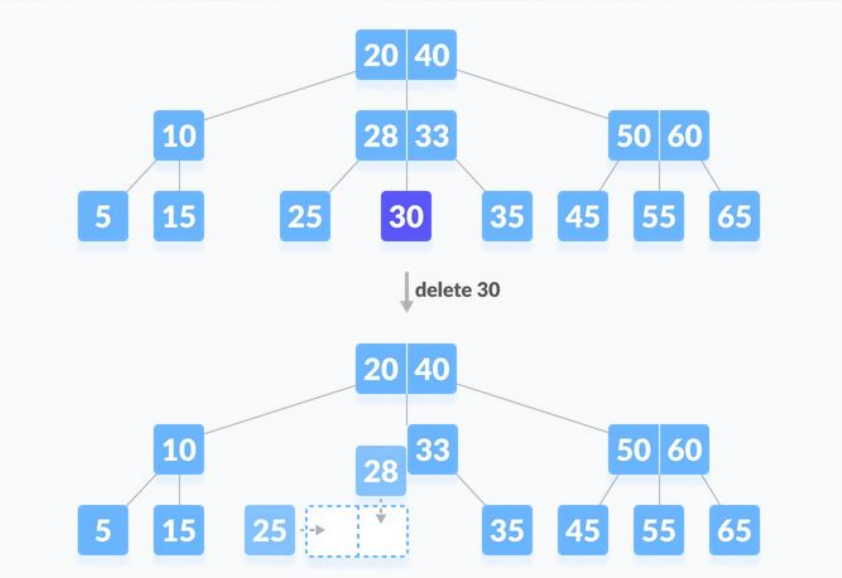


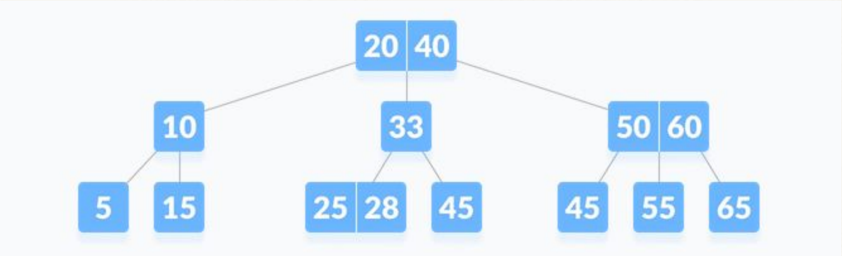


Cheia de șters se află în frunză
și putem împrumuta din stânga

Ștergerea din B-Arbore

Exemplu (t = 2):

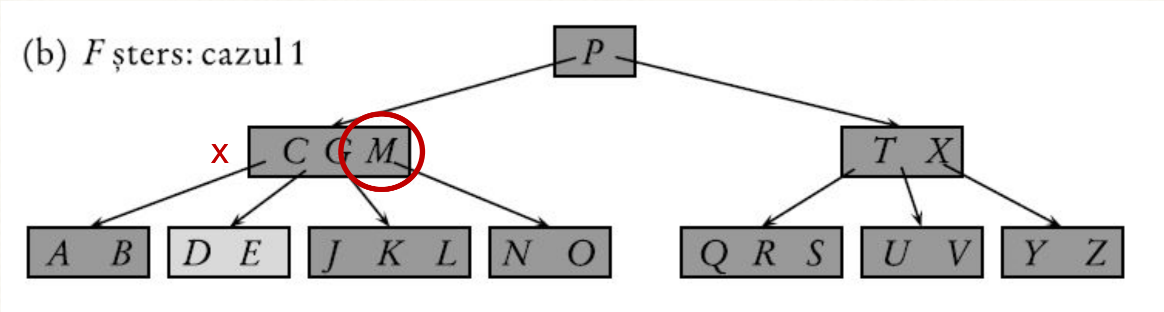




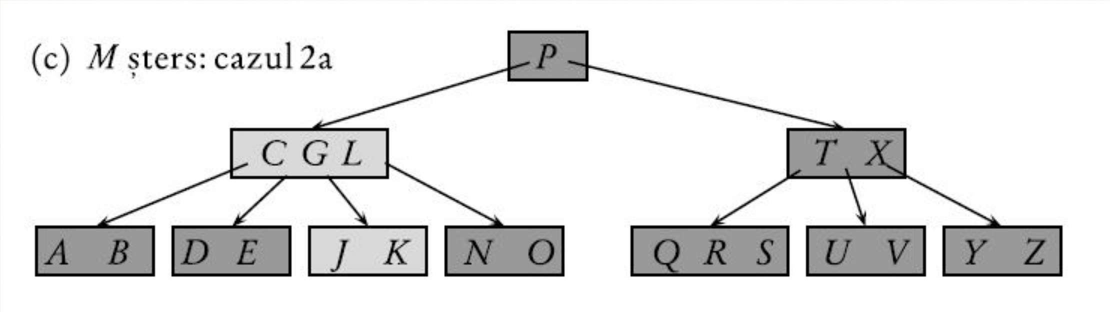
Cheia de șters se află în frunză și **nu** putem împrumuta de nicăieri

Ștergerea din B-Arbore

Exemplu (t = 3):



(b) *F* șters: cazul 1

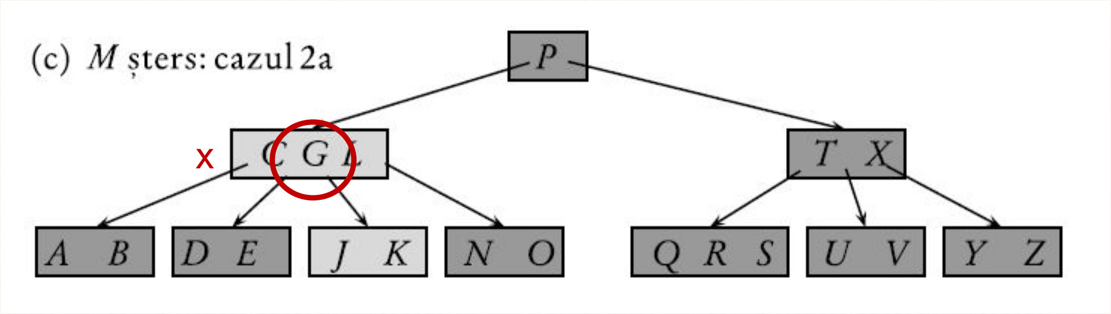


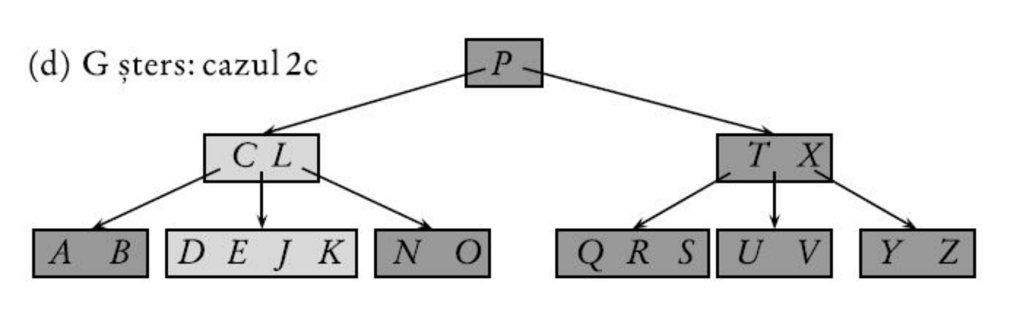
(c) *M* șters: cazul 2a

Cheia de șters se află în nodul intern **x** ȘI fiul stâng **y** are cel puțin *t* chei

Ștergerea din B-Arbore

Exemplu (t = 3):

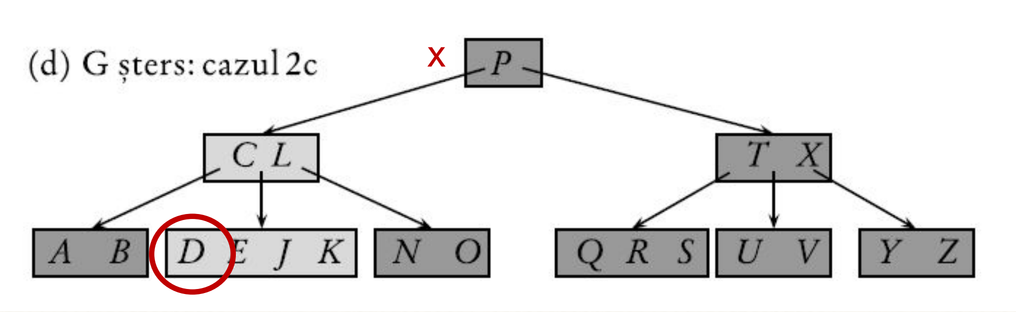


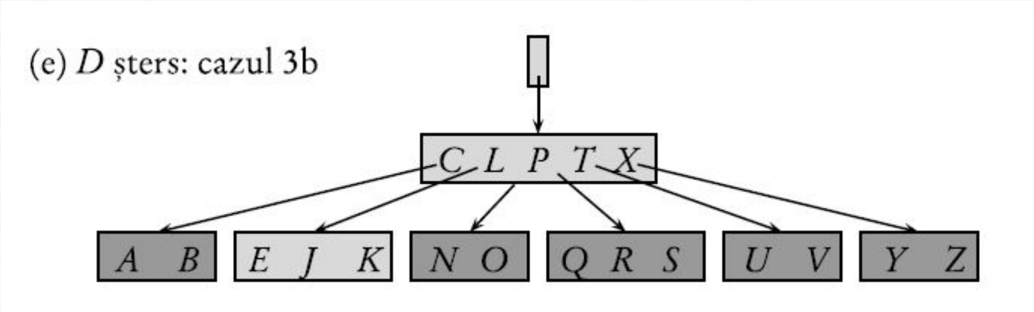


Cheia de șters se află în nodul intern **x** ȘI cei 2 fii **y** și **z** au *t-1* chei

Ștergerea din B-Arbore

Exemplu (t = 3):

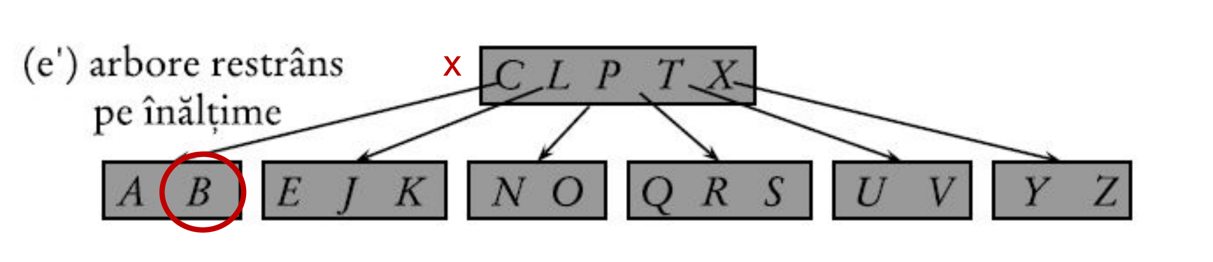


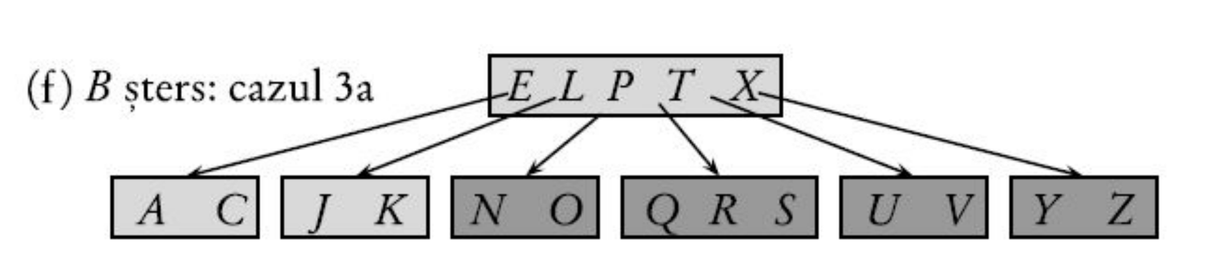


Cheia de șters NU se află în nodul intern **x**
ȘI
r’ și frații lui au *t-1* chei

Ștergerea din B-Arbore

Exemplu (t = 3):





Cheia de șters **NU** se află în nodul intern **x**
ȘI
r' are *t-1* chei și un frate cu *t* chei

Ștergerea din B-Arbore

Complexitate:

Procedura de ștergere într-un B-Arbore are loc descendent și fără reveniri.
La ștergerea unei chei dintr-un nod intern, are loc o serie de înlocuiri, pentru ca ștergerea efectivă să se realizeze în frunză.

Parcurgerea unui B-Arbore descendent fără reveniri: **O(h)**
Operații pe nivel: **O(t)**

Complexitate finală:

$O(t * h) = \mathbf{O(t \log_t n)}$

Exerciții

I. Căutare
   1. Cum se poate găsi cheia minimă din arbore? Cum putem găsi succesorul / predecesorul unei chei?

II. Inserare
   1. Arătați pașii intermediari și rezultatul inserării cheilor V, D, I, R, B, G, M, C, U, P, S, L, Y, E, T, W, J, Z, O, H, K, în această ordine, într-un B-Arbore inițial vid.
   2. Inserăm cheile {1, 2, ..., n} într-un B-Arbore inițial vid. Gradul arborelui este 2. Câte noduri va avea în final acest arbore?

III. Ștergere
   1. Plecând de la ultima configurație calculată în curs la algoritmul de ștergere, prezentați rezultatele obținute prin eliminarea, în ordine, a cheilor C, P, V.

Bibliografie

Cormen – Introducere în algoritmi, Ediția 3
Ion Ivan – Arbori B
Kerttu Pollari-Malmi – $B^{\pm}$-trees
Stanford University - Balanced Trees
https://infolab.usc.edu/csci585/Spring2010/den_ar/indexing.pdf
https://en.wikipedia.org/wiki/B-tree
http://www.cs.cornell.edu/courses/cs312/2008sp/recitations/rec25.html
https://www.programiz.com/dsa/deletion-from-a-b-tree

Arbori de intervale

Arbori de Intervale

Problemă. Se dă un vector cu n numere și operații de genul:

- Adăugăm la poziția i valoarea x (x poate fi și negativ)
- Cerem maximul pe intervalul i, j (ex 3 6)

0	1	2	3	4	5	6	7	8
3	9	2	**5**	**7**	**34**	**6**	11	8

Cum putem face asta?

Șmenul lui Batog

Problemă. Se dă un vector cu n numere și operații de genul:

- Adăugăm la poziția i valoarea x (x poate fi și negativ)
- Cerem minimul pe intervalul i, j (ex 3 6)

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
9			**34**			**11**		

Împărțim vectorul în zone de L (?) și calculăm minimul pe fiecare zonă în parte.

Șmenul lui Batog

Problemă. Se dă un vector cu n numere și operații de genul:

- Adăugăm la poziția i valoarea x (x poate fi și negativ)
    - Pentru că, dacă facem maximul mai mic, trebuie să găsim noul maxim **O(sqrt(n))**
- Cerem maximul pe intervalul i, j (ex 3 6)

0	**1**	**2**	3	4	5	**6**	**7**	8
3	**9**	**2**	5	7	3	**6**	**11**	8
9			**7**			11		

Cum răspundem la 0, 8? Dar la 0, 4? Dar la 1, 7 ?

Care este complexitatea ?

Șmenul lui Batog

Complexitate query: Împărțim în n/L zone de lungime L

O(n/L(nr de zone) + 2 * L(2 zone le pot itera aproape complet)) → L = sqrt(n)

O (sqrt(n) + 2 * sqrt(n)) = **O(sqrt(n))**

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>3</td><td>9</td><td>2</td><td>5</td><td>7</td><td>3</td><td>6</td><td>11</td><td>8</td></tr>
<tr><td colspan="3">9</td><td colspan="3">7</td><td colspan="3">11</td></tr>
</table>

Șmenul lui Batog

Împărțim în zone de:

- sqrt(n) sau…
- sqrt(n)/2
- sqrt(n) * 2 .. și
- Variațiuni
- De ce?
  - Pentru că, în practică, nu sqrt(n) va fi cel mai rapid. Totuși, sqrt(n) este o alegere buna în general.

Șmenul lui Batog

Problemă. Se dă un vector cu n numere. Sortați-l!

problemă: https://leetcode.com/problems/sort-an-array/submissions/

cod: https://pastebin.com/bFHYephh

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>3</td><td>9</td><td>50001</td><td>5</td><td>7</td><td>34</td><td>6</td><td>11</td><td>8</td></tr>
<tr><td colspan="3">3</td><td colspan="3">5</td><td colspan="3">6</td></tr>
</table>

Arbori de Intervale

Problemă. Se dă un vector cu n numere și operații de genul:

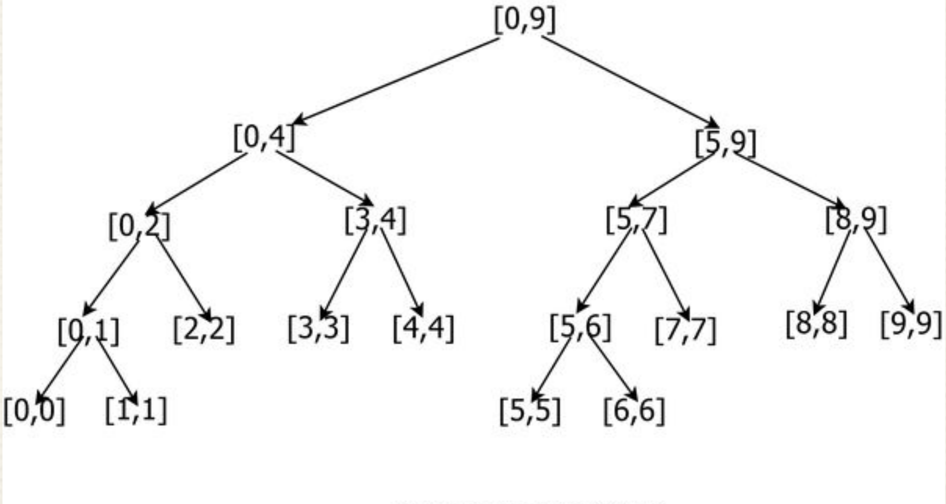
- Adăugăm la poziția i valoarea x (x poate fi și negativ)
- Cerem minimul pe intervalul i, j (ex 3 6)

0	1	2	3	4	5	6	7	8	9
3	9	2	**5**	**7**	**34**	**6**	11	8	44

Arbori de Intervale

Arbore cu rădăcina ținând intervalul [0,n)

Pentru un nod ce ține intervalul [L, R] → fiul stâng ține [L, (L+R)/2], cel drept [(L+R)/2, R]



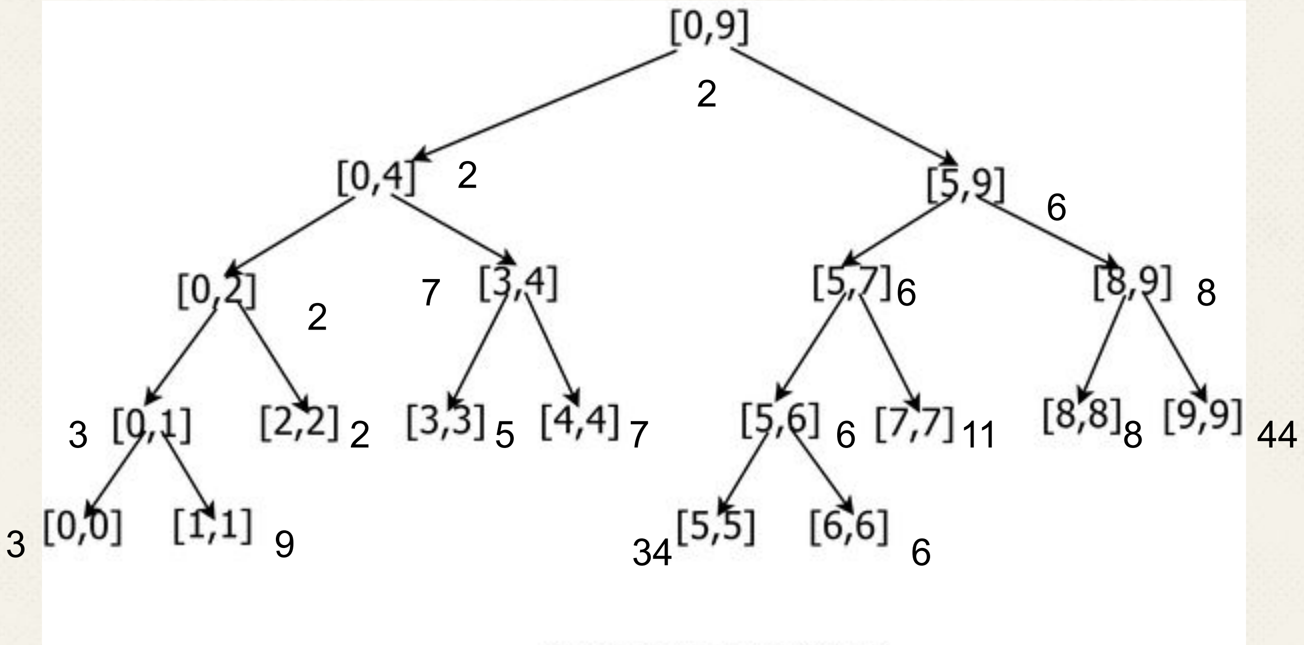
Example: Discrete Segment Tree, Yeming Hu

0	1	2	3	4	5	6	7	8	9
3	9	2	**5**	**7**	**34**	**6**	11	8	44

Arbori de Intervale

0	1	2	3	4	5	6	7	8	9
3	9	2	**5**	**7**	**34**	**6**	11	8	44

Ținem minimul!

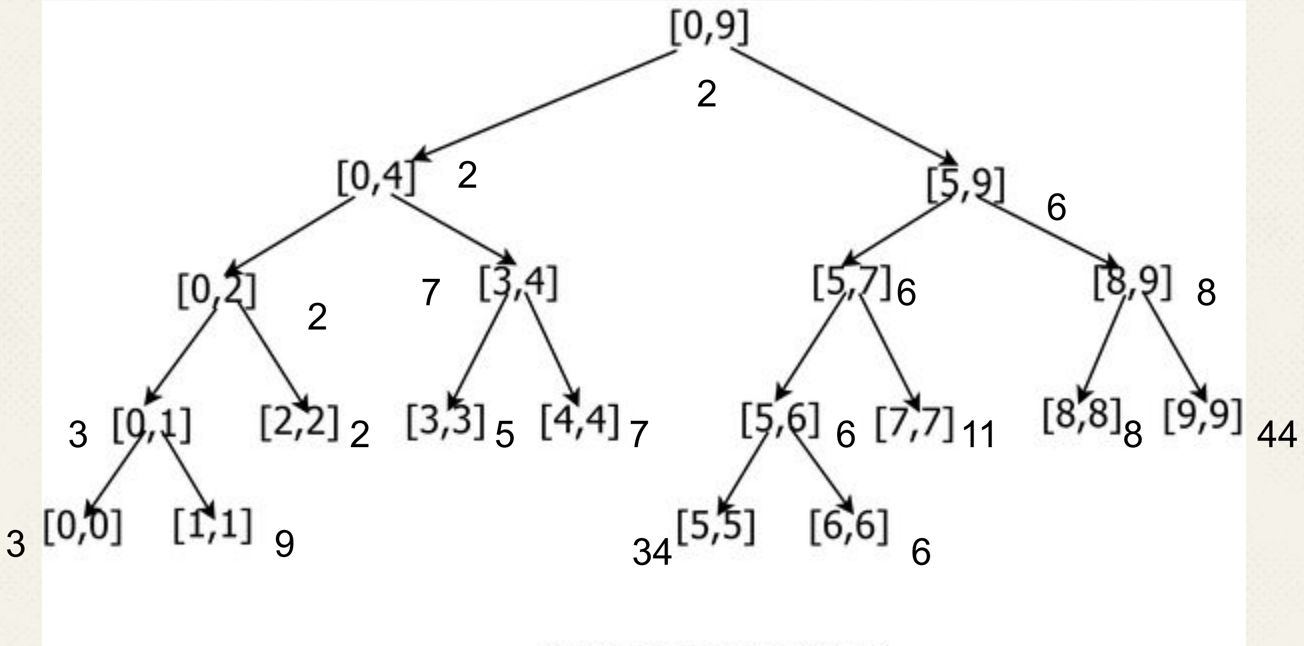


Example: Discrete Segment Tree, Yeming Hu

Arbori de Intervale

0	1	2	3	4	5	6	7	8	9
3	9	2	**5**	**7**	**34**	**6**	11	8	44

Cum îl implementăm?



Example: Discrete Segment Tree, Yeming Hu

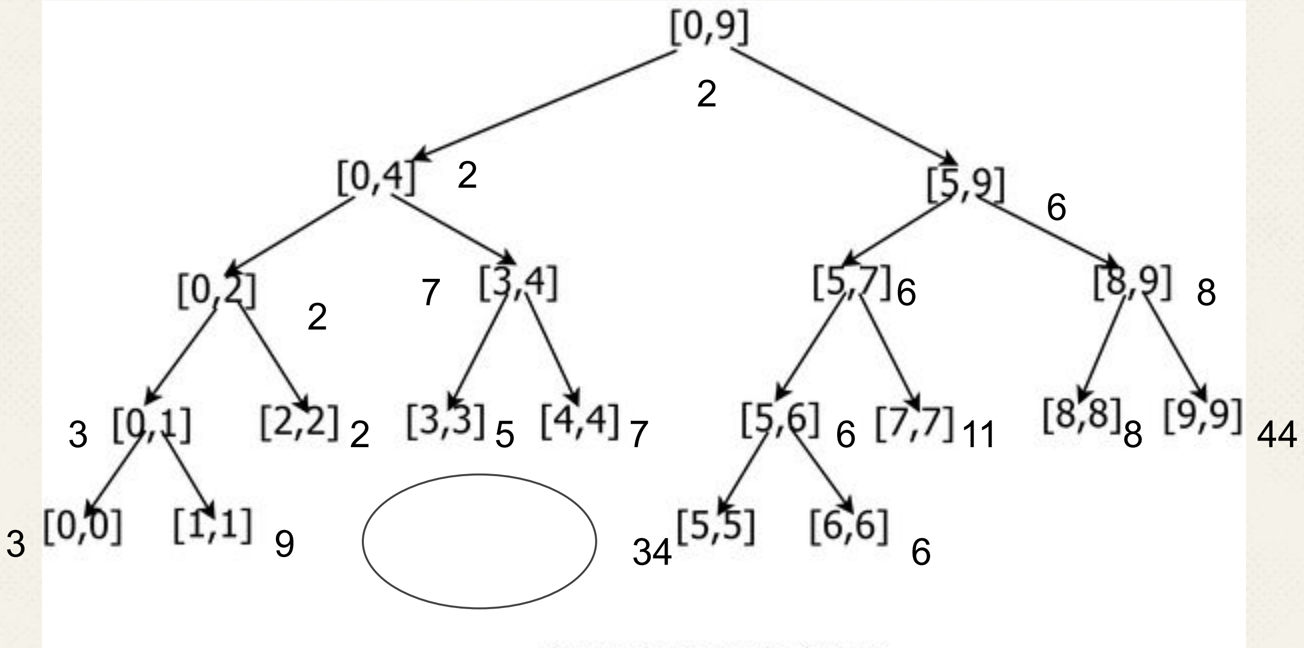
Arbori de Intervale

0	1	2	3	4	5	6	7	8	9
3	9	2	**5**	**7**	**34**	**6**	11	8	44

Cum îl

implementăm?

- Arbore like
- **Vector!**

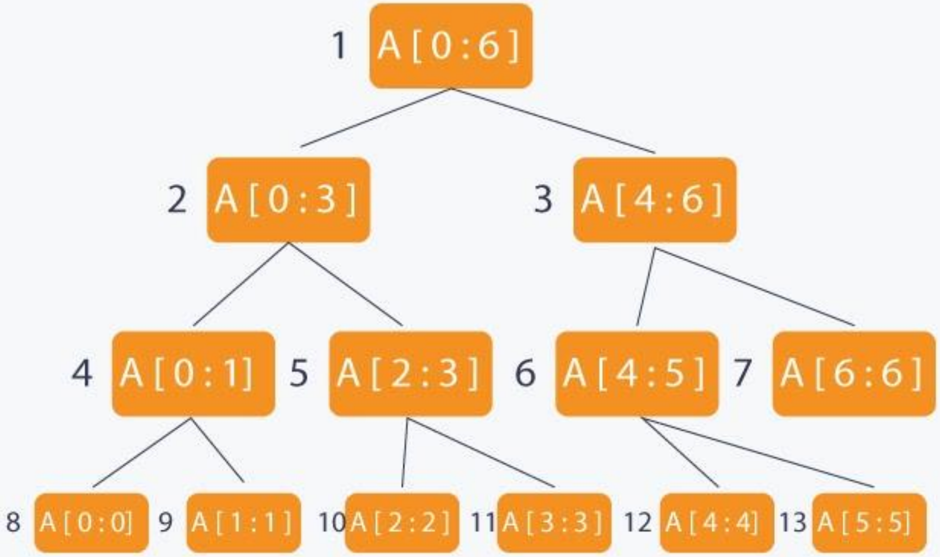


Example: Discrete Segment Tree, Yeming Hu

Arbori de Intervale

tree [1] = A[0:6]
tree [2] = A[0:3]
tree [3] = A[4:6]
tree [4] = A[0:1]
tree [5] = A[2:3]
tree [6] = A[4:5]
tree [7] = A[6:6]
tree [8] = A[0:0]
tree [9] = A[1:1]
tree [10] = A[2:2]
tree [11] = A[3:3]
tree [12] = A[4:4]
tree [13] = A[5:5]

Segment Tree represented as linear array



Segment Tree

Arbori de Intervale

Reprezentare similară cu heapul:

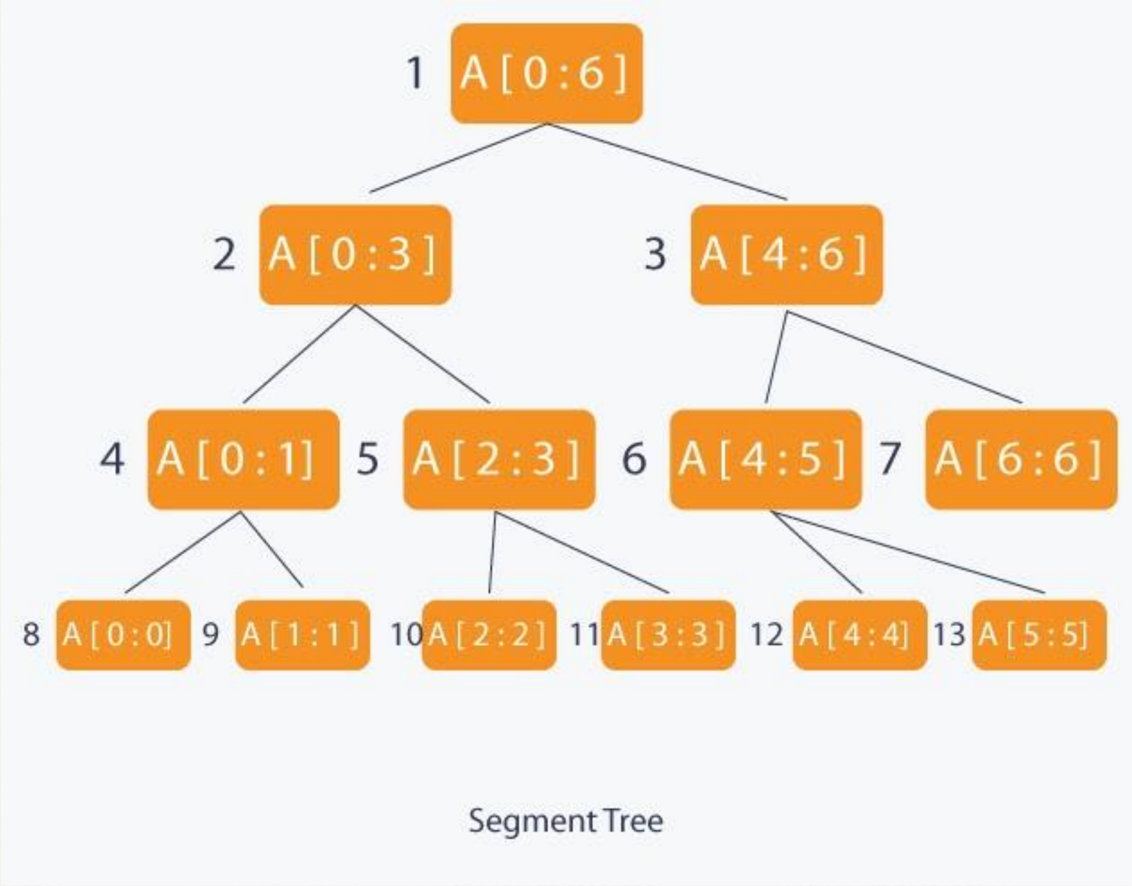
- Rădăcina (1 de multe ori) are intervalul [0,n) [L,R)
  - Fiul stâng are [L, (L+R)/2]; el are poziția în vector i*2
  - Fiul drept are [(L+R)/2 + 1, R]; el are poziția în vector i*2+1
  - Vectorul poate avea niște elemente lipsă pe ultimul rând (vezi 2 slide-uri mai sus).

În total vectorul are 2*n noduri “active”, dar poate avea un pic mai multe.

O(n) memorie.

Operații

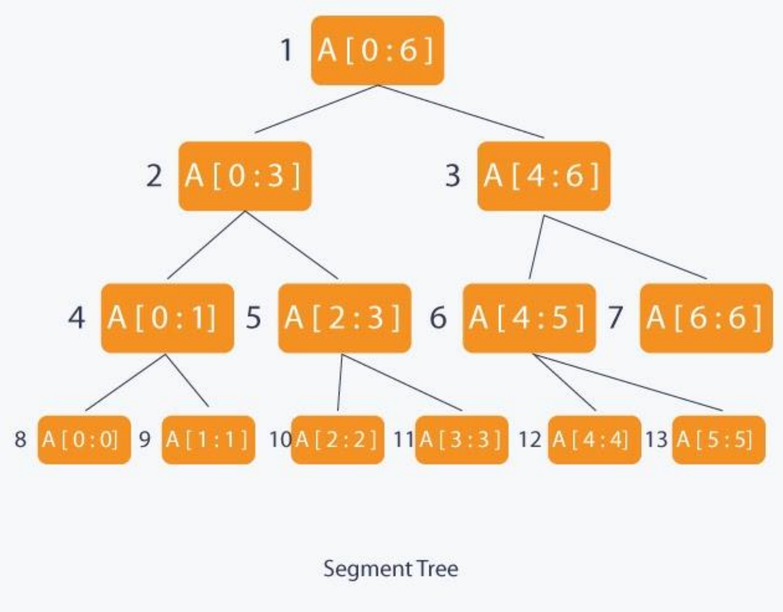
- Query pe index
- Query pe interval
  - Min
  - Sum
- Modificare element
- Modificare interval



Operații

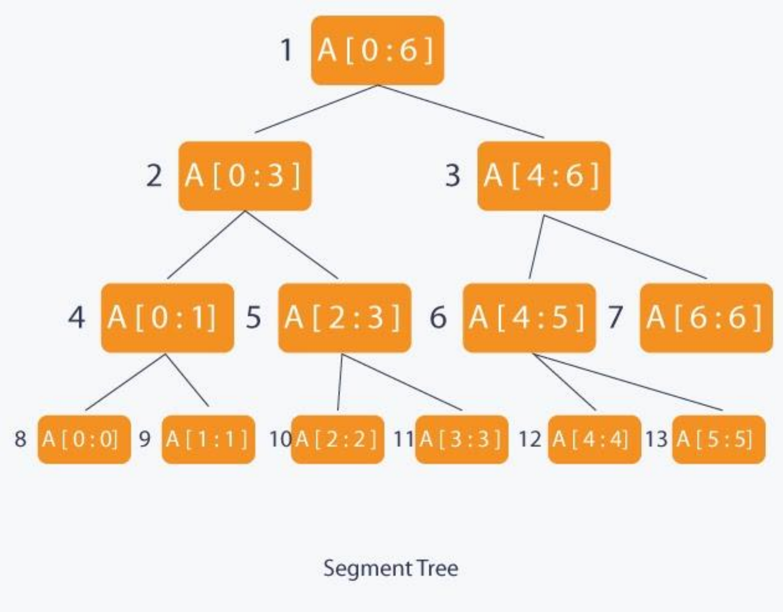
- Query pe index
  - Ori avem “pointeri” spre frunze și răspund direct
  - Ori pornim top down

```
getValue(vector<int> arb_int, int index, int n) {
    int L = 0, R = n, poz = 1;
    while (L != R) {
        if (index > (L + R)/2) {
            L = (L +R)/2, poz = poz*2 + 1;
        }
        else {
            R = (L+R)/2, poz *=2;
        }
    return arb_int[poz]; // L = R;
}
```



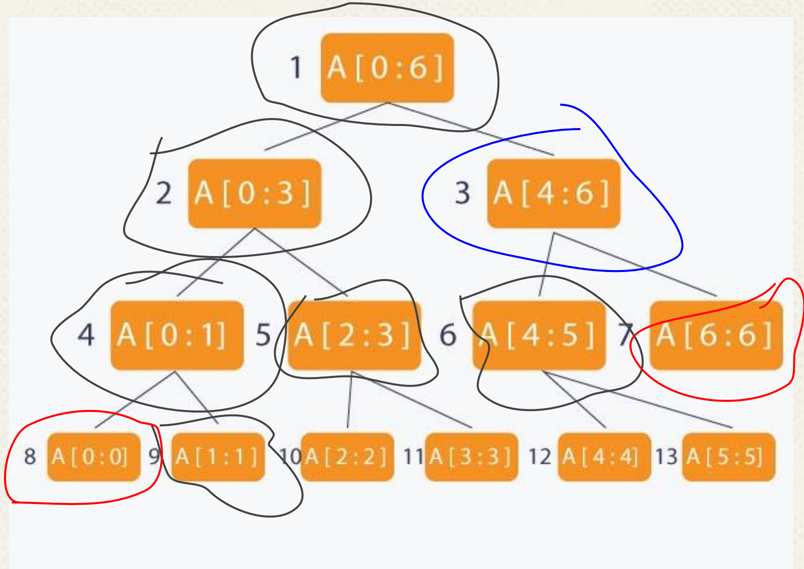
Operații

- Query pe interval
  - Evident, nu luăm toate valorile; ar putea fi liniar
  - Q(1,5) min



Operații

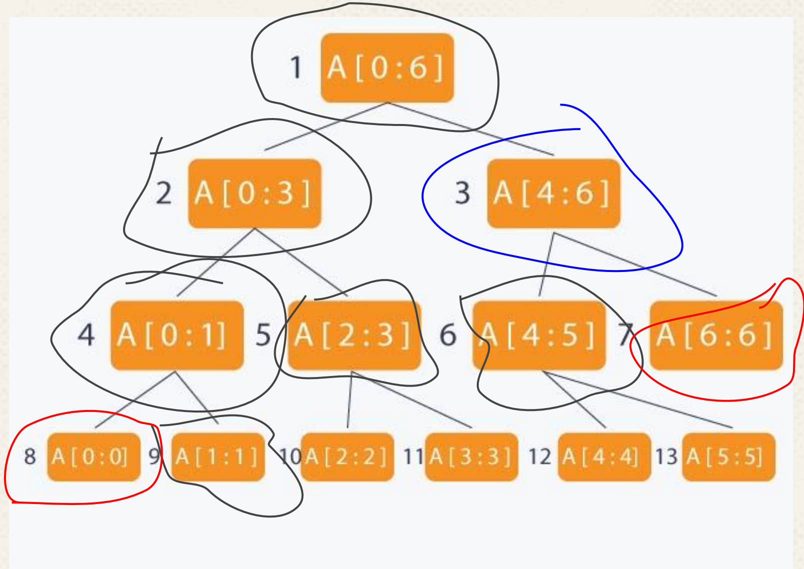
- Query pe interval
  - Evident, nu luăm toate valorile; ar putea fi liniar
  - Q(1,5) min
  - Pornim din rădăcina și mergem recursiv și L și R
  - Dacă intervalul nodului nu se intersectează, oprim
  - **Dacă intervalul e inclus complet, luăm info & ne oprim**
  - Câte noduri putem parcurge?



Segment Tree

Operații

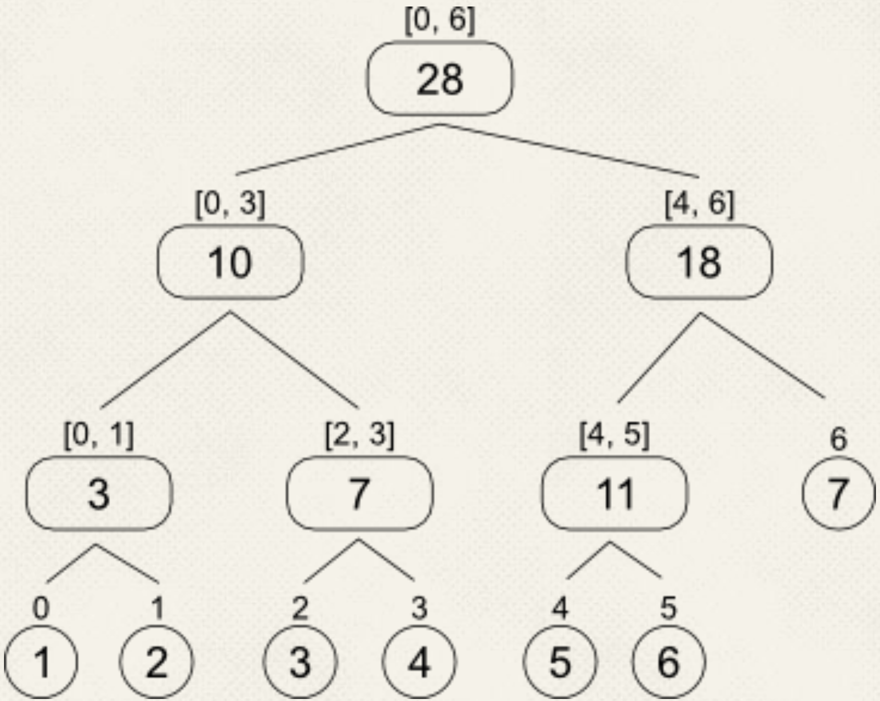
- Query pe interval
  - Evident, nu luăm toate valorile; ar putea fi liniar
  - Q(1,5) min
  - Pornim din rădăcina și mergem recursiv și L și R
  - Caz I — Dacă intervalul nodului nu se intersectează, oprim
  - Caz II — **Dacă intervalul e inclus complet, luăm info & ne oprim**
  - Câte noduri putem parcurge?
    - Doar 4*log n
    - Coborâm pe o ramură până facem un split
      - După split, în fiecare parte, unul dintre fii va fi ori cazul I, ori cazul II, deci se va coborî pe maxim 2 drumuri până jos.



Segment Tree

Operații

- Query pe index
- Query pe interval
  - **Sum (1,5)**
    - **?**

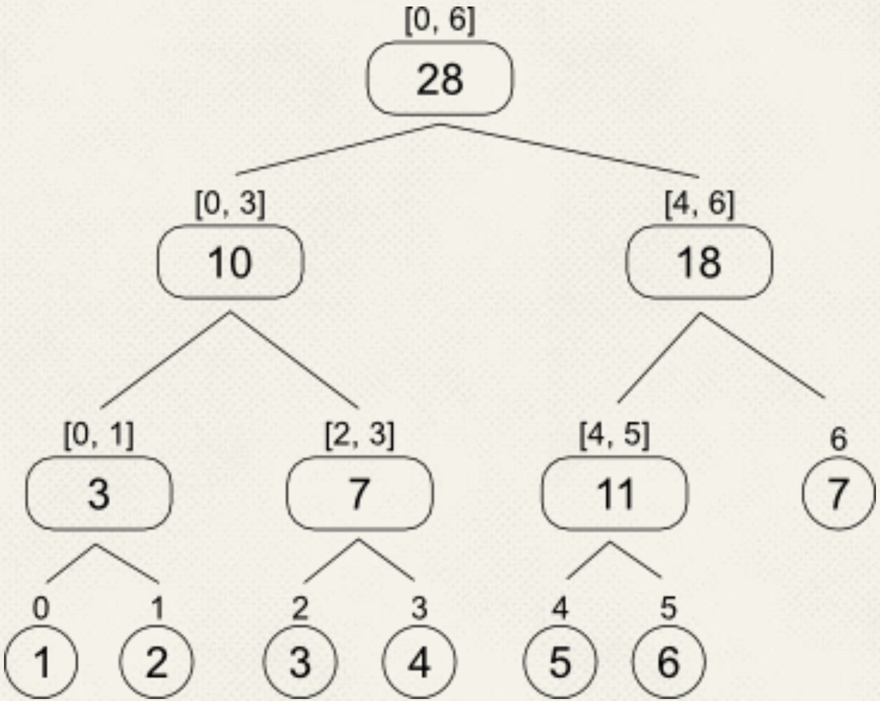


Operații

- Modificare element
  - Dacă țin suma, pot face top-down
  - Dacă țin minim, pot face ori:
    - Top down up
      - coborâm din rădăcină până găsim frunza pe care o modificăm
      - La urcare, facem update tata = min(cei 2 fii)
    - Bottom up
      - Exact ca mai sus, dar avem deja indexul ținut
  - Înapoi la sortare (https://leetcode.com/problems/sort-an-array/submissions/)

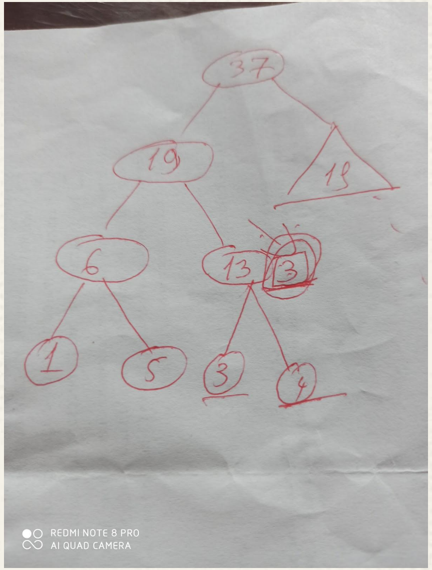
Operații

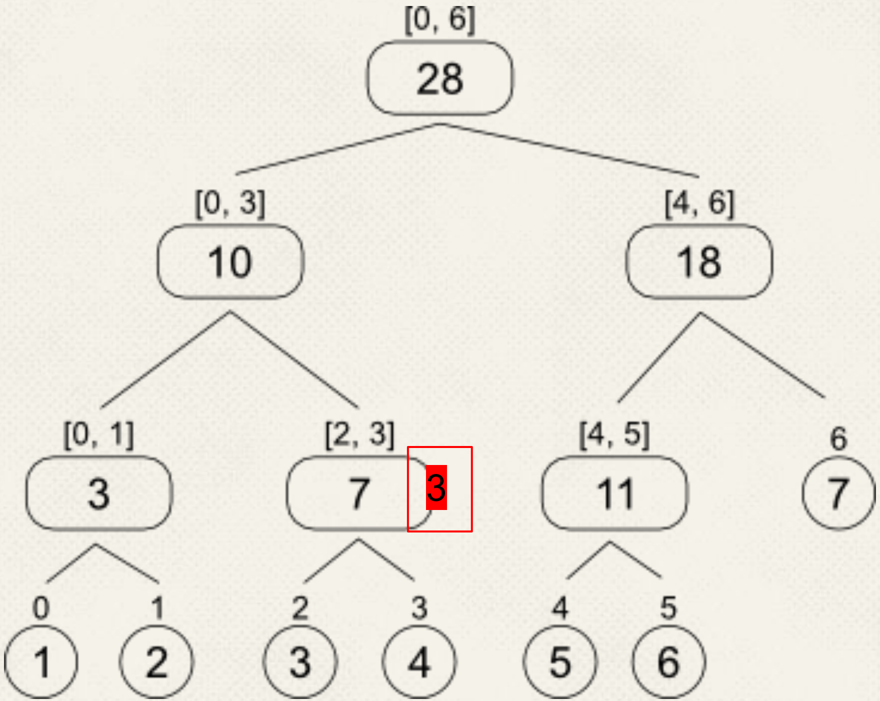
- Modificare pe interval
  - Similar cu query pe interval
  - Merg recursiv în ambii fii
    - Mă opresc dacă nu am intersecție
    - Modific doar nodul actual dacă este inclus de tot în interval
      - Aici trebuie să ținem în nod o informație suplimentară (toate nodurile cresc cu o anumită valoare)
    - Cobor dacă e intersectie parțială



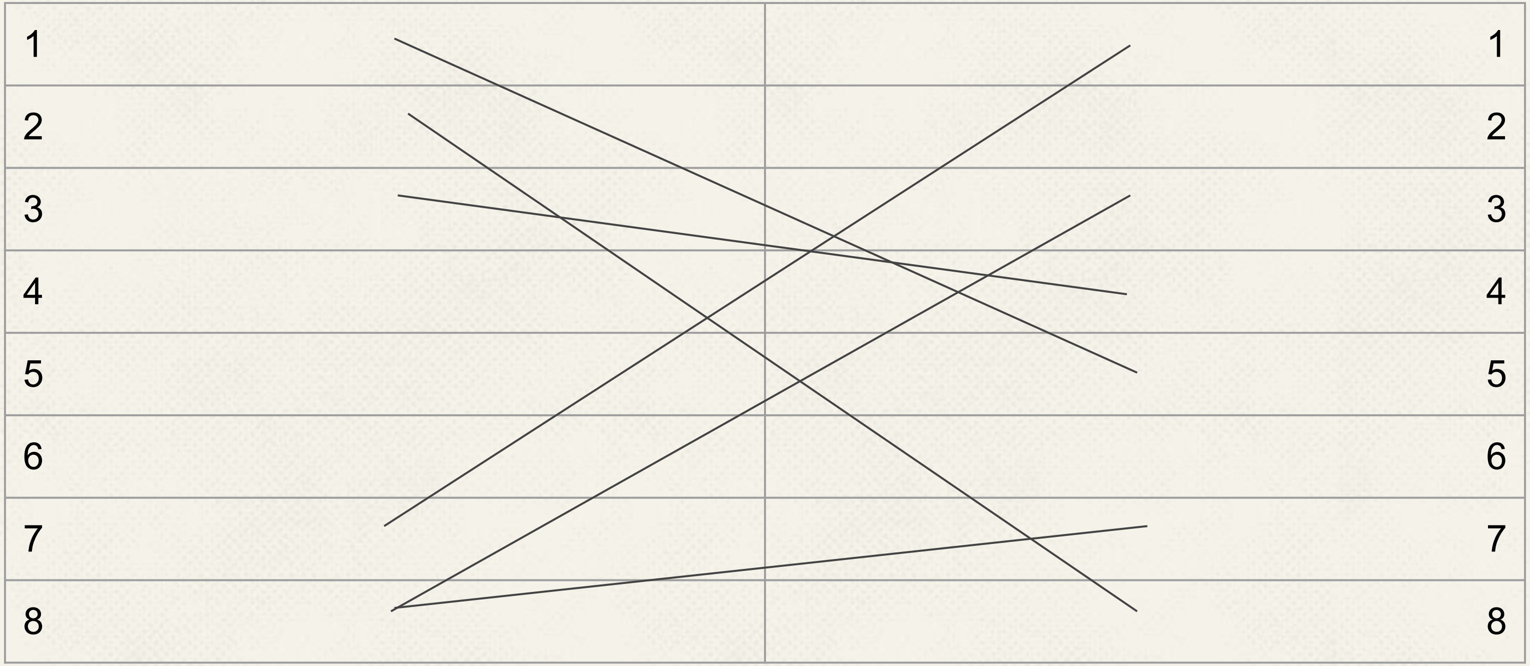
Operații

- Modificare pe interval
  - Add(3, 1, 3) (adaugă 3 la fiecare element din intervalul 1, 3)
  - O mică atenție la query-uri





Câte intersecții am?



Problemă pentru seminar 6!

1 **5**, 2 **8,** 3 **4**, 7 1, 8 3, 8 7

Implementare

- https://www.hackerearth.com/practice/data-structures/advanced-data-structures/segment-trees/tutorial/

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 100100;
int n, q, arb[4 * N];

void update(int nod, int st, int dr, int idx, int val){
    if (st == dr){
        arb[nod] = val;
        return;
    }
    int mid = (st + dr) >> 1;
    if (idx <= mid) update(2 * nod, st, mid, idx, val);
    else update(2 * nod + 1, mid + 1, dr, idx, val);
    arb[nod] = min(arb[2 * nod], arb[2 * nod + 1]);
}

int query(int nod, int st, int dr, int l, int r){
    if (st >= l && dr <= r)
        return arb[nod];
    int mid = (st + dr) >> 1;
    int left_side = 1e9, right_side = 1e9;
    if (l <= mid) left_side = query(2 * nod, st, mid, l, r);
    if (r > mid) right_side = query(2 * nod + 1, mid + 1, dr, l, r);
    return min(left_side, right_side);
}

```

```cpp
int main(){
    cin >> n >> q;
    for (int i=1; i<=n; i++){
        int x; cin >> x;
        update(1, 1, n, i, x);
    }
    while (q--){
        char c; int x, y;
        cin >> c >> x >> y;
        if (c == 'q')
            cout << query(1, 1, n, x, y) << '\n';
        else
            update(1, 1, n, x, y);
    }
    return 0;
}
```