AVL

AVL (veți avea la examen)

- Video (MIT de la minutul 29).
- Lecture Notes

Treapuri

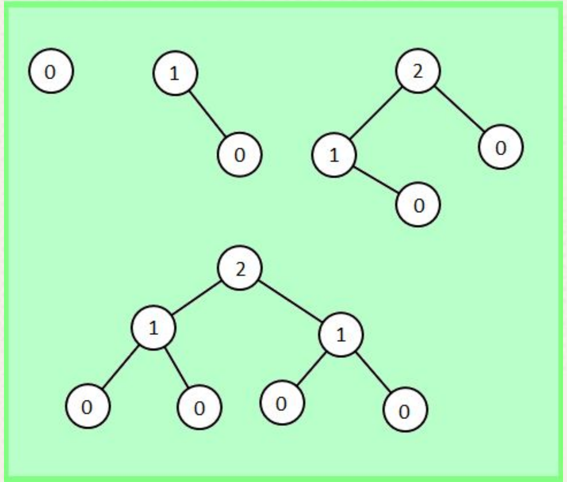
B-Arbori

Arbori echilibrați

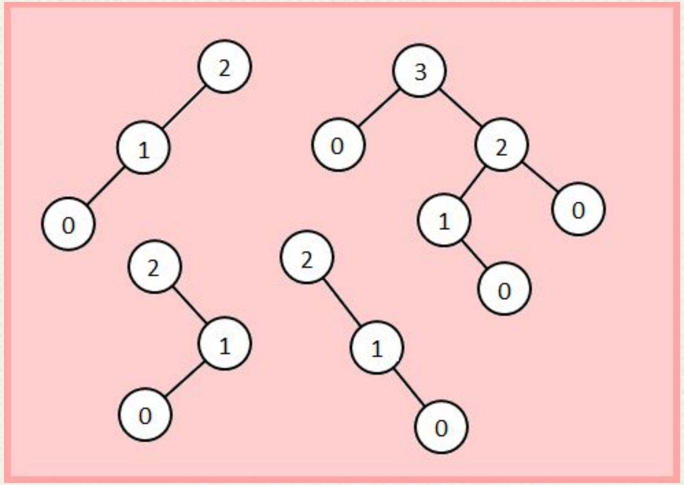
Un **arbore echilibrat** este un arbore în care, **pentru orice nod**, diferența dintre înălțimile subarborilor stâng și drept este de maxim 1.

Exemple:

Echilibrați



Neechilibrați

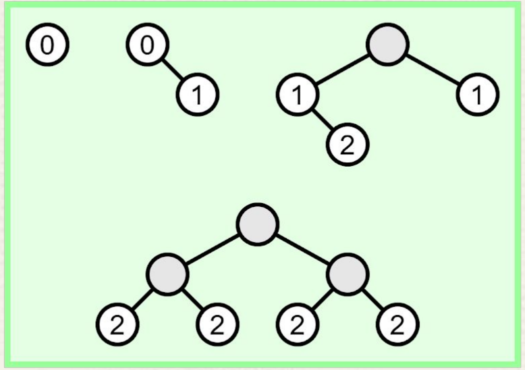


Arbori echilibrați

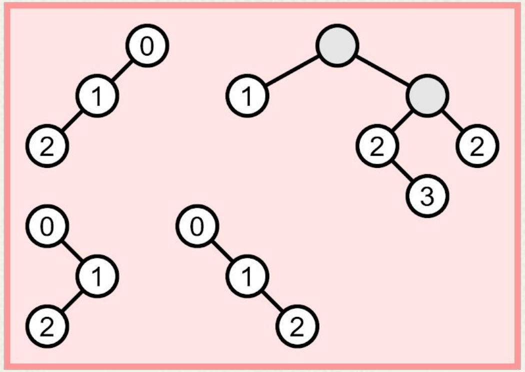
Un **arbore echilibrat** este un arbore în care, **pentru orice nod**, diferența dintre înălțimile subarborilor stâng și drept este de maxim 1.

Exemple:

Echilibrați



Neechilibrați



B-Arbori

Un **B-Arbore** este un arbore echilibrat, destinat căutării eficiente de informație.

Un B-Arbore poate avea mai mult de 2 fii pentru un nod (se poate ajunge și la ordinul sutelor).

Totuși, înălțimea arborelui rămâne O(log n), datorită unei baze a logaritmului convenabilă.

În practică, B-Arborii sunt folosiți pentru baze de date și sisteme de fișiere, pentru citirea și scrierea eficientă pe discul de memorie.
O proprietate importantă a lor este faptul că rețin multă informație. De aceea, B-Arborii reduc numărul de accesări ale discului (accesarea discului este o operație costisitoare).

B-Arbori

Un **B-Arbore** este un arbore echilibrat, destinat căutării eficiente de informație.

Un B-Arbore poate avea mai mult de 2 fii pentru un nod (se poate ajunge și la ordinul sutelor).

Totuși, înălțimea arborelui rămâne O(log n), datorită unei baze a logaritmului convenabilă.

Proprietăți:

1. Un nod poate să conțină mai mult de o cheie
2. Numărul de chei ale unui nod **x** este **n[x]**
3. Un nod **x** are **n[x] + 1** fii
4. Toate frunzele unui B-Arbore se află pe același nivel

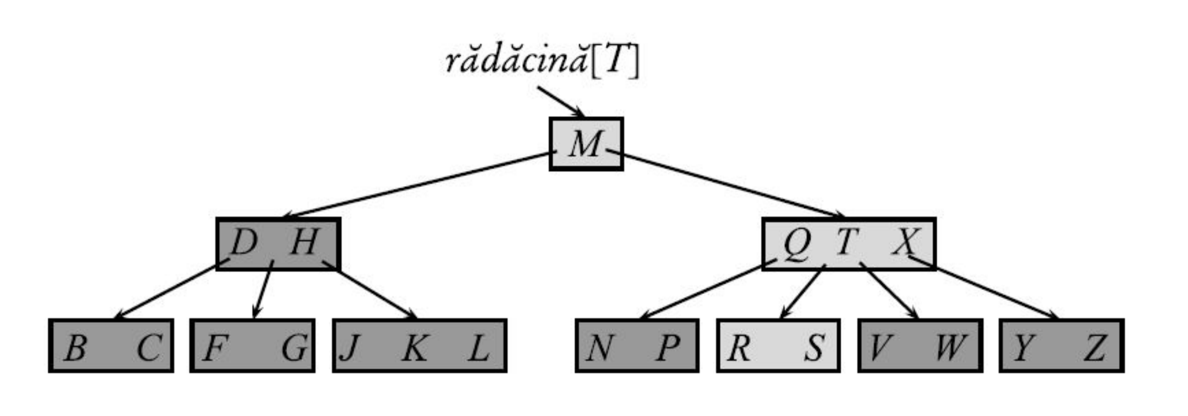
B-Arbori

Proprietăți:

1. Un nod poate să conțină mai mult de o cheie
2. Numărul de chei ale unui nod **x** este **n[x]**
3. Un nod **x** are **n[x] + 1** fii
4. Toate frunzele unui B-Arbore se află pe același nivel

B-Arbori

Exemplu:



Cheile acestui arbore sunt consoanele din alfabetul latin.
Observăm că un nod poate avea mai multe chei, iar fiecare nod **x** care are **n[x]** valori va avea **n[x] + 1** fii.

Căutarea literei ”R” este exemplificată pe traseul hașurat cu o culoare mai deschisă.

B-Arbori

Câmpurile unui nod:

1. **n[x]** – numărul de chei memorate în nodul **x**

2. cele **n[x]** chei, memorate în ordine crescătoare:

   $$cheie_1[x] \le cheie_2[x] \le cheie_3[x] \le \ldots \le cheie_{n[x]}[x]$$

3. o valoare booleană **frunză[x]** – **True**, dacă nodul **x** este frunză, **False**, dacă nodul **x** este nod intern

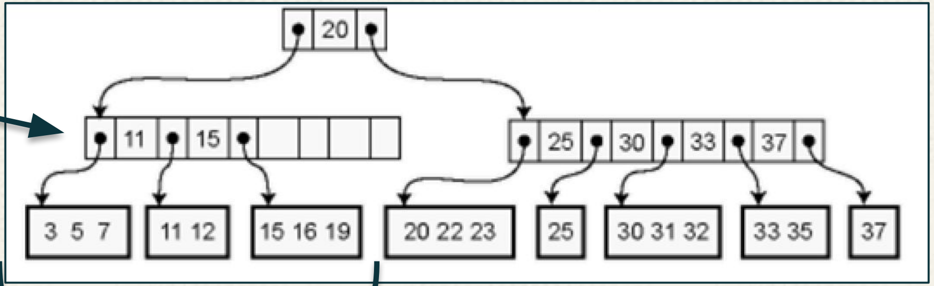
Dacă **x** este un nod intern, atunci el conține **n[x] + 1** pointeri către fiii săi. Nodurile frunză nu au fii, deci nu au aceste câmpuri definite.

B-Arbori

Cheile nodului **x** separă domeniile de chei aflate în fiecare subarbore astfel:

- dacă $\mathbf{k_i}$ este o cheie oarecare memorată într-un subarbore cu rădăcina $\mathbf{fiu_i[x]}$, atunci
  $k_1 \leq cheie_1[x] \leq k_2 \leq cheie_2[x] \leq \ldots \leq cheie_{n[x]}[x] \leq k_{n[x]+1}$

Nodul x



$fiu_1[x], \ldots, fiu_{n[x]+1}[x]$

3, 5, 7 ≤ 11
11 ≤ 11, 12
11, 12 ≤ 15
15 ≤ 15, 16, 19

B-Arbori

Gradul unui B-Arbore

Există o limitare inferioară și una superioară a numărului de chei ce pot fi conținute într-un nod. Exprimăm aceste margini printr-un întreg fixat $t \geq 2$, numit ***grad minim*** al B-Arborelui.

B-Arbori

Restricții:

1. Fiecare nod, cu excepția rădăcinii, trebuie să aibă **cel puțin *t – 1* chei**.
   Consecință: fiecare nod intern trebuie să aibă **cel puțin *t*** fii.

2. Dacă arborele este nevid, atunci rădăcina trebuie să aibă cel puțin o cheie.

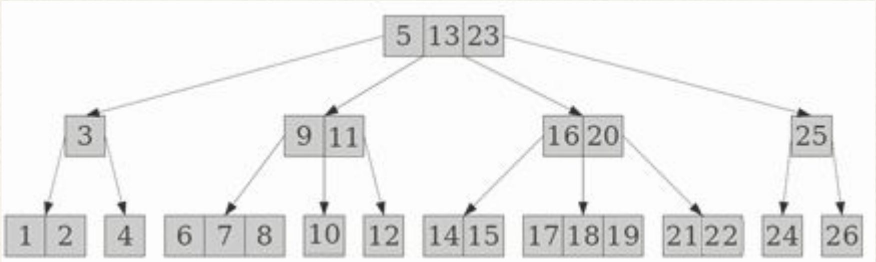
3. Fiecare nod poate să aibă **cel mult *2t – 1*** chei.
   Consecință: orice nod intern poate să aibă **cel mult *2t*** fii.

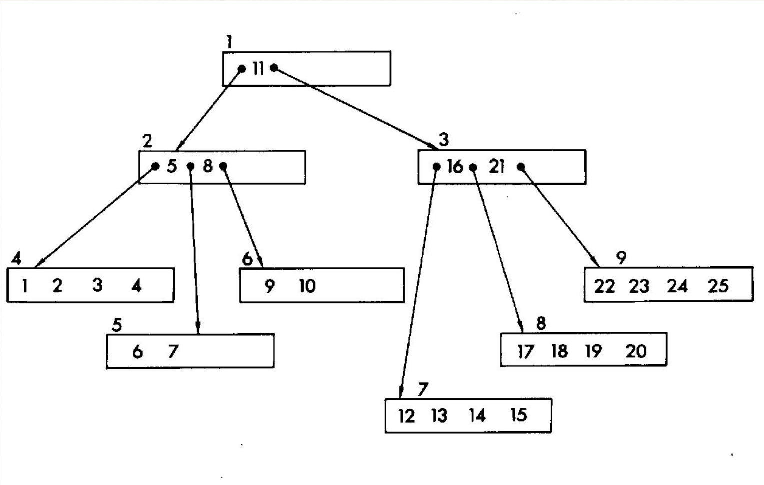
Un nod cu *2t – 1* chei se numește **nod plin**.

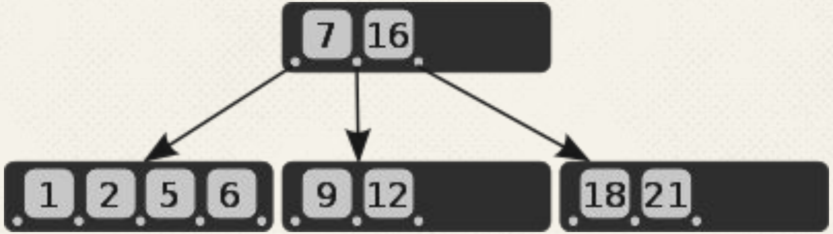
B-Arbori

Exemple de B-Arbori:

B-Arbore de ordin 2 (arbore 2-3-4)







Înălțimea unui B-Arbore

Teoremă: Dacă $n \geq 1$, atunci, pentru orice B-Arbore T cu **n** chei și grad minim *$t \geq 2$:*

$$h \leq \log_t \frac{n+1}{2} \quad - \quad \text{înălțimea arborelui}$$

Demonstrație: Dacă un B-Arbore are înălțimea h, atunci va avea număr minim de chei **dacă** rădăcina conține o singură cheie, iar toate celelalte noduri câte $t - 1$ chei.
În acest caz, există:

- pe nivelul 1: 2 noduri
- pe nivelul 2: 2t noduri
- pe nivelul 3: $2t^2$ noduri
- ...
- pe nivelul h: $2t^{h-1}$

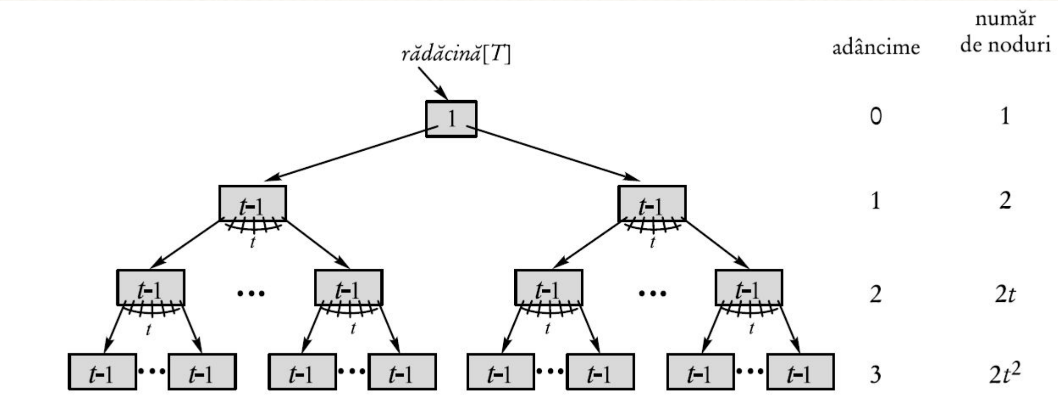
Deci:

$$n \geq 1 + (t-1)\sum_{i=1}^{h} 2t^{i-1} = 1 + 2(t-1)\left(\frac{t^h - 1}{t-1}\right) = 2t^h - 1$$

Înălțimea unui B-Arbore

Exemplu:

Pentru un B-Arbore de înălțime 3, cu număr minim de chei, care are, în fiecare nod, numărul de chei reținute **n[x]**, avem următorul desen:



Înălțimea unui B-Arbore

Concluzie:

Înălțimea unui arbore este

$$h \le \log_t \frac{n+1}{2}$$

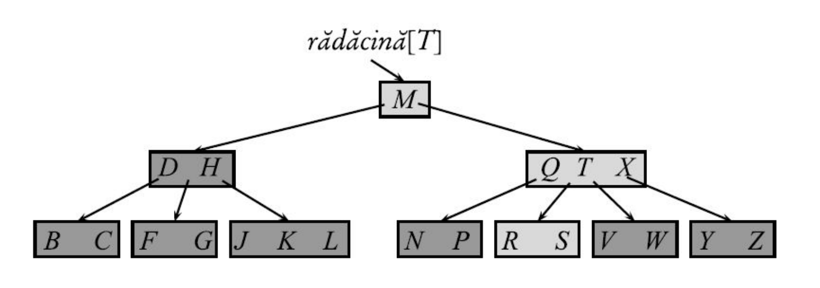
Deci, înălțimea unui B-Arbore crește proporțional cu **O(log n).**

Discuție

Exerciții:

1. De ce nu putem permite gradul minim *t = 1* ?

2. Pentru ce valori ale lui *t*, arborele de mai jos este un B-Arbore, conform definiției?



3. Desenați toti B-Arborii corecți cu grad minim 2 care să reprezinte mulțimea {1, 2, 3, 4, 5}.

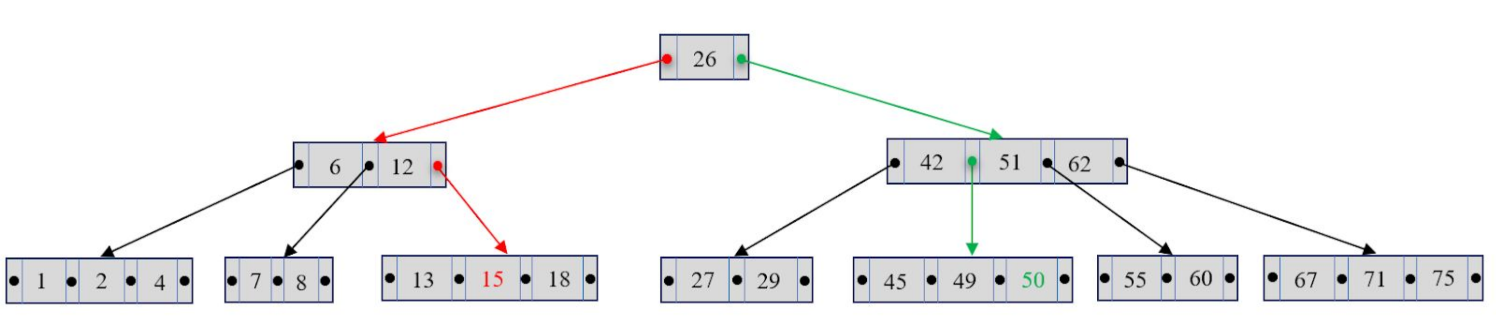
Operații de bază

Căutarea în B-Arbore

Căutarea într-un B-Arbore este asemănătoare cu o căutare într-un arbore binar.

Într-un B-Arbore, căutarea se realizează comparând cheia căutată **x** cu cheile nodului curent, plecând de la nodul rădăcină.

Căutare reușită pentru 50 și nereușită pentru 17.

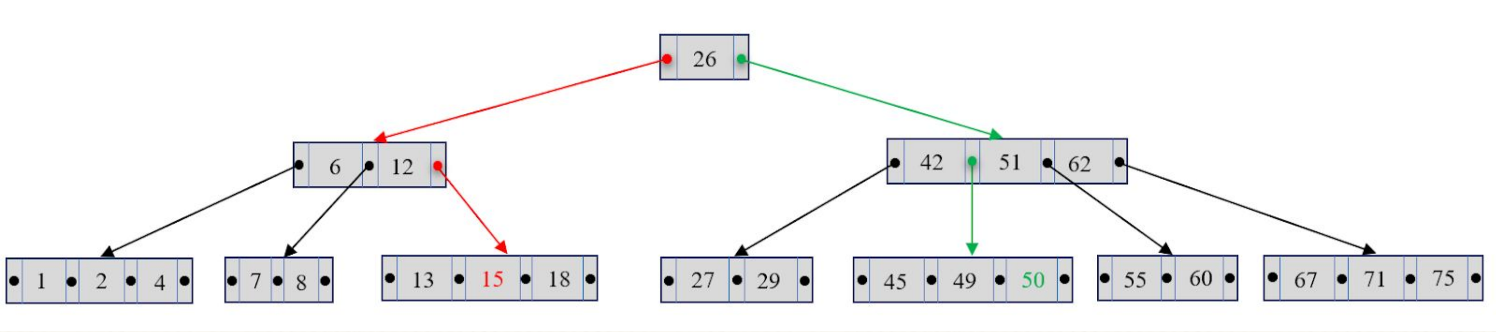


Căutarea în B-Arbore

Algoritm:

1. Căutăm cheia **x** în rădăcină
2. Dacă nu o găsim, atunci continuăm căutarea în fiul corespunzător valorii **x**
3. Dacă găsim cheia, returnăm perechea de valori (y, i), reprezentând nodul, respectiv poziția în nod pe care s-a găsit valoarea **x**.

Putem afla indicele fiului care trebuie explorat în continuare la pasul 2 folosind **căutarea binară dacă numărul de valori din fiecare nod este mare**.



Căutarea în B-Arbore

Căutarea într-un B-Arbore este asemănătoare cu o căutare într-un arbore binar.

Într-un B-Arbore, căutarea se realizează comparând cheia căutată **x** cu cheile nodului curent, plecând de la nodul rădăcină.

Algoritm:

1. Căutăm cheia **x** în rădăcină
2. Dacă nu o găsim, atunci continuăm căutarea în fiul corespunzător valorii **x**
3. Dacă găsim cheia, returnăm perechea de valori (y, i), reprezentând nodul, respectiv poziția în nod pe care s-a găsit valoarea **x**.

Putem afla indicele fiului care trebuie explorat în continuare la pasul 2 folosind căutarea binară.

Căutarea în B-Arbore

Complexitate:

Procesul se repetă de cel mult **O(h)** ori, în cazul în care valoarea căutată se află într-o frunză.

Căutarea valorii într-un nod se realizează (folosind căutarea binară) în **O(log t)**.

Complexitate finală:

$O(h * \log t) = O(\log t * \log_t n)$

$= O(\log t * (\log n / \log t))$

$= \mathbf{O(\log n)}$

Inserarea în B-Arbore

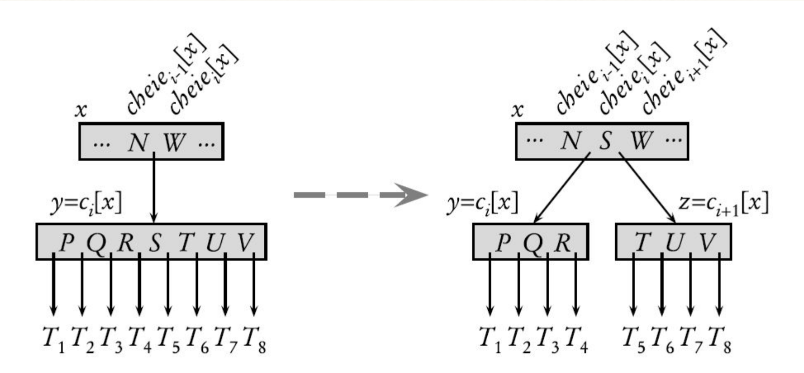
Pentru a insera o cheie **x** într-un B-Arbore, trebuie distinse două cazuri: când nodul unde trebuie introdus are mai puțin de *2t-1* chei, respectiv când are *2t-1* chei.

Algoritm:

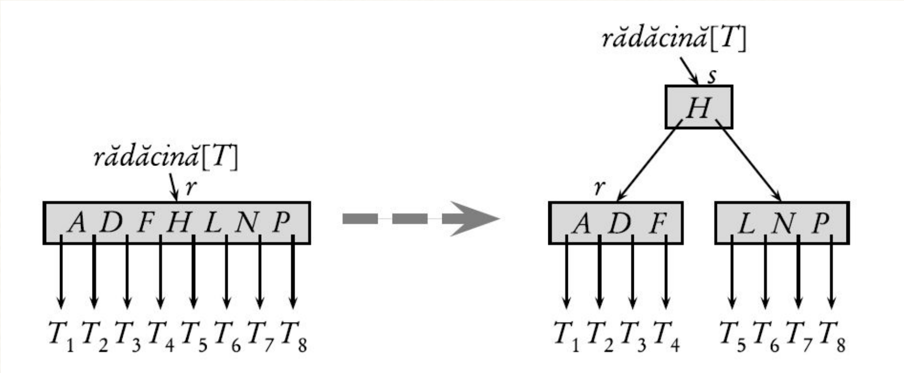
1. Aplicăm operația de căutare pentru a găsi nodul unde trebuie introdusă cheia. Notăm acest nod cu **X** și va fi o **frunză**.
2. Dacă **X** are mai puțin de 2t-1 chei, atunci inserarea se efectuează fără a modifica structura arborelui.
3. Dacă **X** are 2t-1 chei, atunci acesta trebuie divizat. Rezultă, astfel, două noduri noi, F_s (fiul din stânga) și F_d (fiul din dreapta).
4. Eliminăm cea mai mare cheie din F_s (cheia mediană). O notăm cu **M**.
5. **M** devine părintele celor două noduri F_s și F_d.
6. Se încearcă (recursiv) adăugarea lui **M** în părintele lui **X**.

Inserarea în B-Arbore

Divizarea unui nod (t = 4):



Nod intermediar



Rădăcină

Inserarea în B-Arbore

Complexitate:

Căutarea nodului în care trebuie introdusă cheia: **$O(\log_t n)$**
Pentru un nivel: **$O(t)$**
Recursivitatea: $O(h)$ = **$O(\log_t n)$**

Complexitatea finală: **$O(t \log_t n)$**

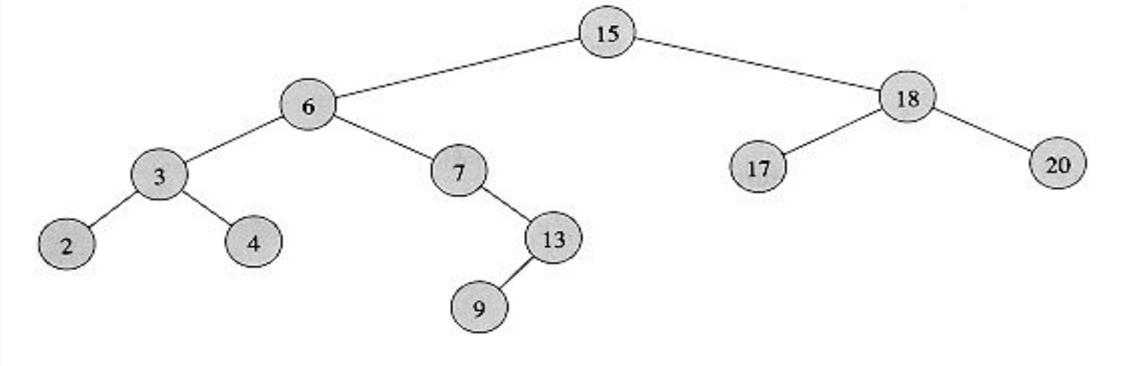
Arbori Binari de Căutare Construiți Aleator

- Amestecăm bine de tot și inserăm elementele în arborele binar de căutare. Ce înălțime va avea?

Arbori Binari de Căutare Construiți Aleator

Lema 13.3. Notăm cu T arborele care rezultă prin inserarea a n chei distincte $k_1, k_2, \ldots, k_n$ (în ordine) într-un arbore binar de căutare inițial vid. Cheia k_i este un strămoş al cheii k_j în T, pentru $1 \le i < j \le n$, dacă şi numai dacă:

- **$k_i = \min\{ k_l : 1 \le l \le i$ şi $k_l > k_j \}$** // k_i este cel mai mic număr mai mare decât k_j din primele i, practic în procesul de inserare a lui j vom ajunge în k_i și vom merge în stânga
- SAU **$k_i = \max\{ k_l : 1 \le l \le i$ şi $k_l < k_j \}$**
  - 13 e fiu al lui 7 (până la momentul inserării lui 13, 7 era cel mai mare număr mai mic)
  - Ulterior, proprietatea e valabilă și pentru 9 și 14.
  - Ce trebuia să se întâmple ca 18 să fie fiu al lui 7?



Arbori Binari de Căutare Construiți Aleator

Demonstrație:

'⇒': Presupunem că k_i este un strămoş al lui $k_\square$. Notăm cu T_i arborele care rezultă după ce au fost inserate în ordine cheile $k_1, k_2, \ldots, k_i$. Drumul de la rădăcină la nodul k_i în T_i este acelaşi cu drumul de la rădăcină la nodul k_i în T. De aici, rezultă că, dacă s-ar insera în arborele T_i nodul $k_\square$, acesta ($k_\square$) ar deveni fie fiu stâng, fie fiu drept al nodului k_i. Prin urmare (vezi exerciţiul 13.2-6), k_i este fie cea mai mică valoare dintre $k_1, k_2, \ldots, k_i$ care este mai mare decât $k_\square$, fie cea mai mare valoare dintre cheile $k_1, k_2, \ldots, k_i$ care este mai mică decât $k_\square$.

'⇐': Presupunem că k_i este cea mai mică valoare dintre $k_1, k_2, \ldots, k_i$ care este mai mare decât $k_\square$. (Cazul când k_i este cea mai mare cheie dintre $k_1, k_2, \ldots, k_i$ care este mai mică decât $k_\square$ se tratează simetric). Compararea cheii $k_\square$ cu oricare dintre cheile de pe drumul de la rădăcină la k_i în arborele T produce aceleaşi rezultate ca şi compararea cheii k_i cu cheile respective. Prin urmare, pentru inserarea lui $k_\square$, se va parcurge drumul de la rădăcină la k_i, apoi $k_\square$ se va insera ca descendent al lui k_i.

Arbori Binari de Căutare Construiți Aleator

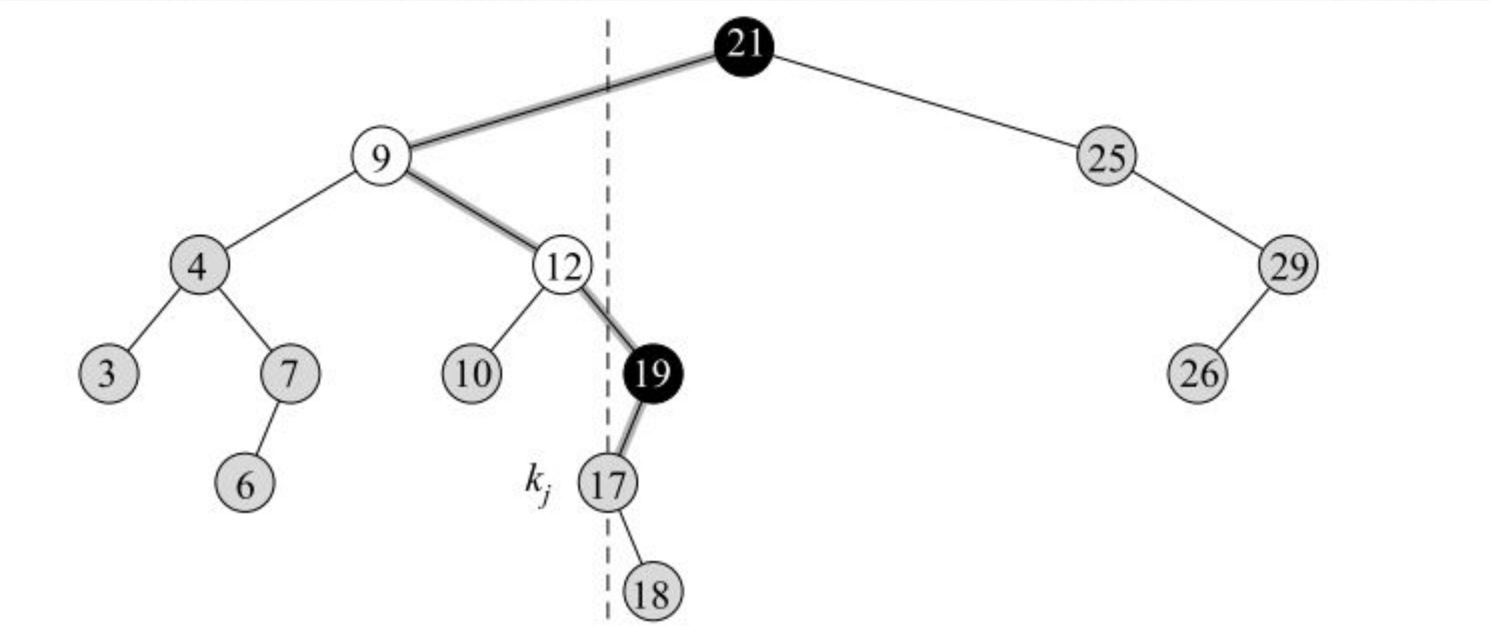
Corolarul 13.4. Fie T arborele care rezultă prin inserarea a n chei distincte $k_1, k_2, \ldots, k_n$ (în ordine) într-un arbore binar de căutare iniţial vid. Pentru o cheie k_j dată, cu $1 \leqslant j \leqslant n$, definim mulţimile:

- $G_j = \{ k_i : 1 \leqslant i < j$ şi $k_l > k_i > k_j$ pentru toţi indicii $l < i$ cu $k_l > k_j \}$
- $L_j = \{ k_i : 1 \leqslant i < j$ şi $k_l < k_i < k_j$ pentru toţi indicii $l < i$ cu $k_l < k_j \}$

Atunci cheile de pe drumul de la rădăcină la k_j sunt chiar cheile din $G_j \cup L_j$, iar adâncimea oricărei chei k_j din T este $d(k_j, T) = |G_j| + |L_j|$.

Arbori Binari de Căutare Construiți Aleator

Cu negru sunt nodurile care sunt, la inserarea lor, cel mai mic element mai mare decât 19 (G → greater). Similar, cele cu alb sunt elemente care, la inserarea lor, erau cele mai mari elemente mai mici decât 19 (L → lower).



Arbori Binari de Căutare Construiți Aleator

Practic, pentru a calcula înălțimea unui arbore, trebuie să calculăm $\mathbf{max}_{1 \le j \le n}$**(|G□| + |L□|)**.

Simplificăm și discutăm cum calculăm de câte ori se modifică, în medie, minimul, dacă inserăm **n** elemente pe rând.

Exercițiu: Care este probabilitatea ca k_i să fie minimul primelor i numere?

Arbori Binari de Căutare Construiți Aleator

Practic, pentru a calcula înălțimea unui arbore, trebuie să calculăm $\mathbf{\max_{1 \le j \le n}(|G_j| + |L_j|)}$.

Simplificăm și discutăm cum calculăm de câte ori se modifică, în medie, minimul, dacă inserăm **n** elemente pe rând.

Răspuns: Probabilitatea ca k_i să fie minimul primelor i numere este **1/i**.

Prin urmare, numărul mediu de modificări este $\sum_{i=1}^{n} \frac{1}{i} = H_n$,

unde $\mathbf{H_n = \ln(n) + O(1)}$ este al n-lea număr armonic.

→ Avem **log(n)** modificări.

Arbori Binari de Căutare Construiți Aleator

Lema 13.5. Fie $k_1, k_2, \ldots, k_n$ o permutare oarecare a unei mulţimi de n numere distincte şi fie |S| variabilă aleatoare reprezentând cardinalul mulţimii.

$$S = \{ k_i : 1 \leqslant i \leqslant n \text{ şi } k_l > k_i \text{ pentru orice } l < i \} \quad \textbf{(13.1)}$$

Atunci $\Pr\{ |S| \geqslant (\beta + 1)H_n \} \leqslant 1/(n^2)$, unde H_n este al n-lea număr armonic, iar $\beta \approx 4{,}32$ verifică ecuaţia $(\ln \beta - 1)\beta = 2$.

Prin urmare, e foarte probabil să avem maxim O(log(n)) modificări ale minimului.

Arbori Binari de Căutare Construiți Aleator

Teorema 13.6. Înălţimea medie a unui arbore binar de căutare construit aleator cu n chei distincte este O(lg n).

Arbori Binari de Căutare Construiți Aleator

Teorema 13.6. Înălţimea medie a unui arbore binar de căutare construit aleator cu n chei distincte este O(lg n).

<u>Demonstrație:</u> Fie $k_1, k_2, \ldots, k_{\square}$ o permutare oarecare a celor n chei şi fie T arborele binar de căutare care rezultă prin inserarea cheilor în ordinea specificată, pornind de la un arbore iniţial vid. Vom discuta prima dată probabilitatea ca adâncimea $d(k_{\square}, T)$ a unei chei date $k_{\square}$ să fie cel puţin t, pentru o valoare t arbitrară. Conform caracterizării adâncimii $d(k_{\square}, T)$ din ***corolarul 13.4***, dacă adâncimea lui $k_{\square}$ este cel puţin t, atunci cardinalul uneia dintre cele două mulţimi $G_{\square}$ şi $L_{\square}$ trebuie să fie cel puţin t/2.

Prin urmare, $\Pr\{ d(k_{\square}, T) \geqslant t \} \leqslant \Pr\{ |G_{\square}| \geqslant t/2 \} + \Pr\{ |L_{\square}| \geqslant t/2 \}$.

Arbori Binari de Căutare Construiți Aleator

Să examinăm la început $\Pr\{\ |G\square| \geqslant t/2\ \}$. Avem

$$\Pr\{\ |G\square| \geqslant t/2\ \} = \Pr\{\ |\{k_i : 1 \leqslant i < j \text{ şi } k\square > k_i > k\square,\ \forall l < i\}| \geqslant t/2\ \}$$

$$\leqslant \Pr\{\ |\{k_i : i \leqslant n \text{ şi } k\square > k_i,\ \forall l < i\}| \geqslant t/2\ \}$$

$$= \Pr\{\ |S| \geqslant t/2\ \},$$

unde S este definit în relaţia **(13.1.)** $S = \{\ k_i : 1 \leqslant i \leqslant n \text{ şi } k\square > k_i,\ \forall l < i\ \}$.

În sprijinul acestei afirmaţii, să observăm că probabilitatea nu va descreşte dacă vom extinde intervalul de variaţie al lui i de la $i < j$ la $i \leqslant n$, deoarece, prin extindere, se vor adăuga elemente noi la mulţime. Analog, probabilitatea nu va descreşte dacă se renunţă la condiţia $k_i > k\square$, deoarece, prin aceasta, se înlocuieşte o permutare a (de regulă) mai puţin de **n** elemente (şi anume acele chei k_i care sunt mai mari decât $k\square$) cu o altă permutare oarecare de n elemente. Folosind o argumentare similară, putem demonstra că

$$\Pr\{\ |L\square|\ | \geqslant t/2\ \} \leqslant \Pr\{\ |S| \geqslant t/2\ \}.$$

Arbori Binari de Căutare Construiți Aleator

Folosind o argumentare similară, putem demonstra că

$$\Pr\{\, |L_{\square}| \geqslant t/2 \,\} \leqslant \Pr\{\, |S| \geqslant t/2 \,\}$$

şi apoi, folosind inegalitatea **(13.2)**, obţinem:

$$\Pr\{\, d(k_{\square}, T) \geqslant t \,\} \leqslant 2 * \Pr\{\, |S| \geqslant t/2 \,\} .$$

Dacă alegem $t = 2(\beta + 1)H_{\square}$, unde $H_{\square}$ este al n-lea număr armonic, iar $\beta \approx 4.32$ verifică ecuaţia $(\ln \beta - 1)\beta = 2$, putem aplica **lema 13.5** pentru a concluziona că

$$\Pr\{\, d(k_{\square}, T) \geqslant 2(\beta + 1)H_{\square} \,\} \leqslant 2 * \Pr\{\, |S| \geqslant (\beta + 1)H_{\square} \,\} \leqslant 2/n^2 .$$

Arbori Binari de Căutare Construiți Aleator

Deoarece discutăm despre un arbore binar de căutare construit aleator şi cu cel mult n noduri, probabilitatea ca adâncimea oricăruia dintre noduri să fie cel puţin 2(β + 1)H□ este, folosind *inegalitatea lui Boole**, de cel mult $n*(2/n^2) = 2/n$. Prin urmare, în cel puţin 1 − 2/n din cazuri, înălţimea arborelui binar de căutare construit aleator este mai mică decât 2(β + 1)H□ şi în cel mult 2/n din cazuri înălţimea este cel mult n. În concluzie, înălţimea medie este cel mult

(2(β + 1)H□)(1 − 2/n) + n(2/n) = O(lg n).

**Inegalitatea lui Boole*:

Fie $A_1, A_2, \ldots, A_{\square}$ în K cu $\bigcap_{i=1}^{n-1} A_i \neq 0$. Atunci: $Pr(\bigcap_{i=1}^{n-1} A_i) \geqslant (\sum_{i=1}^{n} Pr(A_i)) - n - 1$