Facultatea de Matematica si Informatica
Universitatea din Bucuresti

Structuri de Date si Algoritmi

- suport de curs -

Dobrovat Anca - Madalina

An universitar 2019 – 2020
Semestrul I
Seriile 21 + 25

Curs 7 & 8

12/11/2019

Facultatea de Matematica si Informatica
Universitatea din Bucuresti

Curs 7&8 - Cuprins

3. Structuri arborescente

Arbori binari echilibrati AVL.

Performanta cautarii in arbori binari de cautare echilibrati AVL.

Arbori binari stricti. Proprietati matematice. Aplicatii

Teorema AVL. Margine superioara si margine inferioara pentru inaltimea unui arbore binar echilibrat AVL

Sursa: – R. Ceterchi: "Structuri de date si Algoritmi. Aspecte matematice si aplicatii", Editura Univ. din Bucuresti, 2001

Credit: – material de laborator organizat de Ana – Maria Ciucu

Arbori binari de cautare echilibrati AVL

Procedeul clasic de construire a arborelui binar de căutare ne dă un arbore a cărui formă depinde foarte mult de ordinea în care sunt furnizate valorile nodurilor. În cazul cel mai general nu obţinem un arbore de înălţime minimă.

Cazul cel mai favorabil, în care obţinem înălţime minimă, este cel în care ni se furnizează pe rând mijloacele intervalelor (subintervalelor) vectorului sortat.

Cazul cel mai nefavorabil este cel în care valorile vin în ordine crescătoare (sau descrescătoare), caz în care arborele binar de căutare obţinut este degenerat.

Problemă: **cum modificăm algoritmul** de construcţie astfel încât să **obţinem înălţime minimă pentru arbore**, pentru a îmbunătăţi timpul de căutare?

Arbori binari de cautare echilibrati AVL

Arborii binari de cautare sunt eficienti (optimi) doar atunci când sunt *echilibrati (balanced).*

***Ideal* este ca înaltimea arborelui sa fie *O(logn)*, unde n este numarul de noduri din arbore.**

Solutie - reechilibrarea (rebalancing) arborelui în timpul operatiei de inserare astfel încât sa se pastreze si proprietatea arborelui binar de cautare.

Arbori binari de cautare echilibrati AVL

Algoritmi pentru mentinerea arborilor binari de cautare echilibrati:

- AVL (- introdusi în 1962; - G.M. Adelson-Velskii si E.M. Landis),

- arbori rosu-negru (red/black),

- arbori splay,

- arbori binari de cautare construiti aleator (randomized).

Ei difera prin invariantii pe care îi asigura si prin momentul si modul în care se face reechilibrarea.

Arbori binari de cautare echilibrati AVL

Definiţie Se numeşte arbore binar de căutare echilibrat AVL (Adelson-Velskii-Landis) un arbore care în fiecare nod are proprietatea că înălţimile subarborilor stâng şi drept diferă cu cel mult 1.

Pentru un nod dat, fie hl şi hr înălţimile subarborelui stâng, respectiv drept.
Avem trei situaţii posibile în acest nod, codificate cu valorile variabilei

$$bal = hr - hl,$$

pe care o numim factor de echilibru, în felul următor:

$$bal = \begin{cases} 1, & hl = hr - 1 \\ 0, & hl = hr \\ -1, & hl = hr + 1 \end{cases}$$

Informaţia despre valoarea factorului de echilibru în fiecare nod p al unui arbore o vom scrie într-un nou câmp al lui p, câmpul bal: -1..1.
Algoritm de inserare - cu operatii suplimentare, re-echilibrari

Arbori binari de cautare echilibrati AVL

Arborii AVL asigura invariantul de înaltime:

Fie un nod x din arbore. Înaltimea subarborelui sau stâng si cea a subarborelui sau drept difera prin cel mult 1,

$$|h(x \to left) - h(x \to right)| \le 1$$

factor de echilibrare (balance factor).

Arbore echilibrat AVL
- factorul de echilibru: -1, 0, 1

```
struct nod
{
    int info;
    int bal;
    nod *left, *right;
};
```

Arbori binari de cautare echilibrati AVL

Cautarea într-un arbore AVL este aceeasi ca la arbori binari de cautare, deoarece invariantul de înaltime intervine doar la operatia de inserare.

Inserarea

! Cu operaţii suplimentare, re-echilibrari

- dezechilibrare → nerespectarea formulei | hs – hd | ≤ 1

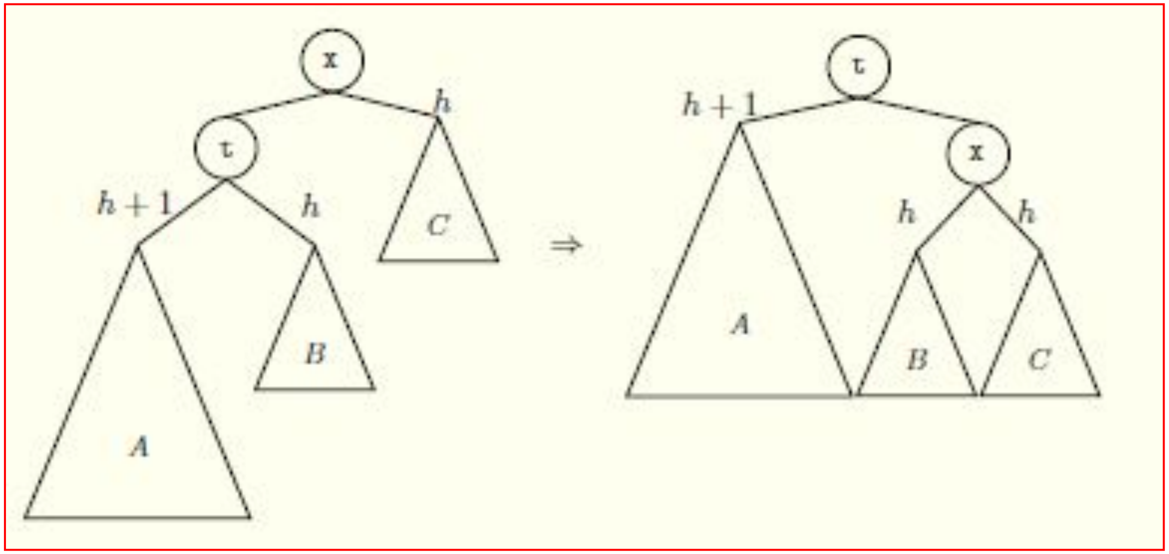
Etape:

1. o cheie se insereaza intr-o prima faza, ca intr-un a.b.c.

2. se parcurge drumul invers (care este unic) si se cauta pe acest drum primul nod care nu este echilibrat; Acest nod trebuie echilibrat si el se va afla intotdeauna intr-unul din cele 4 cazuri:

Arbori binari de cautare echilibrati AVL

1. Rotatie simpla la dreapta




```
void rotatie_dreapta(nod *&p)
{
    nod *t = p->left;
    p->left = t->right;
    t->right = p;
    p -> bal = p->bal + (1 - min(t->bal,0));
    t->bal = t->bal + (1 + max(p->bal,0));
    p = t;
}
```

Arbori binari de cautare echilibrati AVL

1. Rotatie simpla la dreapta

```
void rotatie_dreapta(nod *&p)
{
    nod *t = p->left;
    p->left = t->right;
    t->right = p;
    p -> bal = p->bal + (1 - min(t->bal,0));
    t->bal = t->bal + (1 + max(p->bal,0));
    p = t;
}
```

Etape:

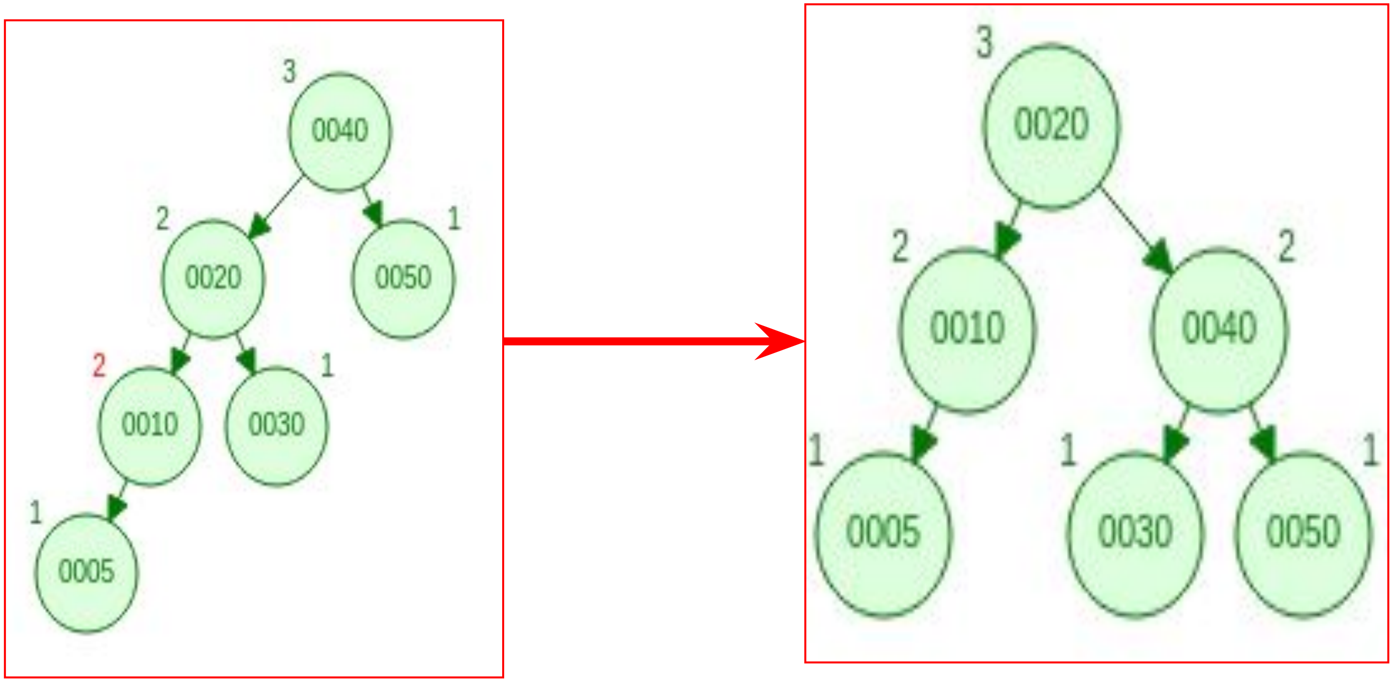
1. **Se pastreaza intr-un pointer adresa subarborelui stang al nodului dezechilibrat;**
2. **Fiul drept al fiului stang al nodului initial, devine fiu stang pentru acesta, dupa re-echilibrare**
3. **Noul fiu drept din subarborele stang va contine adresa nodului dezechilibrat initial.**
4. **Se recalculeaza factorii de echilibru (exista si alte metode…)**

Arbori binari de cautare echilibrati AVL

Rotatie simpla la dreapta Exemplu

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

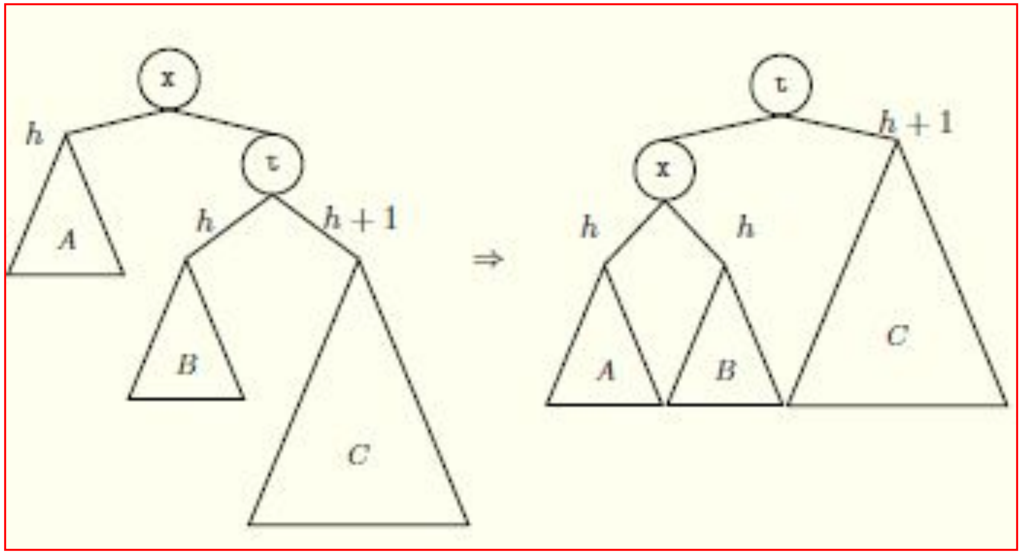
Se insereaza, pe rand: 40, 20, 50, 10, 30, 5



Facultatea de Matematica si Informatica
Universitatea din Bucuresti

Arbori binari de cautare echilibrati AVL

2. Rotatie simpla la stanga




```
void rotatie_stanga(nod *&p)
{
    nod *t = p->right;
    p->right = t->left;
    t->left = p;
    p -> bal = p->bal - (1 + max(t->bal,0));
    t->bal = t->bal - (1 - min(p->bal,0));
    p = t;
}
```

Arbori binari de cautare echilibrati AVL

2. Rotatie simpla la stanga

```
void rotatie_stanga(nod *&p)
{
    nod *t = p->right;
    p->right = t->left;
    t->left = p;
    p -> bal = p->bal - (1 + max(t->bal,0));
    t->bal = t->bal - (1 - min(p->bal,0));
    p = t;
}
```

Etape:

1. **Se pastreaza intr-un pointer adresa subarborelui drept al nodului dezechilibrat;**

2. **Fiul stang al fiului drept al nodului initial, devine fiu drept pentru acesta, dupa re-echilibrare**

3. **Noul fiu stang din subarborele drept va contine adresa nodului dezechilibrat initial.**

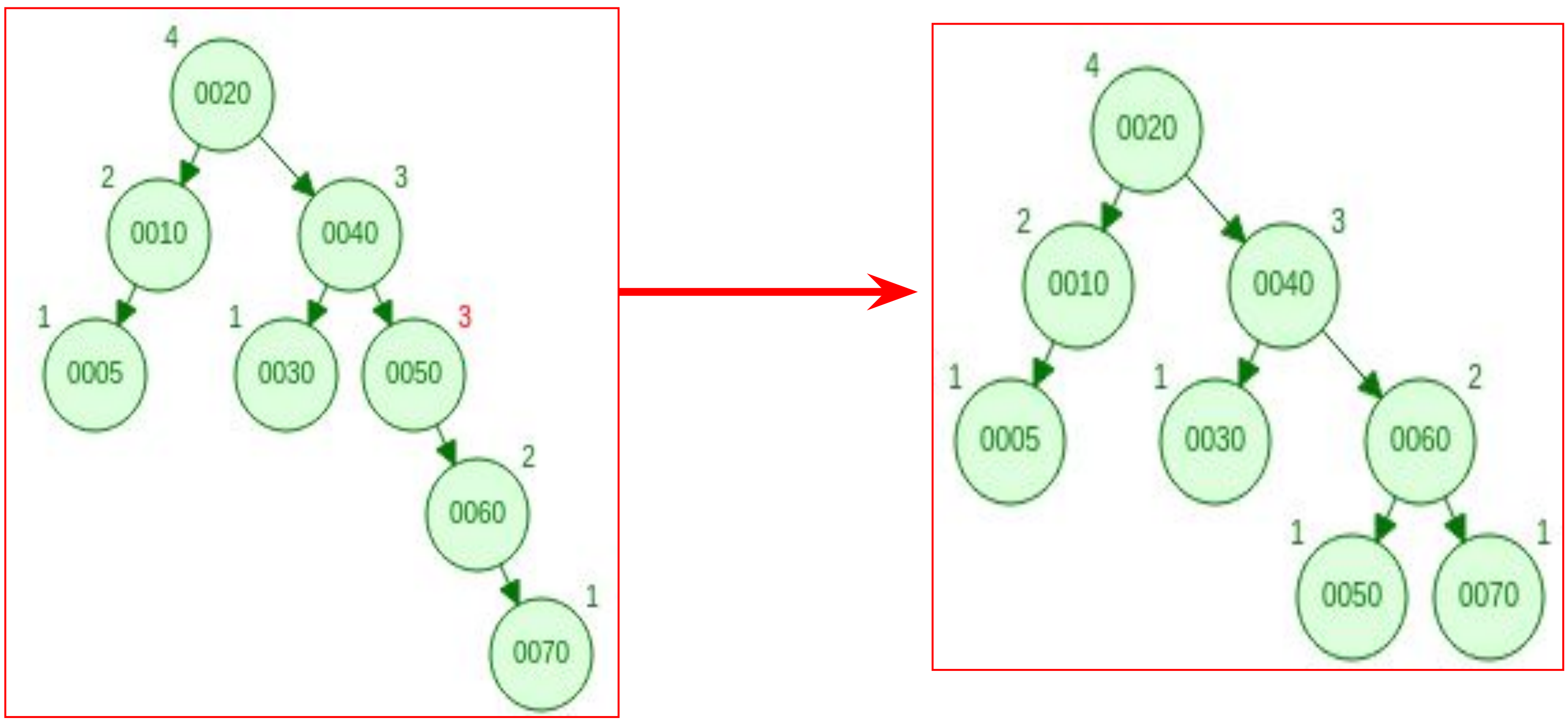
4. **Se recalculeaza factorii de echilibru (exista si alte metode…)**

Arbori binari de cautare echilibrati AVL

Rotatie simpla la stanga Exemplu

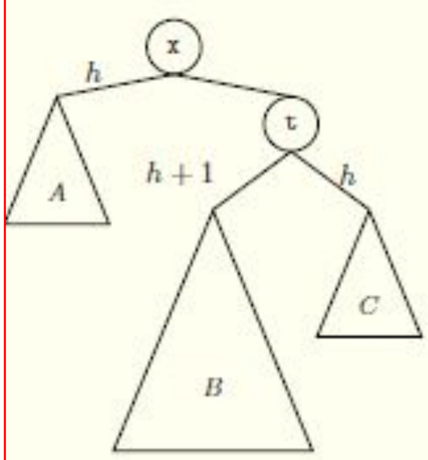
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Se insereaza, pe rand: 40, 20, 50, 10, 30, 5, 60, 70

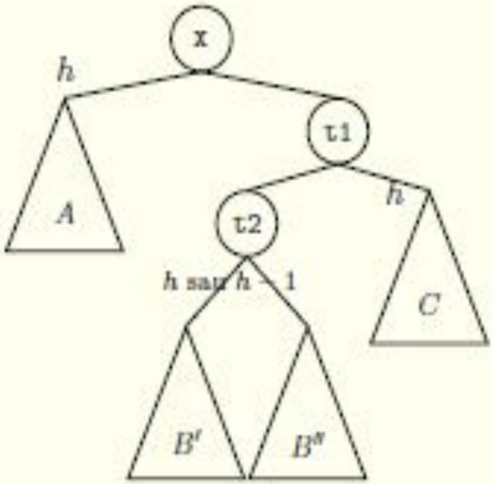


Arbori binari de cautare echilibrati AVL

3. Rotatie dubla Dreapta - Stanga



(a) Subarborele x iniţial

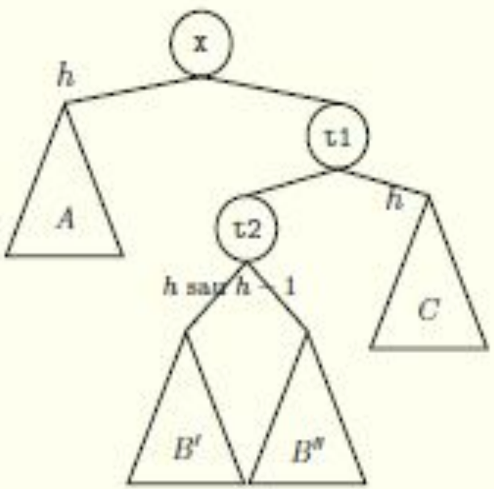


(b) Detaliere

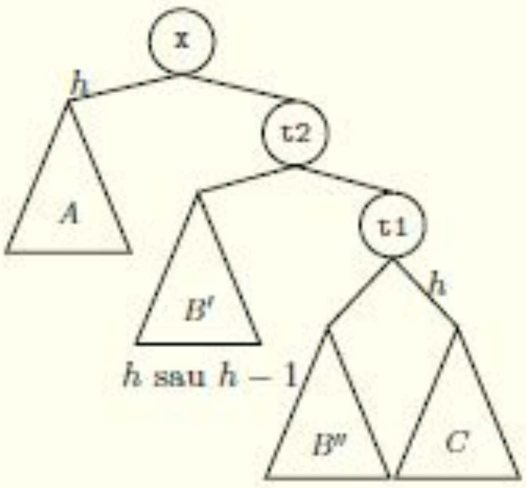
```
void rotatie_dreapta_stanga(nod *&p)
{
    rotatie_dreapta(p->right);
    rotatie_stanga(p);
}
```

Arbori binari de cautare echilibrati AVL

Rotatie dubla Dreapta - Stanga



(b) Detaliere

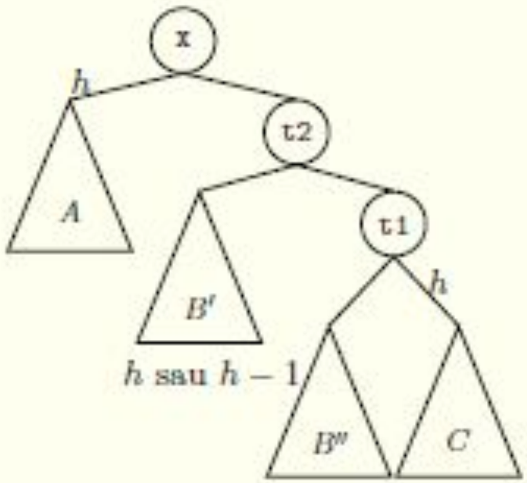


(c) După rotaţia dreapta în t1 (x->right)

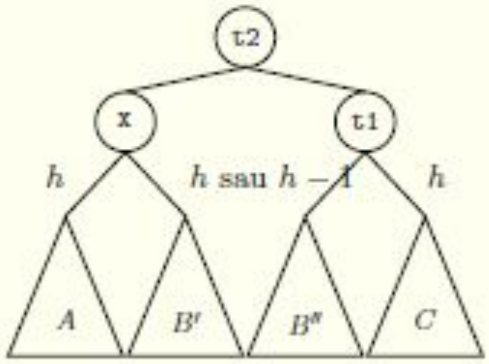
```
void rotatie_dreapta_stanga(nod *&p)
{
    rotatie_dreapta(p->right);
    rotatie_stanga(p);
}
```

Arbori binari de cautare echilibrati AVL

Rotatie dubla Dreapta - Stanga



(c) După rotaţia dreapta în t1 (x->right)



(d) După rotaţia stânga în x

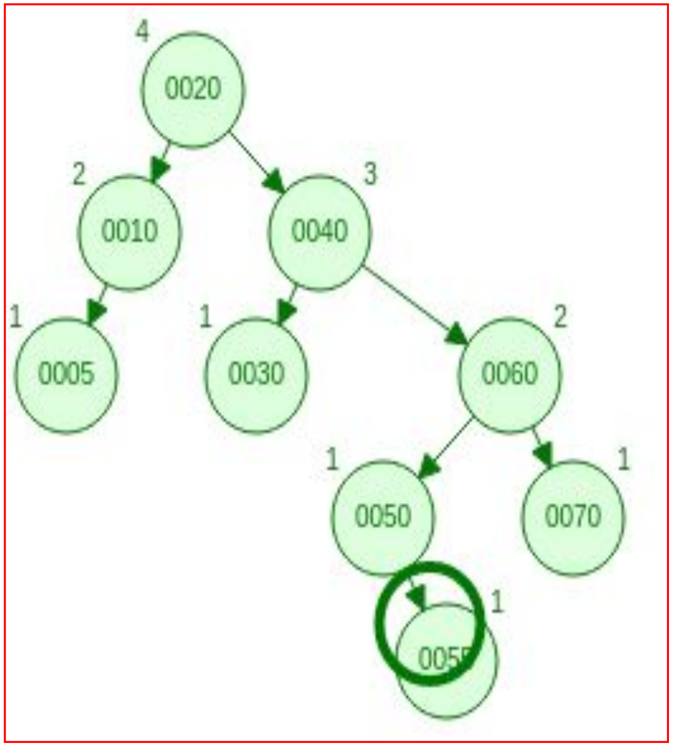
```
void rotatie_dreapta_stanga(nod *&p)
{
    rotatie_dreapta(p->right);
    rotatie_stanga(p);
}
```

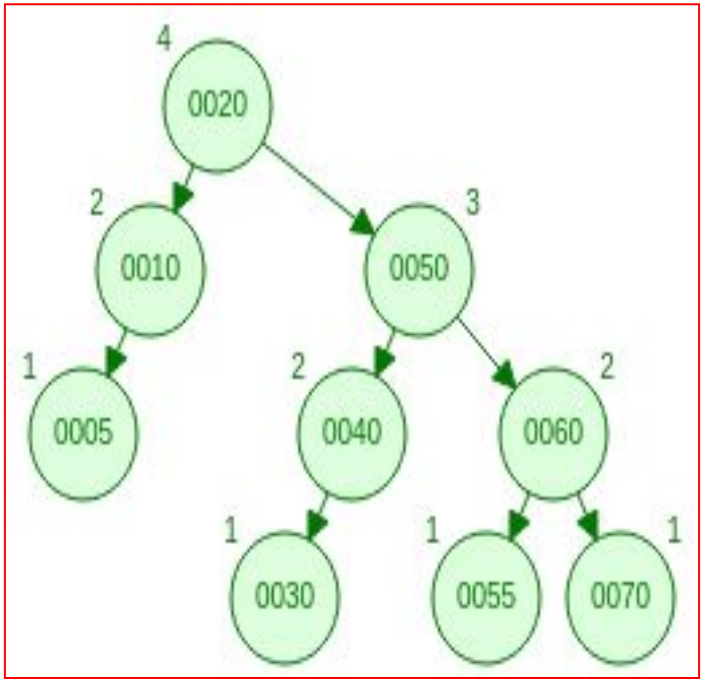
Arbori binari de cautare echilibrati AVL

Rotatie dubla Dreapta – Stanga Exemplu

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

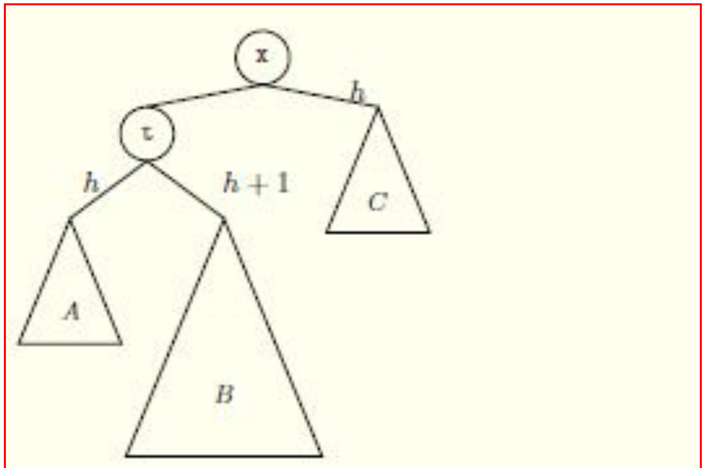
Se insereaza, pe rand: 40, 20, 50, 10, 30, 5, 60, 70, 55



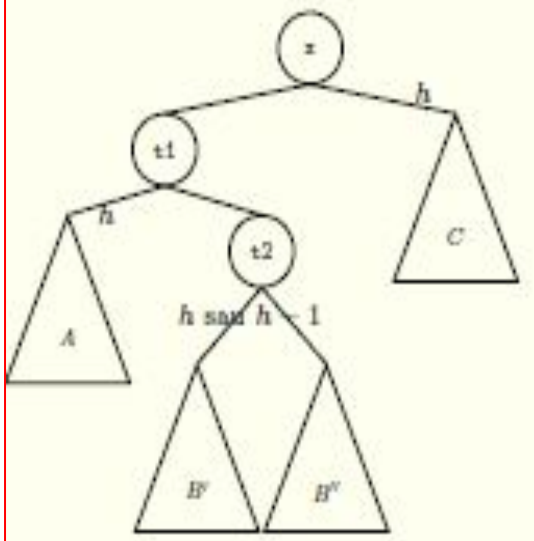


Arbori binari de cautare echilibrati AVL

4. Rotatie dubla Stanga - Dreapta



(a) Subarborele x iniţial

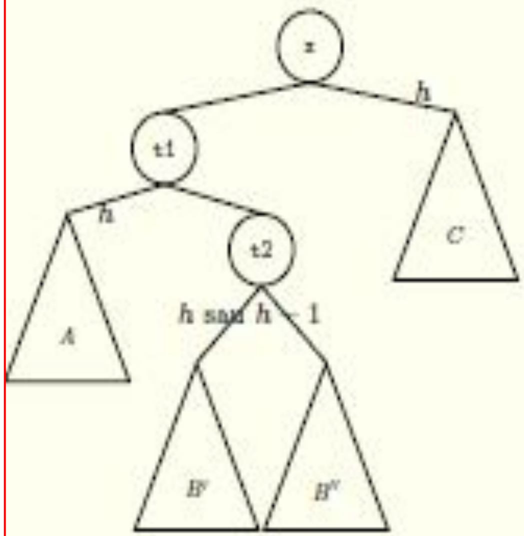


(b) Detaliere

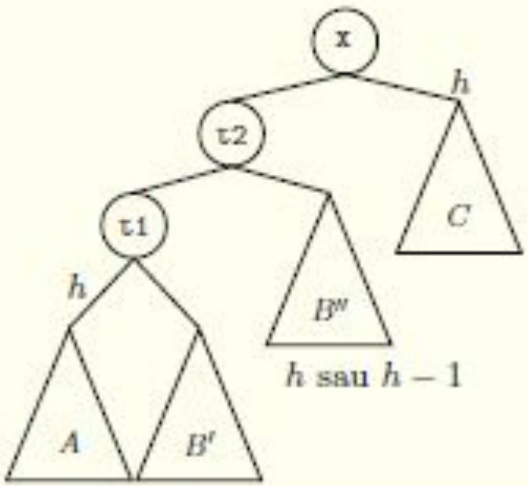
```
void rotatie_stanga_dreapta(nod *&p)
{
    rotatie_stanga(p->left);
    rotatie_dreapta(p);
}
```

Arbori binari de cautare echilibrati AVL

Rotatie dubla Stanga - Dreapta



(b) Detaliere

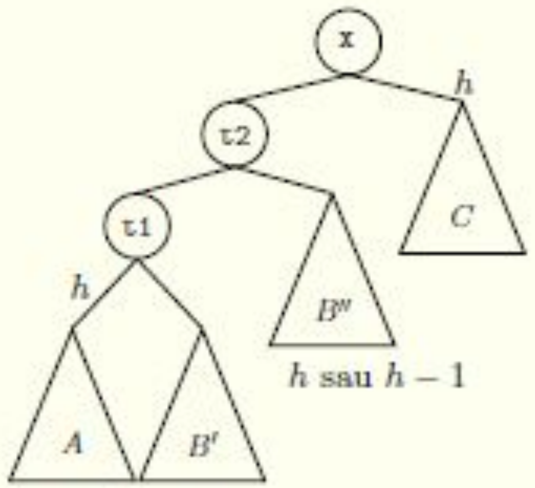


(c) După rotaţia stânga în t1 (x->left)

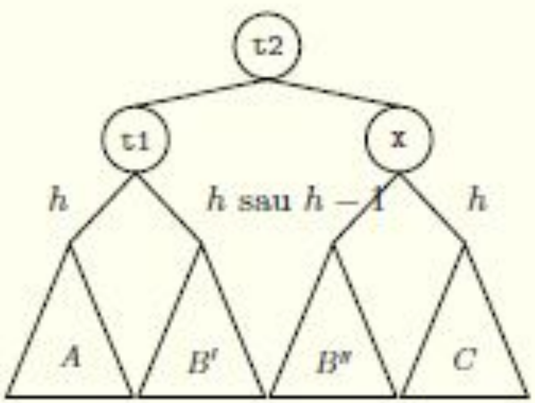
```
void rotatie_stanga_dreapta(nod *&p)
{
    rotatie_stanga(p->left);
    rotatie_dreapta(p);
}
```

Arbori binari de cautare echilibrati AVL

Rotatie dubla Stanga - Dreapta



(c) După rotaţia stânga în t1 (x->left)



(d) După rotaţia dreapta în x

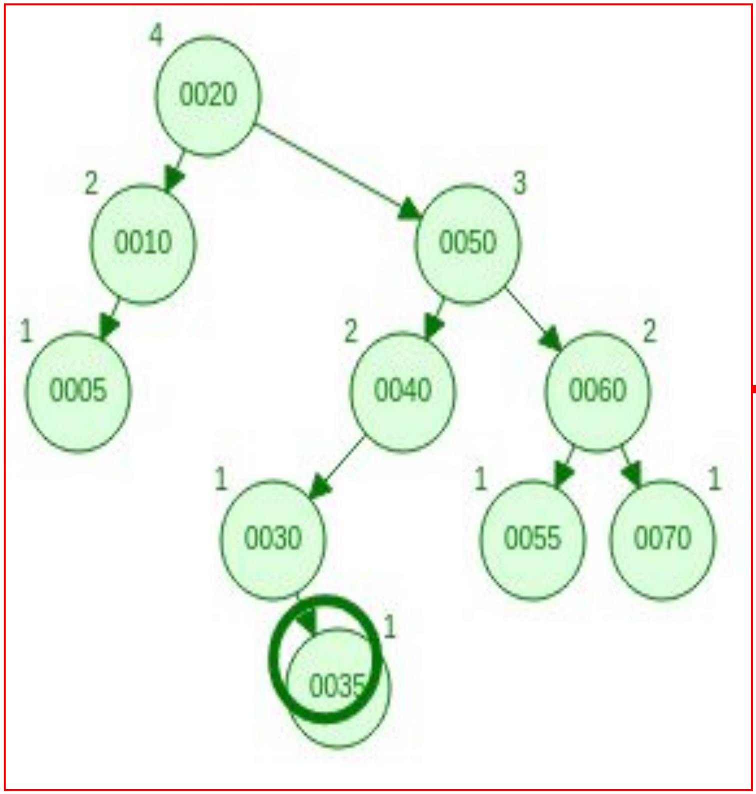
```
void rotatie_stanga_dreapta(nod *&p)
{
    rotatie_stanga(p->left);
    rotatie_dreapta(p);
}
```

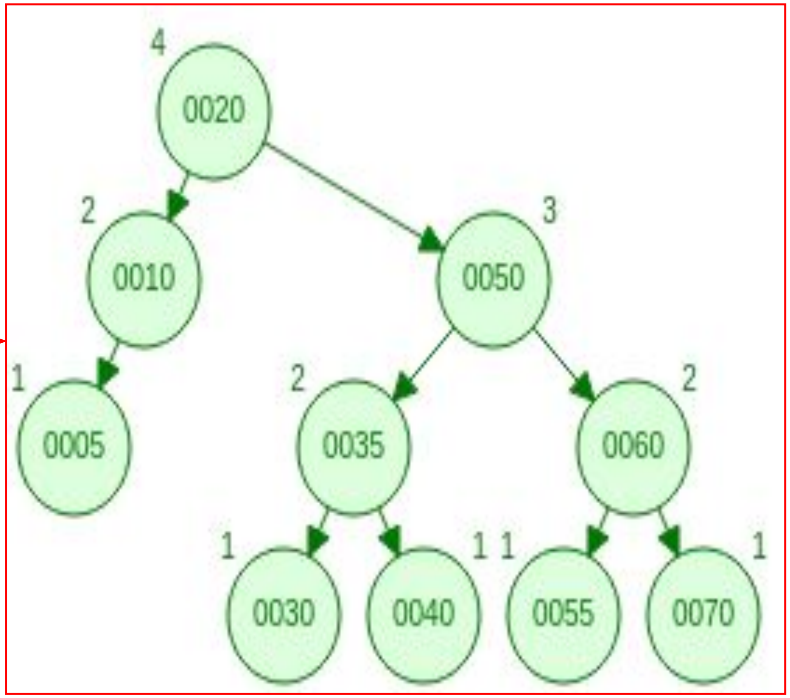
Arbori binari de cautare echilibrati AVL

Rotatie dubla Stanga – Dreapta Exemplu

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Se insereaza, pe rand: 40, 20, 50, 10, 30, 5, 60, 70, 55, 35

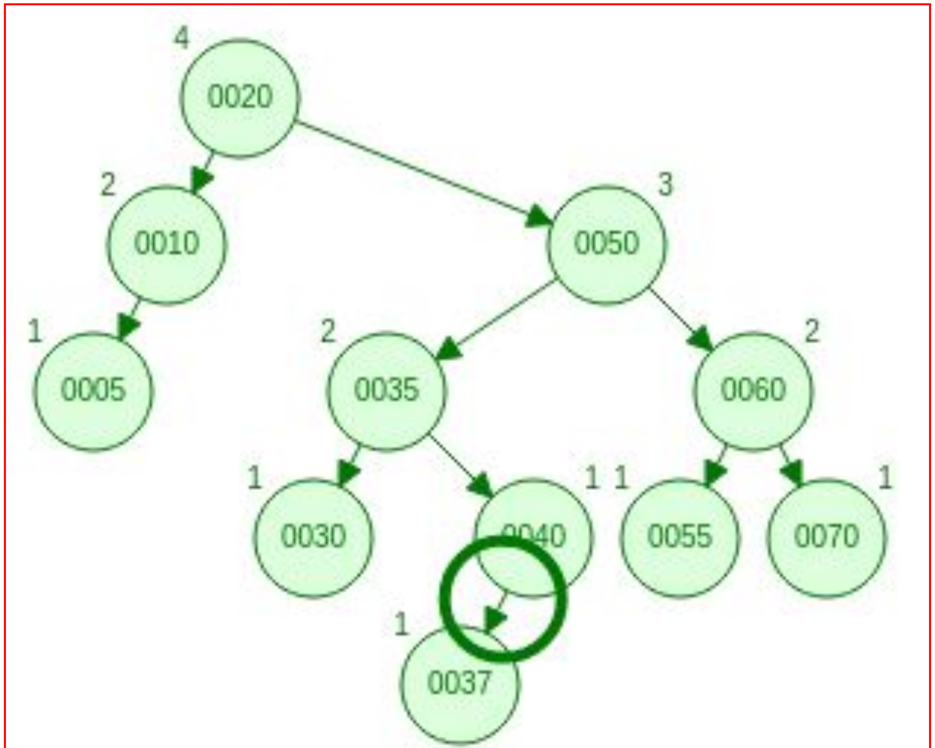




Arbori binari de cautare echilibrati AVL

Aplicati re-echilibrarea pentru inserarea cheii 37

Se insereaza, pe rand: 40, 20, 50, 10, 30, 5, 60, 70, 55, 35, 37

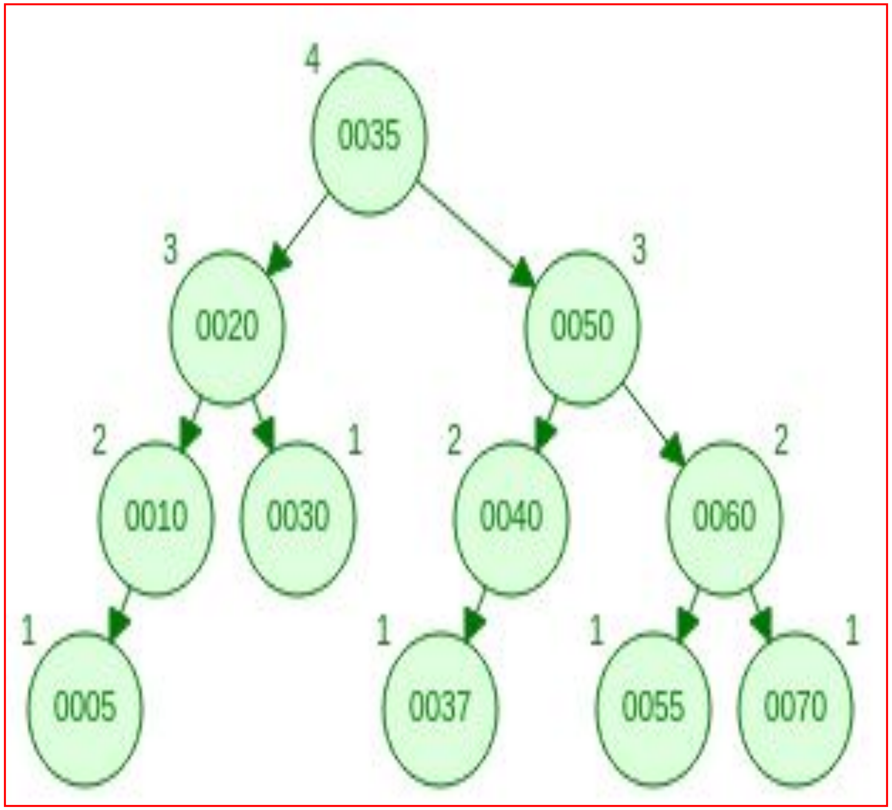


https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Arbori binari de cautare echilibrati AVL

Aplicati re-echilibrarea pentru inserarea cheii 37

Se insereaza, pe rand: 40, 20, 50, 10, 30, 5, 60, 70, 55, 35, 37



Arbori binari de cautare echilibrati AVL

Echilibrarea

```
void echilibreaza(nod *&p)
{
    if (p->bal == -2)
        if (p->left->bal == 1)
            rotatie_stanga_dreapta(p);
        else
            rotatie_dreapta(p);
    else if (p->bal == 2)
        if (p->right->bal == -1)
            rotatie_dreapta_stanga(p);
        else
            rotatie_stanga(p);
}
```

Factori de ech

Nod	Descen-dent	Rotatie
-2	Left / 1	SD
-2	Left / -1	SS
2	Right / -1	DS
2	Right / 1	DD

Arbori binari de cautare echilibrati AVL

Inserarea unei valori

```
if (p == NULL)
{
    p = new nod;
    p->info = val;
    p->bal = 0;
    p->left = NULL;
    p->right = NULL;
    return true;
}

if (p->info == val) return false;
```

Daca valoarea nu exista deja, se introduce si se stabileste factorul de echilibru nul.

Arbori binari de cautare echilibrati AVL

Inserarea unei valori

```
if (p->info > val)
    if (inserare_recursiva(p->left,val)==true)
        p->bal--;
    else
        return false;
else if (inserare_recursiva(p->right,val)==true)
    p->bal++;
else
    return false;
```

Se cauta recursiv conform regulii dintr-un a.b.c.

Obs. Daca valoarea se cauta in subarborele drept, atunci factorul de echilibru se incrementeaza, iar daca se cauta in subarborele stang, valoarea se decrementeaza.

Arbori binari de cautare echilibrati AVL

Inserarea unei valori

```
if (p->info > val)
    if (inserare_recursiva(p->left,val)==true)
        p->bal--;
    else
        return false;
else if (inserare_recursiva(p->right,val)==true)
    p->bal++;
else
    return false;
```

```
if (p->bal == 0 || p->bal == 1 || p->bal == -1)
    return true;
else
{
    echilibreaza(p);
    return false;
}
```

La finalul cautarii, daca e necesar, se aplica operatia de re-echilibrare

Arbori binari de cautare echilibrati AVL

Stergerea unei valori intr-un AVL

Etape:

1. se identifică nodul de sters in a.b.c.

2. se sterge nodul conform regulilor unui a.b.c.:

- frunza se sterge efectiv;
- nodul cu un singur descendet □ fiul il inlocuieste in structura;
- nodul are 2 descendenti □ se inlocuieste cu valoarea cea mai mica din subarborele drept.

3. Sunt analizate toate nodurile in sens invers, pana la radacina si se rezolva situatiile de dezechilibru conform celor 4 tipuri anterior prezentate.

Operatia de stergere se incheie cand au fost verificate toate locatiile de dezechilibru posibil.

Arbori binari de cautare echilibrati AVL

Stergerea unei valori

```
if (p==NULL) return false;

if (p->info > val)
    if (stergere_recursiva(p->left,val) == true)
        p->bal++;
    else
        return false;
else if (p->info < val)
    if (stergere_recursiva(p->right,val) == true)
        p->bal--;
    else
        return false;
```

Se cauta recursiv conform regulii dintr-un a.b.c.

Obs. Daca valoarea se cauta in subarborele drept, atunci factorul de echilibru se incrementeaza, iar daca se cauta in subarborele stang, valoarea se decrementeaza.

Arbori binari de cautare echilibrati AVL

Stergerea unei valori

Stergerea unui nod care are cel mult un descendent

```
else if (p->left == NULL || p->right == NULL)
{
    if (p->left != NULL)
    {
        p->info = p->left->info;
        p->left = NULL;
        p->bal++;
        return true;
    }
    else if (p->right != NULL)
    {
        p->info = p->right->info;
        p->right = NULL;
        p->bal--;
        return true;
    }
```

Obs. Daca nodul este frunza, se sterge efectiv (primeste valoarea NULL)

Arbori binari de cautare echilibrati AVL

Stergerea unei valori

```
nod *minim(nod *x)
{
    while (x->left)
        x = x->left;
    return x;
}
```

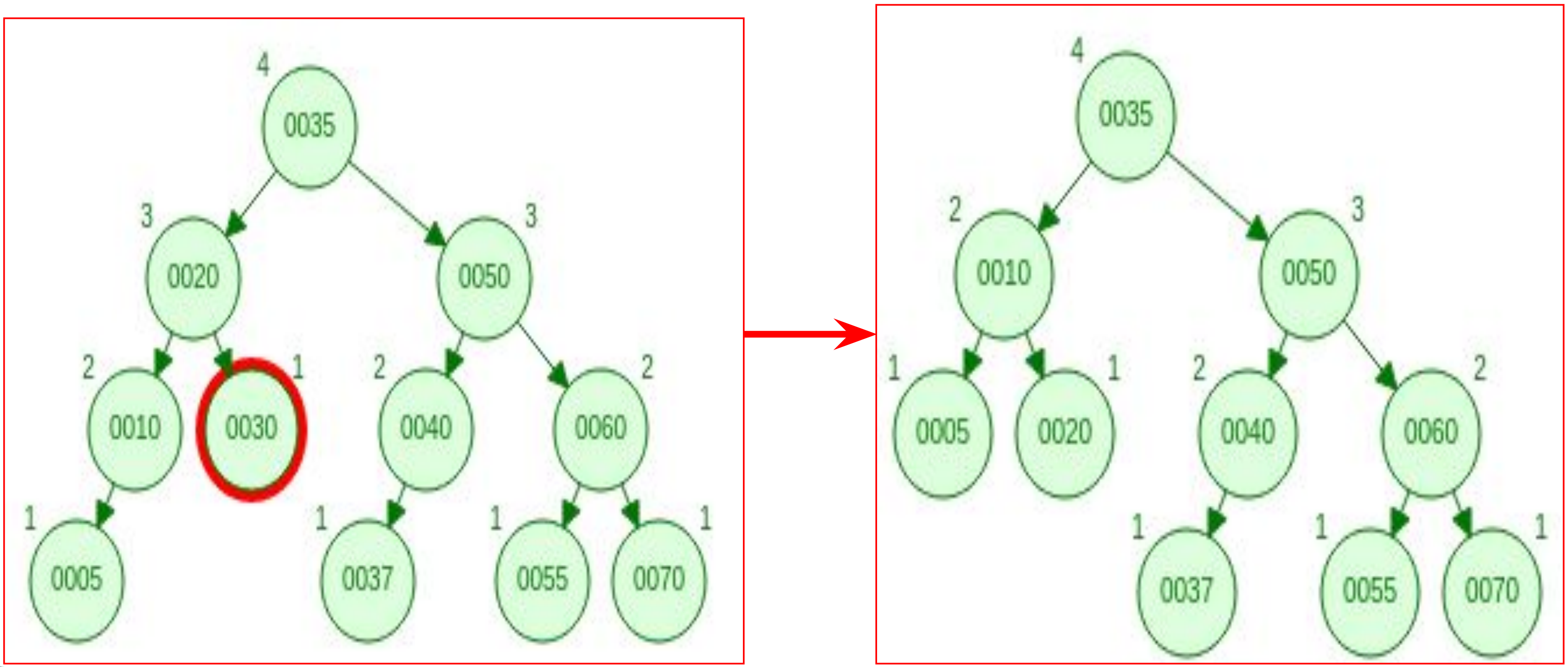
```
else
{
    nod *y = minim(p->right);
    p->info = y->info;
    if (stergere_recursiva(p->right,y->info) == true)
        p->bal--;
    else
        return false;
}
if (p->bal == 2 || p->bal == -2)
    echilibreaza(p);
if (p->bal == 0)
    return true;
else
    return false;
```

Stergerea unui nod care are doi descendenti

Arbori binari de cautare echilibrati AVL

Exemplu – stergerea valorii 30

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

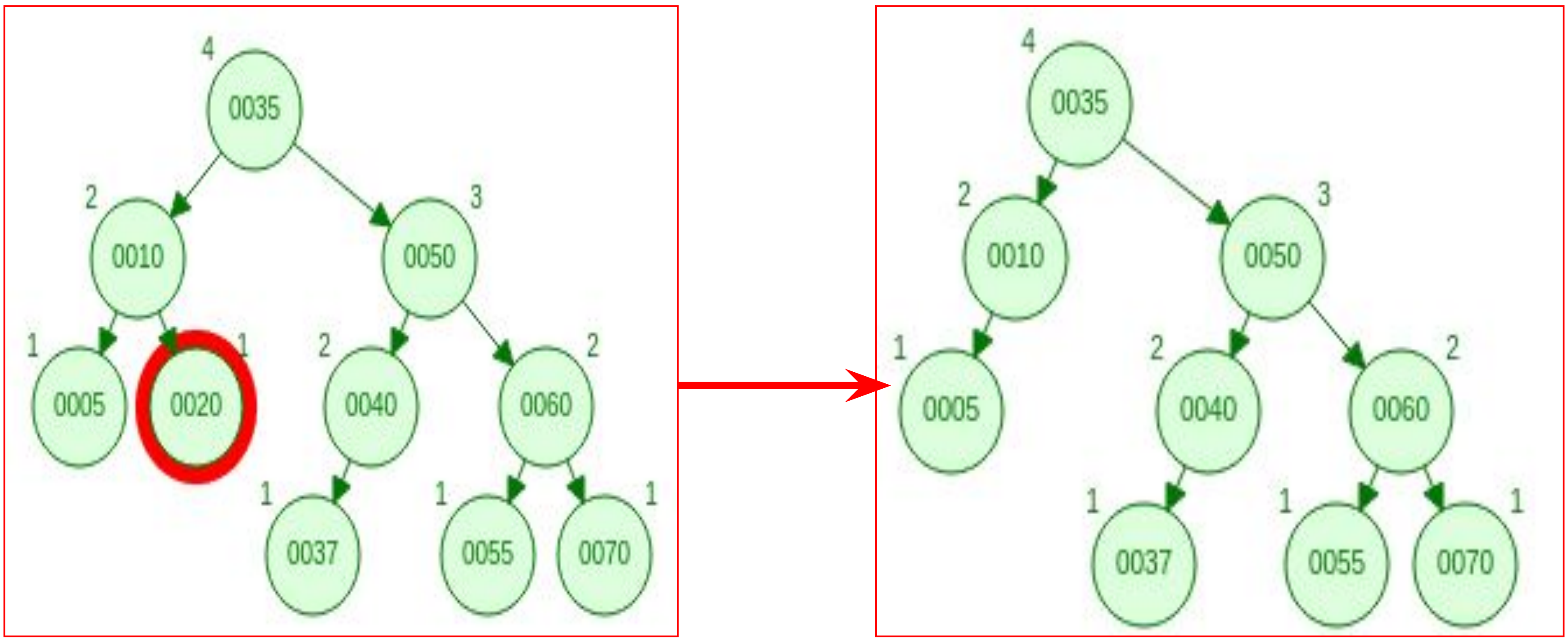


Facultatea de Matematica si Informatica
Universitatea din Bucuresti

Arbori binari de cautare echilibrati AVL

Exemplu – stergerea valorilor 20 si 5

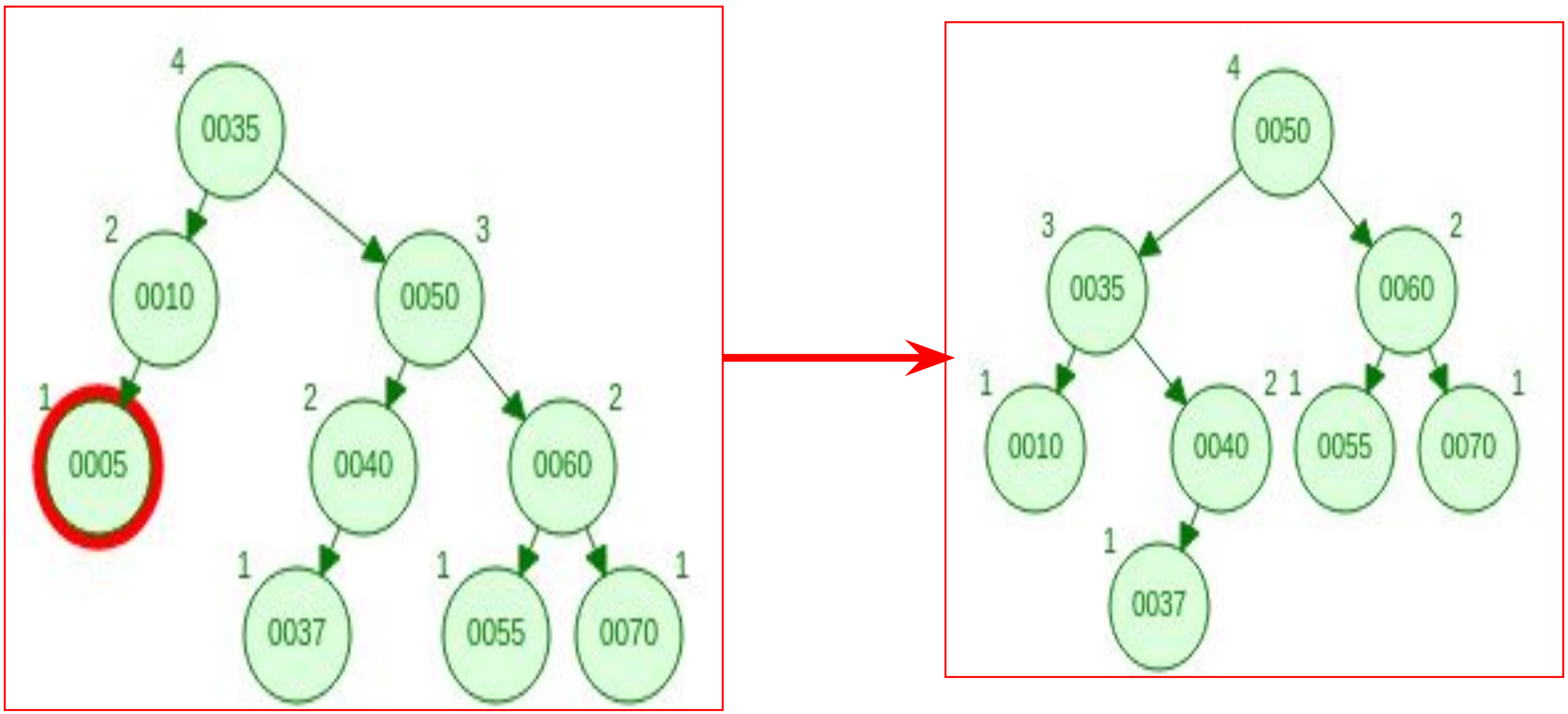
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html



Arbori binari de cautare echilibrati AVL

Exemplu – stergerea valorilor 20 si 5

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html



Arbori binari de cautare echilibrati AVL

Costuri

http://software.ucv.ro/~mburicea/lab6ASDr.pdf

Arborii AVL - alternativa putin costisitoare la arborii binari obisnuiti.

-proprietatea de echilibru AVL a unui arbore binar ordonat duce la cautari mult mai rapide decat in cazul unui arbore binar ordonat obisnuit, datorita inaltimii mai mici.

S-a demonstrat ca un arbore echilibrat AVL va avea intotdeauna inaltimea cuprinsa intre [log2N+1] si [1,43·log2N+1], unde N reprezinta numarul de chei din arbore

Abordare OOP pentru AVL si Arbori Rosu si Negru

http://www.ionivan.ro/ANUL-UNIVERSITAR%202010-2011/ZZZZ-cartea%20structuri%20date/F00017000-arboriechilibrati.pdf

Curs 7 - Cuprins

3. Structuri arborescente

Arbori binari stricti. Proprietati matematice. Aplicatii

Teorema AVL. Margine superioara si margine inferioara pentru inaltimea unui arbore binar echilibrat AVL

Sursa: – R. Ceterchi: "Structuri de date si Algoritmi. Aspecte matematice si aplicatii", Editura Univ. din Bucuresti, 2001

Arbori binari stricti

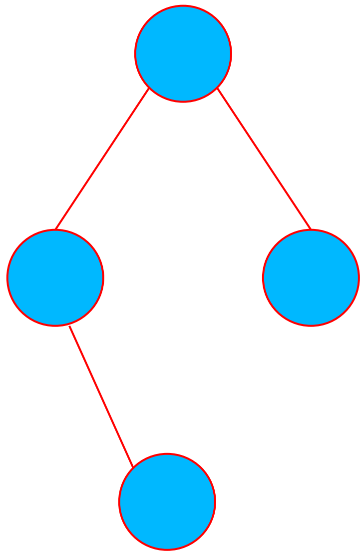
Un **arbore binar strict** este un arbore binar ın care fiecare nod are **fie nici un fiu, fie exact doi fii**.

Nodurile **cu doi copii** se vor numi **noduri interne**, iar cele fara copii se vor numi **noduri externe** sau **frunze**.

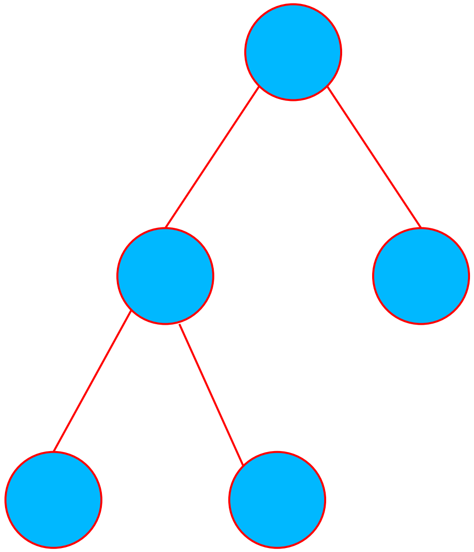
Nodurile externe pot fi de alt tip decat nodurile interne.

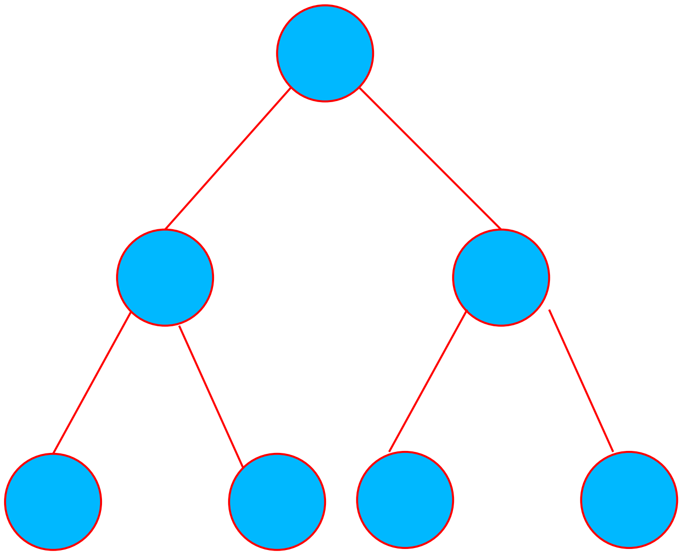
Arbori binari stricti

Arbore binar nestrict



Arbori binari stricti



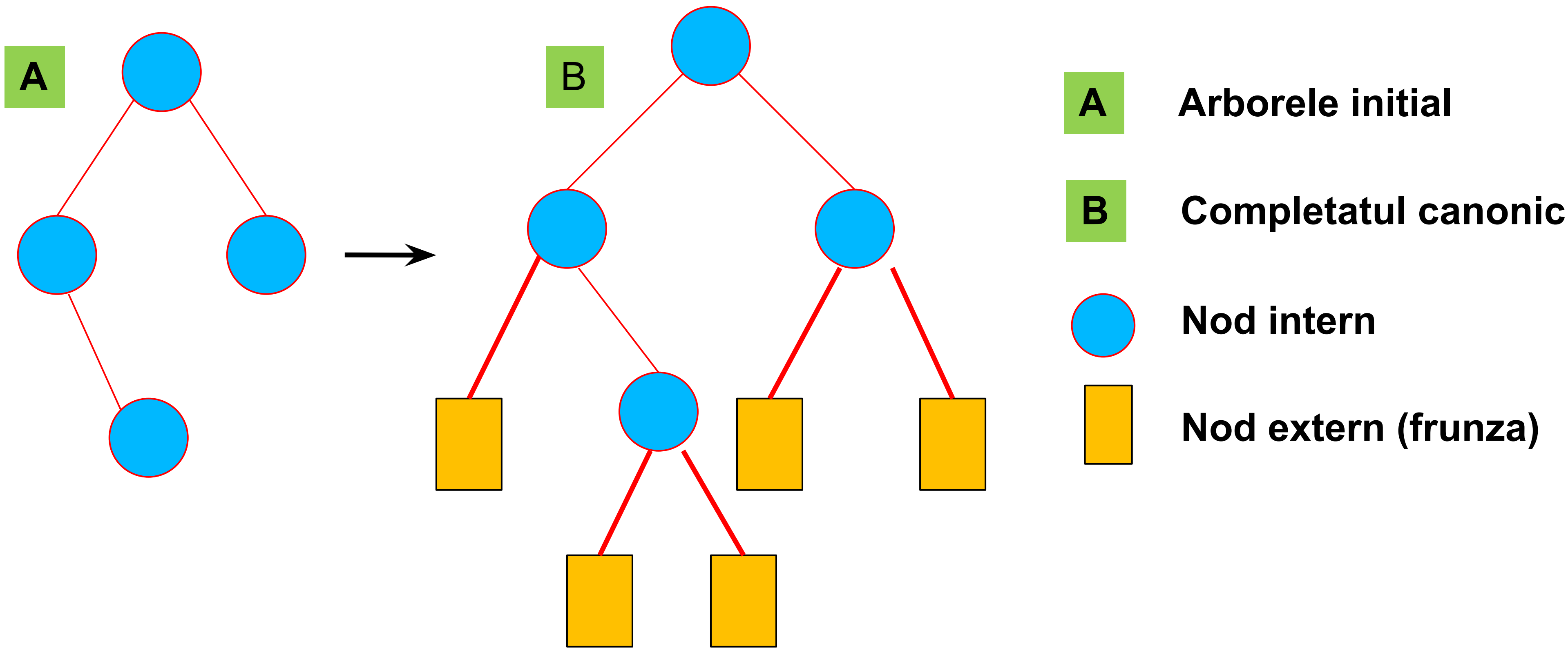


Arbore binar strict si complet pe niveluri

Arbori binari stricti

Completare canonica a unui arbore binar oarecare la unul strict

Fiecare fiu vid se inlocuieste cu un nod de tip special nodurile arborelui initial devin toate noduri interne, iar cele adaugate canonic, vor fi frunze.



Conventie – noduri interne = cercuri, noduri externe = dreptunghiuri.

Arbori binari stricti

Exemple de aplicatii ale structurii de arbore binar strict

Abreviere: arbore binar strict (abs)

- reprezentari de expresii aritmetice cu operatori binari

- algoritmi

- proceduri de decizie

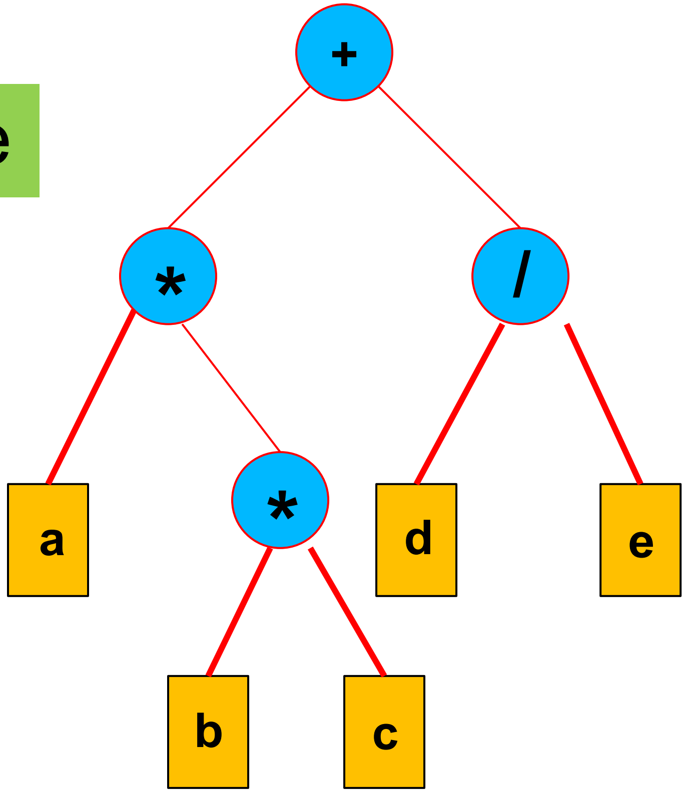
- codificare binara

Arbori binari stricti

Exemple de aplicatii ale structurii de arbore binar strict

- reprezentari de expresii aritmetice cu operatori binari

E = a * b * c + d / e

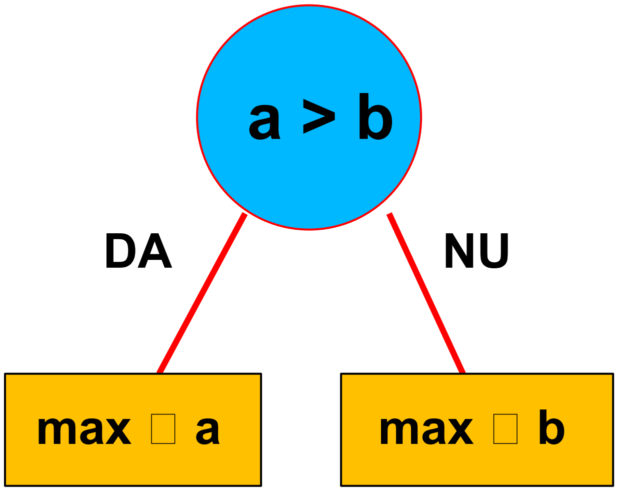


Arbori binari stricti

Exemple de aplicatii ale structurii de arbore binar strict

- proceduri de decizie

Daca (a>b) atunci
    max □ a
Altfel
    max □ b



Facultatea de Matematica si Informatica
Universitatea din Bucuresti

Arbori binari stricti

Exemple de aplicatii ale structurii de arbore binar strict

- algoritmi

Ordonarea a 3 numere

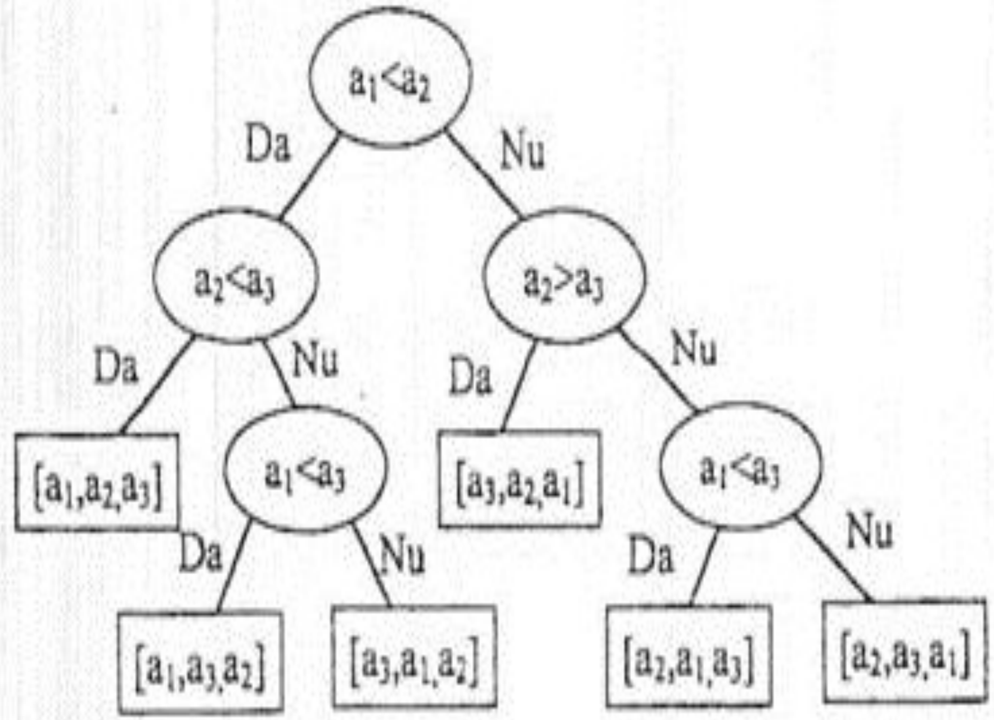


Fig.4.1.6 Arbore binar strict asociat sortării mulțimii $\{a_1, a_2, a_3\}$.

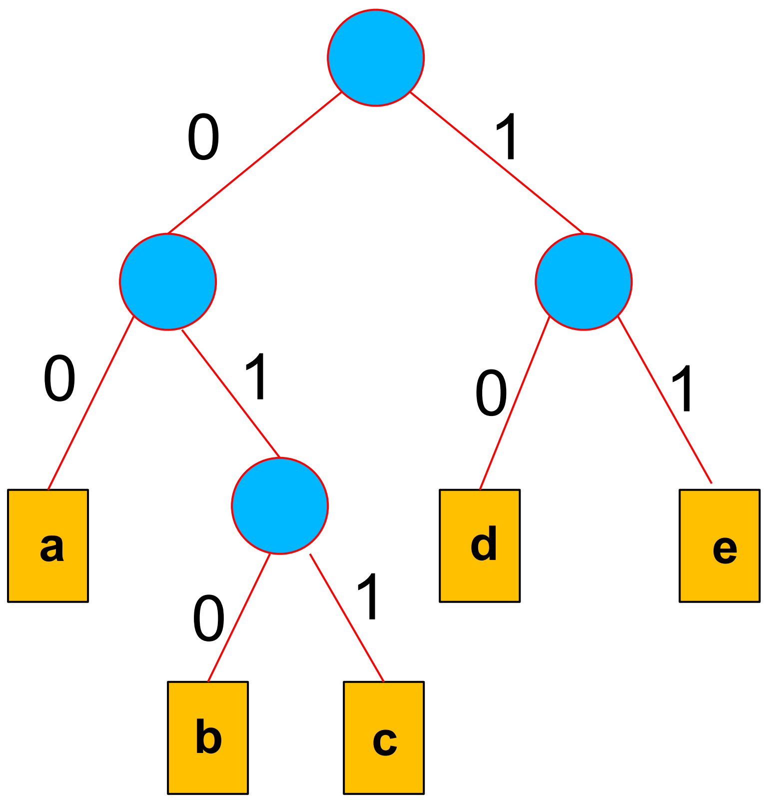
Sursa: R. Ceterchi - “Structuri de date si Algoritmi. Aspecte matematice si Aplicatii (2001)”

Arbori binari stricti

Exemple de aplicatii ale structurii de arbore binar strict

- codificare binara

Coduri binare peste alfabetul {a, b, c, d, e} asociat abs.



a – 00
b – 010
c – 011
d – 10
e – 11

Arbori binari stricti

N_E = numărul nodurilor externe şi
N_I = numărul nodurilor interne ale unui arbore binar strict

Propoziţia 1. Într-un arbore binar strict, numărul nodurilor externe şi al celor interne sunt legate prin relaţia:

$$N_E = N_I + 1.$$

Demonstratie.
Numaram in 2 moduri arcele dintr-un abs.
1. Din orice nod intern pornesc 2 arce :

Nr arce = $2 * N_I$. *(1)*

2. In fiecare nod din arbore (cu exceptia radacinii) intra un singur arc:

Nr arce = $N_I + N_E - 1$. *(2)*

Din (1) si (2) □ $2 * N_I = N_I + N_E - 1$ □ $N_I = N_E - 1$ □ $N_E = N_I + 1$

Arbori binari stricti

Proprietati ale abs

lungime externă a unui arbore binar strict = suma lungimilor drumurilor de la rădăcină până la fiecare nod extern.

E = mulţimea frunzelor

lungime internă a unui arbore binar strict = suma lungimilor drumurilor de la rădăcină la toate nodurile interioare.

I = mulţimea nodurilor interioare

$$L_E = \sum_{x \in E} l(r, x) \qquad\qquad L_I = \sum_{y \in I} l(r, y) \, .$$

r este rădăcina, *l(r, x)* lungimea drumului de la *r* la nodul *x*.
(Drumul de la rădăcină la un nod se măsoară în număr de arce.)

Arbori binari stricti

Propoziţia 2. Într-un arbore binar strict este adevarata relaţia:

$$L_E = L_I + 2N_I \ .$$

Demonstratie.

inducţie după n = numărul total de noduri ale unui arbore binar strict. (Ştim că n nu poate lua orice valoare din mulţimea numerelor naturale.)

(a) *n = 1, 3, 5 trivial ...*

(b) Presupunem că relaţia $L_E = L_I + 2N_I$ este adevărată pentru orice arbore binar strict care are un număr total de noduri mai mic decât un număr natural m dat, deci pentru orice $n < m$.

Arbori binari stricti

Propoziţia 2. Într-un arbore binar strict este adevarata relaţia:

$$L_E = L_I + 2N_I \ .$$

Demonstratie (cont.)

Fie un arbore binar strict *T* cu numărul total de noduri *n, n* < *m.*

T este **compus** dintr-un **nod rădăcină**, intern, şi fiii săi **stâng** şi **drept**, T^s *şi* T^d, care sunt la rândul lor subarbori binari stricţi.

Notam cu N_I^s, N_E^s, L_I^s, L_E^s caracteristicile lui T^s şi cu N_I^d, N_E^d, L_I^d, L_E^d caracteristicile lui T^d

Arbori binari stricti

Propoziţia 2. Într-un arbore binar strict este adevarata relaţia:

$$L_E = L_I + 2N_I \,.$$

Demonstratie (cont.)

(1) $N_I = N_I^s + N_I^d + 1$
(2) $N_E = N_E^s + N_E^d$

(3) $L_E = L_E^s + N_E^s + L_E^d + N_E^d$
(4) $L_I = L_I^s + N_I^s + L_I^d + N_I^d$

(5) $L_E^s = L_I^s + 2N_I^s$
(6) $L_E^d = L_I^d + 2N_I^d$

$$L_E = L_E^s + N_E^s + L_E^d + N_E^d = L_I^s + N_I^s + N_E^s + L_I^d + N_I^d + N_I^d + N_E^d = (L_I^s + N_I^s + L_I^d + N_I^d) + (2N_I^s + 1) + (2N_I^d + 1)$$

$$L_E = L_I + 2(N_I^s + N_I^d = 1) = L_I + 2N_I$$

Arbori binari stricti

Propoziţia 3. Într-un arbore binar strict de adâncime *d* avem următoarea inegalitate.

$$N_E \leq 2^d \ .$$

Demonstratie

Inegalitatea din enunţ revine la demonstrarea următoarelor afirmaţii:

1. **Dintre toţi arborii binari stricţi de adâncime dată, *d*, cel cu număr maxim de frunze este cel care are toate frunzele la ultimul nivel (adică *d*).**

2. **Un arbore cu toate frunzele la nivelul *d* are exact 2^d frunze, adică $N_E = 2^d$.**

Arbori binari stricti

Propoziţia 3. Într-un arbore binar strict de adâncime *d* avem următoarea inegalitate.

$$N_E \leq 2^d \ .$$

Demonstratie (cont)

(inductie pt. (2)):

(verif) $d = 0$ trivial
(ipot. ind.) $d = k$, numărul de frunze (aflate toate la nivelul k) este $N_E = 2^k$.

Putem construi dintr-un asemenea arbore, în mod foarte simplu şi direct, un arbore binar strict care are adâncime $d = k + 1$ şi proprietatea că toate frunzele sunt la nivelul $k + 1$.

Arbori binari stricti

Proprietati ale abs

Propoziţia 3. Într-un arbore binar strict de adâncime *d* avem următoarea inegalitate.

$$N_E \leq 2^d \ .$$

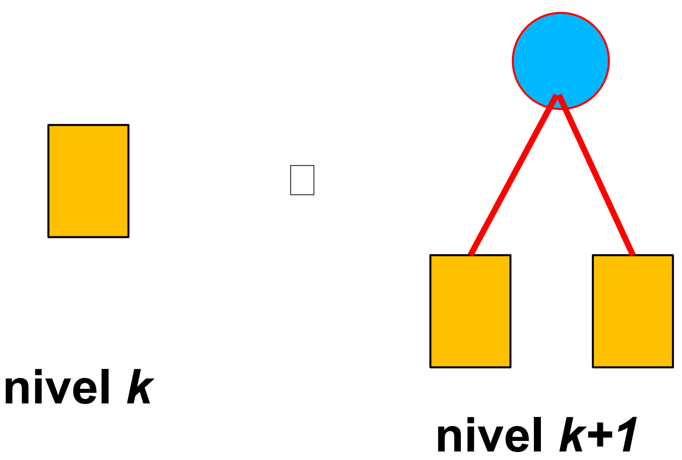
Demonstratie (cont)

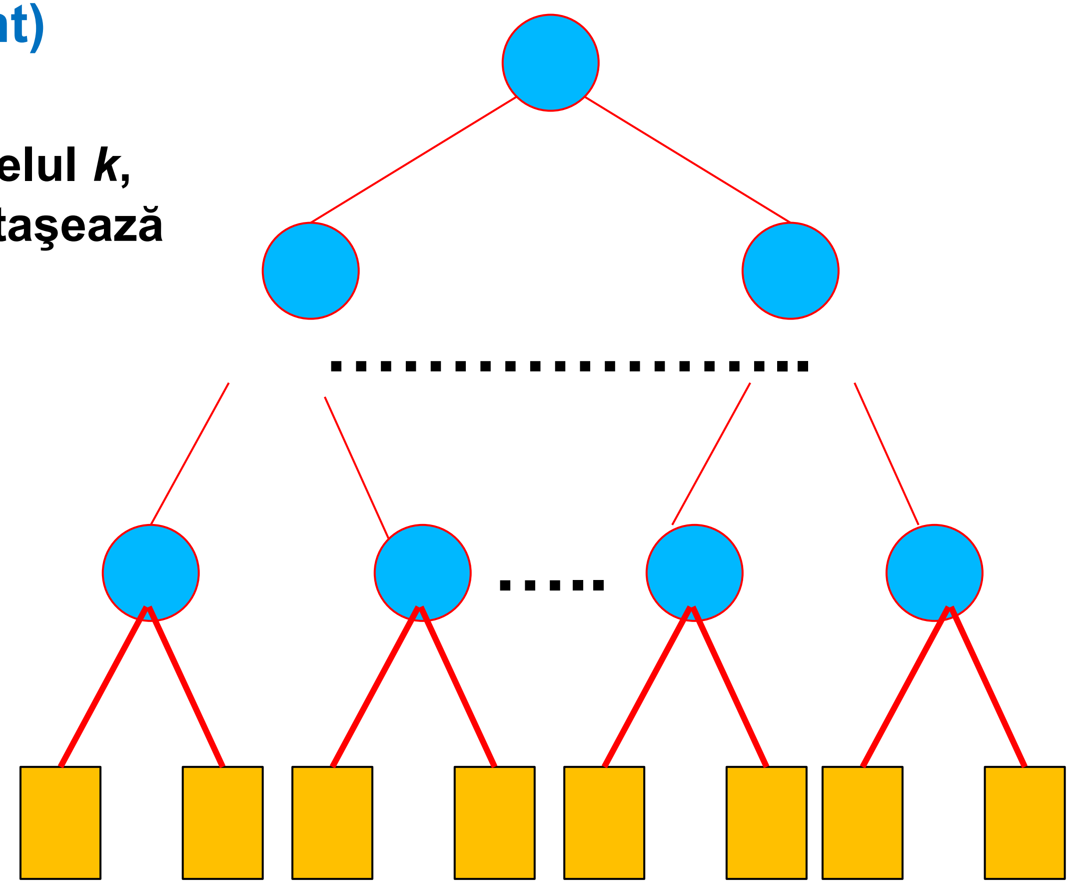
Se înlocuieşte fiecare frunză de la nivelul *k*, cu un nod interior, iar acestora li se ataşează câte *două* frunze, procedeu care produce un arbore cu de două ori mai multe frunze decât precedentul. Deci, pentru arborele de adâncime *k + 1* şi toate frunzele la acest nivel avem $N_E = 2 * 2^k = 2^{k+1}$

Arbori binari stricti

Propoziţia 3. Demonstratie (cont)

Se înlocuieşte fiecare frunză de la nivelul *k*, cu un nod interior, iar acestora li se ataşează câte *două* frunze





Arbori binari stricti

Corolar. Într-un arbore binar strict de adâncime *d* avem inegalitatea

$$d \geq \lceil log_2 N_E \rceil.$$

Propoziţia 4. Dintre toţi arborii binari stricţi cu acelaşi număr de frunze, fixat, N_E, au **lungime externă minimă** aceia cu proprietatea că **frunzele** lor **sunt repartizate pe cel mult două niveluri adiacente.**

Arbori binari stricti

Propoziţia 5. Lungimea externă minimă a unui arbore binar strict cu N_E frunze este dată de formula:

$$L_E^{min} = N_E \lfloor \log_2 N_E \rfloor + 2(N_E - 2^{\lfloor \log_2 N_E \rfloor}).$$

Demonstratie

Fie d = h(T)= inaltimea unui a.b.s. T pe care se atinge lungimea externa minima, avem 2 cazuri (cf. Prop. 4):

Cazul (a): Toate frunzele sunt la acelasi nivel, d = h(T), daca $N_E = 2^d$.

$$L_E = N_E * d = N_E * log_2 N_E,$$

Unde în partea dreaptă avem exact valoarea expresiei din enunţ, deoarece

$$N_E - 2^{log2\ NE} = N_E - N_E = 0.$$

Arbori binari stricti

Propoziţia 5. Lungimea externă minimă a unui arbore binar strict cu N_E frunze este dată de formula:

$$L_E^{min} = N_E \lfloor \log_2 N_E \rfloor + 2(N_E - 2^{\lfloor \log_2 N_E \rfloor}).$$

Demonstratie (cont).

Cazul (b): Frunzele nu sunt toate la acelasi nivel. Dar atunci ele sunt repartizate doar pe nivelurile d - 1 (fie y nr. de frunze de la acest nivel) si d (fie 2x nr. de frunze de la acest nivel, x= nr. de noduri interne de la nivelul d).

Se rezolva sistemul:

$$\begin{cases} x + y = 2^{d-1} & (1) \\ x + y = N_E & (2) \end{cases}$$

Avem

$$\begin{cases} \text{nr. de frunze la nivelul } d-1 = y = 2^d - N_E, \\ \text{nr. de frunze la nivelul } d = 2x = 2N_E - 2^d. \end{cases}$$

Arbori binari stricti

Propoziţia 6. Într-un arbore binar strict avem următoarea inegalitate

$$L_E^{medie} \geq \lfloor \log_2 N_E \rfloor.$$

Demonstratie

Prin lungime medie înţelegem media raportată la numărul de frunze. Deoarece am estimat în Propoziţia 5 lungimea minimă, putem estima acum media ei şi obţinem

$$L_E^{min}/N_E \geq \lfloor \log_2 N_E \rfloor + 2(N_E - 2^{\lfloor \log_2 N_E \rfloor})/N_E.$$

$$L_E^{medie} \geq L_E^{min}/N_E > \lfloor \log_2 N_E \rfloor.$$

Arbori binari stricti si echilibrati AVL

Teorema AVL

Margine superioara si margine inferioara pentru inaltimea unui arbore binar echilibrat AVL

Fie T un arbore binar strict si echilibrat AVL, cu n noduri interne. Fie h(T) inaltimea lui.

Avem: $\log_2(n + 1) <= h(T) <= 1.4404 * \log_2(n + 2) – 0.328$

Echivalent. Sunt satisfacute urmatoarele inegalitati:

(1) $h(T) >= \log_2(n + 1)$.
(2) $h(T) <= (1/\log_2\Phi)\log_2(n + 2) + (\log_2 5 /2\log_2\Phi - 2)$,

unde $\Phi = (1+sqrt(5))/2$

Sursa: R. Ceterchi - “Structuri de date si Algoritmi. Aspecte matematice si Aplicatii (2001)”

Arbori binari stricti si echilibrati AVL

Teorema AVL

Margine superioara si margine inferioara pentru inaltimea unui arbore binar echilibrat AVL

Fie T un arbore binar strict si echilibrat AVL, cu n noduri interne. Fie h(T) inaltimea lui.

Avem: $\log_2(n + 1) <= h(T) <= 1.4404 * \log_2(n + 2) - 0.328$

Demonstratie:

Inegalitatea (1) este adevarata pentru a.b.s. in general (rezulta din Corolarul de la Prop. 3).

Pentru a dem. ineg. (2) construim o clasa particulara de a.b.s. si echil. AVL, **arborii Fibonacci**.

Arbori binari stricti si echilibrati AVL

Teorema AVL

Margine superioara si margine inferioara pentru inaltimea unui arbore binar echilibrat AVL

Arborii Fibonacci

Numerele Fibonacci (de ordinul 1): $F_1 = F_2 = 1$ şi relaţia de recurenţă $F_{n+2} = F_{n+1} + F_n$, pentru $n \geq 1$.

Formula Binet pentru numere Fibonacci: $F_n = (1/\sqrt{5})(\phi^n - \bar{\phi}^n)$, unde $\phi = (1+\sqrt{5})/2$.

Formula lui Binet ne permite să calculăm F_n fără a cunoşte a priori valorile F_{n-1} şi F_{n-2}. Putem deduce inductiv formula

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Arbori binari stricti si echilibrati AVL

Teorema AVL

Margine superioara si margine inferioara pentru inaltimea unui arbore binar echilibrat AVL

Arborii Fibonacci

Verificăm pentru $n = 1$ şi $n = 2$.

$$F_1 = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^1 - \left(\frac{1-\sqrt{5}}{2}\right)^1\right] = \frac{2\sqrt{5}}{2\sqrt{5}} = 1$$

$$F_2 = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^2 - \left(\frac{1-\sqrt{5}}{2}\right)^2\right] = \frac{1}{\sqrt{5}}\frac{1+2\sqrt{5}+5-1-\sqrt{5}-5}{4} = \frac{4\sqrt{5}}{4\sqrt{5}} = 1$$

Presupunem acum că formula este adevărată pentru $n = k-2$ şi $n = k-1$:

$$F_{k-2} = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2} - \left(\frac{1-\sqrt{5}}{2}\right)^{k-2}\right] \text{ şi } F_{k-1} = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{k-1}\right]$$

Demonstrăm că $F_k = F_{k-2} + F_{k-1}$.

Arbori binari stricti si echilibrati AVL

Teorema AVL

Arborii Fibonacci

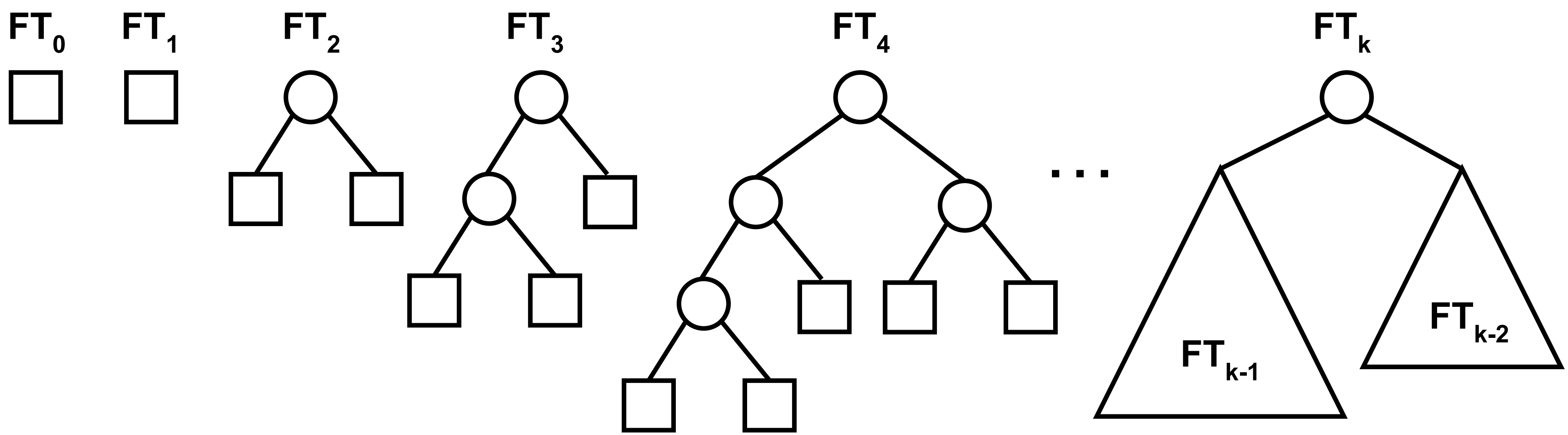
$$
\begin{aligned}
F_{k-2}+F_{k-1} &= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2}-\left(\frac{1-\sqrt{5}}{2}\right)^{k-2}+\left(\frac{1+\sqrt{5}}{2}\right)^{k-1}-\left(\frac{1-\sqrt{5}}{2}\right)^{k-1}\right] \\
&= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2}\left(1+\frac{1+\sqrt{5}}{2}\right)-\left(\frac{1-\sqrt{5}}{2}\right)^{k-2}\left(1+\frac{1-\sqrt{5}}{2}\right)\right] \\
&= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2}\left(\frac{3+\sqrt{5}}{2}\right)-\left(\frac{1-\sqrt{5}}{2}\right)^{k-2}\left(\frac{3-\sqrt{5}}{2}\right)\right] \\
&= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2}\left(\frac{6+2\sqrt{5}}{4}\right)-\left(\frac{1-\sqrt{5}}{2}\right)^{k-2}\left(\frac{6-2\sqrt{5}}{4}\right)\right] \\
&= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2}\left(\frac{1+2\sqrt{5}+5}{4}\right)-\left(\frac{1-\sqrt{5}}{2}\right)^{k-2}\left(\frac{1-2\sqrt{5}+5}{4}\right)\right] \\
&= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k-2}\left(\frac{1+\sqrt{5}}{2}\right)^{2}-\left(\frac{1-\sqrt{5}}{2}\right)^{k-2}\left(\frac{1-\sqrt{5}}{2}\right)^{2}\right] \\
&= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k}-\left(\frac{1-\sqrt{5}}{2}\right)^{k}\right] = F_k
\end{aligned}
$$

Arbori binari stricti si echilibrati AVL

Teorema AVL

Arborii Fibonacci

Construim prin recurenţă familia de arbori binari $(FT_k)_{k\geq 0}$, FT_k = Arbore Fibonacci (Fib Tree) de ordin k.



F_1	F_2	F_3	F_4	F_5		F_{k+1}
1	1	2	3	5	. . .	$F_k + F_{k-1}$

Arbori binari stricti si echilibrati AVL

Teorema AVL

Arborii Fibonacci

Lema 1: Pentru orice $k \geq 0$ arborele FT_k este a.b.s.
Lema 2: Pentru orice $k \geq 1$ arborele FT_k are caracteristicile:

(a) $h(FT_k) = k - 1$.

(b) $N_E(FT_k) = F_{k+1}$.

(c) $N_I(FT_k) = F_{k+1} - 1$.

Demonstrație: Este suficient să demonstrăm (a) şi (b), (c) este consecinţă a lui (b) prin Prop 1.
Inducţie după k, $k \geq 1$.
$k = 1$: FT_1 este ... cu $h(FT_1) = 0$ şi $N_E(FT_1) = 1 = F_2$.
Pp. (a) şi (b) adev. pentru FT_m, orice $m < k$.
Fie k oarecare, fixat, $k \geq 3$. Avem:

(a) $h(FT_k) = h(FT_{k-1}) + 1 = (k - 2) + 1 = k - 1$.

(b) $N_E(FT_k) = N_E(FT_{k-1}) + N_E(FT_{k-2}) = F_k + F_{k-1} = F_{k+1}$.

Arbori binari stricti si echilibrati AVL

Teorema AVL

Arborii Fibonacci

Lema 3: Pentru orice $k \geq 0$ arborele FT_k este echilibrat AVL.

Demonstrație: Pt. $k = 0, 1, 2$ direct. Pt. $k \geq 3$, (inductie), de dem. in nodul radacina se fol. (a) din Lema 2.

Lema 4: În familia arborilor binari stricţi şi echilibraţi AVL de înălţime data, h, arborii Fibonacci au număr minim de noduri interne.

Demonstraţie: Inducţie după h.
$h = 0$. Singurii a.b.Fib. ... au $N_I = 0$.
$h = 1$. T_1= a.b.s. de înălţime 1 şi nr minim de noduri interne, are 1 nod intern (rădăcina) şi 2 frunze, i.e. $T_1 = FT_2$.
Notăm cu T_h un a.b.s. şi echil. AVL de înălţime h care are nr. minim de noduri interne.

Arbori binari stricti si echilibrati AVL

Teorema AVL

Arborii Fibonacci

Lema 4: În familia arborilor binari stricţi şi echilibraţi AVL de înălţime data, h, arborii Fibonacci au număr minim de noduri interne.

Ipot. inducţie: pentru orice k, $k < h$ avem $T_k = FT_{k+1}$.

h oarecare, $h \geq 2$: fie T_h ca mai sus. Are nod rad. cu fii $left(T_h)$ şi $right(T_h)$. Putem pp. ca $h(left(T_h)) > h(right(T_h))$. Avem:

(i) $h(left(T_h)) = h - 1$ şi $N_I(left(T_h))$ minim, deci $left(T_h) = T_{h-1}$.

(ii) $h(right(T_h)) = h - 2$ şi $N_I(right(T_h))$ minim, deci $right(T_h) = T_{h-2}$.

Dar, cf. ipot. ind., $T_{h-1} = FT_h$ şi $T_{h-2} = FT_{h-1}$, deci, din (i) $left(T_h) = FT_h$ si din (ii) $right(T_h) = FT_{h-1}$, din care rezulta ca $T_h = FT_{h+1}$.

Observatie: Cf. Lemei 1 nr. minim de noduri interne pentru înălţime h dată va fi $N_I(FT_{h+1}) = F_{h+2} - 1$.

Arbori binari stricti si echilibrati AVL

Teorema AVL

Demonstratie ineg. 2

Fie T un a.b.s. şi echil AVL, cu $n = N_I(T)$ noduri interne şi înălţime $h = h(T)$. Cf. Obs. de după lema 4, avem

$$n \geq F_{h+2} - 1.$$

$$\frac{1}{\sqrt{5}}\phi^{h+2} - \frac{1}{\sqrt{5}}\bar{\phi}^{h+2} - 1 \leq n,$$

unde $\phi = \frac{1+\sqrt{5}}{2}$, $\bar{\phi} = \frac{1-\sqrt{5}}{2}$.

Arbori binari stricti si echilibrati AVL

Teorema AVL

Demonstratie ineg. 2

. Din $-1 \leq \bar{\phi} \leq 0$ rezultă

$$n \geq \frac{1}{\sqrt{5}}\phi^{h+2} - 2,$$

$$n + 2 \geq \frac{1}{\sqrt{5}}\phi^{h+2},$$

$$log_2(n+2) \geq (h+2)log_2\phi - \frac{1}{2}log_2 5.$$

Desfac, în fcţ. de h ... rezultă

$$h \leq \frac{1}{log_2\phi}log_2(n+2) + \frac{log_2 5}{2log_2\phi} - 2 = a\ log_2(n+2) + b,$$

si $a < 1.4404$, $b < -0.328$.

Perspective - curs 9

Arbori binari stricti cu ponderi

Algoritmul lui Huffman