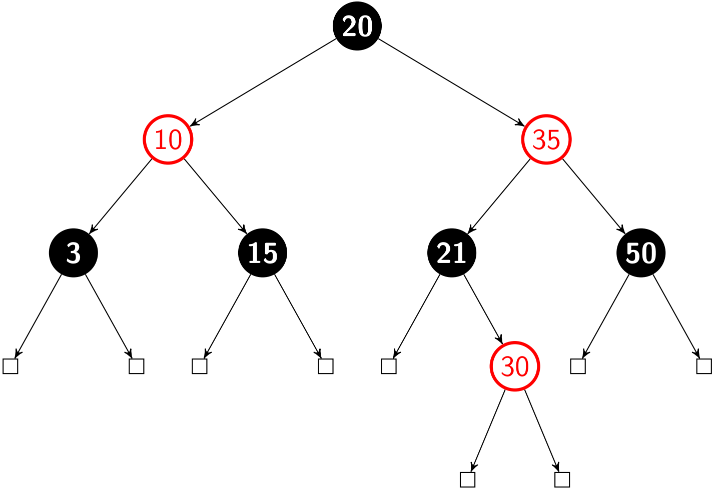
Arbori de căutare echilibrați.

SD 2017/2018

Arbori bicolori (*red-black trees*)

- *Symmetric binary B-tree*, Rudolf Bayer, 1972.
- Echilibrarea este menținută cu ajutorul unei colorări a nodurilor.
- Arborii roșu-negru sunt arbori binari de căutare care satisfac următoarele proprietăți:
  1. un nod este colorat cu roșu sau negru;
  2. rădăcina și nodurile frunză (*nil* – care fac parte din structură) sunt colorate cu negru;
  3. dacă un nod este roșu, atunci fiii săi sunt negri;
  4. drumurile de la un nod la nodurile de pe frontieră au același număr de noduri negre.

Arbori bicolori - exemplu



Arbori bicolori

Lemă:
Orice subarbore al unui arbore bicolor are cel puțin $2^{bh(v)} - 1$ *noduri interne, unde:*

- v *rădăcina subarborelui,*
- $bh(v)$ *numărul de noduri negre aflate pe un drum de la* v *la un nod de pe frontieră, excluzându-l pe* v*;*

Demonstrație.
La curs. □

Arbori bicolori

Teoremă:

Un arbore bicolor cu n noduri interne are înălțimea $h \leq 2\log_2(n+1)$.

Demonstrație.

Conform proprietății 3,

$n \geq 2^{h/2} - 1 \quad \Rightarrow \quad h/2 \leq \log_2(n+1) \quad \Rightarrow \quad h \leq 2\log_2(n+1)$. $\square$

Arbori bicolori: operații

Corolar:
Într-un arbore bicolor cu n noduri, operația de căutare are complexitatea timp $O(\log n)$.

Operația de inserare

- Se caută poziția de inserare și se inserează noua valoare ca în cazul arborilor binari de căutare obișnuiți.
- Se colorează noul nod cu roșu.
- Se restaurează proprietățile de arbore bicolor prin recolorare de noduri și aplicare de rotații simple.

Operația de inserare

- Proprietatea 1: satisfăcută.
- Proprietatea 2 – satisfăcută (ambii fii ai nodului inserat sunt *nil*). Dacă nodul inserat este rădăcina $\rightarrow$ recolorare în negru.
- Proprietatea 4 – satisfăcută (noul nod roșu înlocuiește o frunză).
- Poate să nu fie respectată proprietatea 3 - dacă părintele nodului este roșu.
  - Mută mai sus această situație prin recolorarea nodurilor până când poate fi reparată prin operații de rotație și recolorare.

Operația de inserare: restaurarea proprietății 3

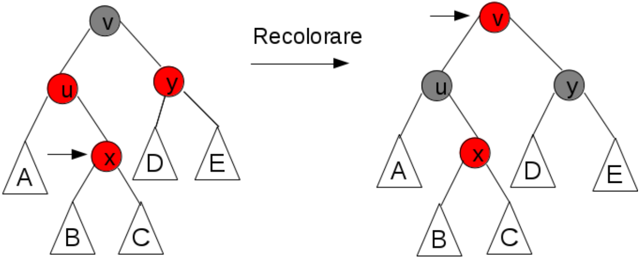
- **Caz 1:** "unchiul" nodului inserat este roșu →
  Se recolorează "părintele" și "unchiul" în negru și "bunicul" in roșu.

- **Caz 2:** "unchiul" nodului inserat este negru și nodul inserat este fiul drept al unui fiu stâng →
  Se aplică o rotație simplă la stânga între nodul curent și nodul părinte.

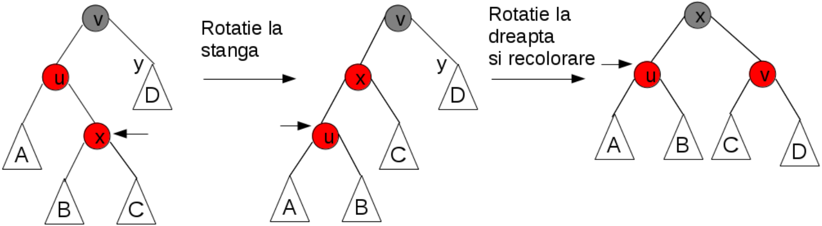
- **Caz 3:** "unchiul" nodului inserat este negru și nodul inserat este fiul stâng al unui fiu stâng →
  Se aplică o rotație simplă la dreapta între nodul "părinte" și nodul "bunic" + se recolorează nodurile "părinte" (în negru) și "bunic" (în roșu).

Obs.: Operații similare se aplică pentru cazul simetric.

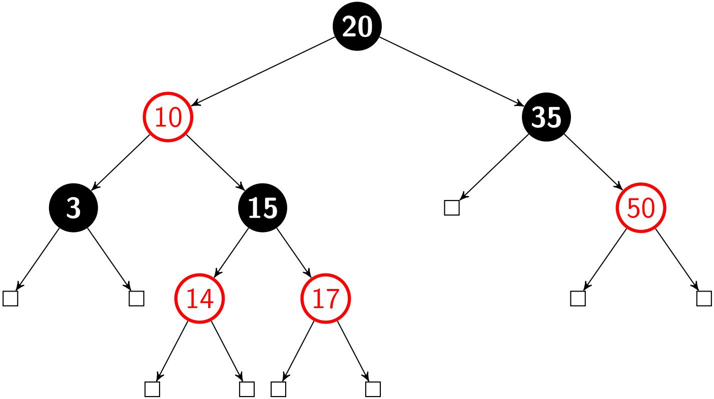
Operația de inserare - Caz 1



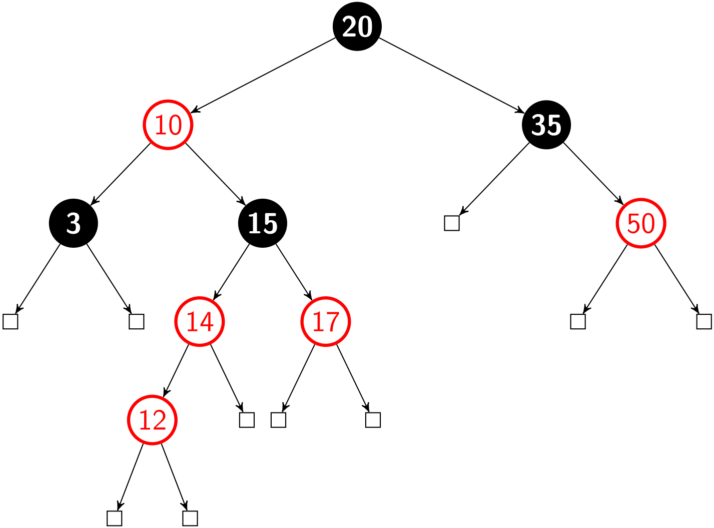
Operația de inserare - Caz 2 și 3



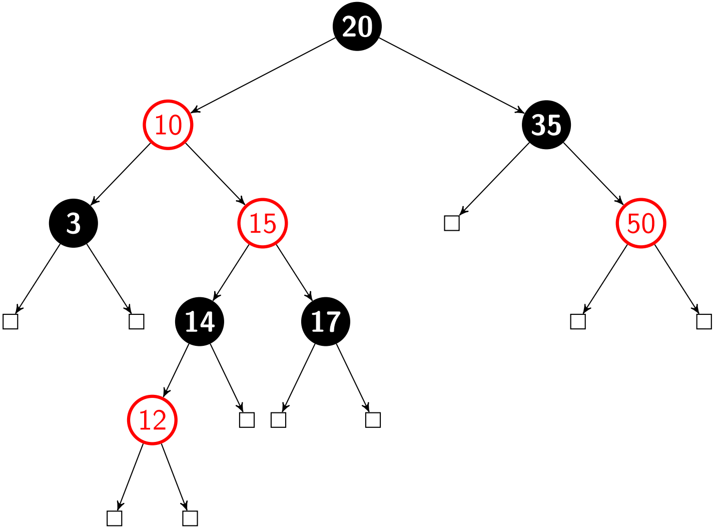
Inserare – exemplu: nodul 12



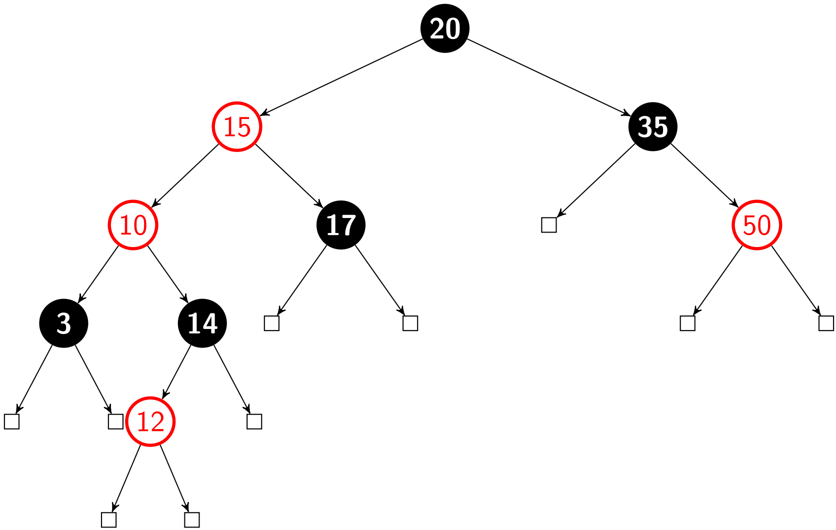
Inserare – CAZUL 1: recolorare



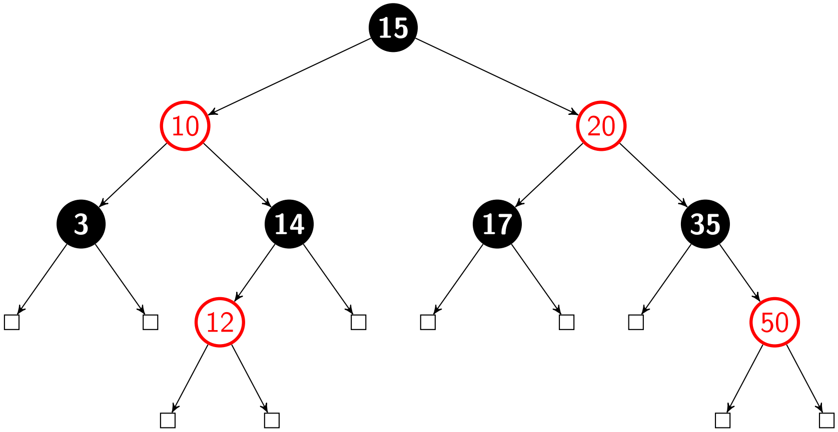
Inserare – CAZUL 2: rotație la stânga



Inserare – CAZUL 3: rotație la dreapta + recolorare



Inserare – Arborele roșu-negru valid



Operația de inserare: algoritm

Se consideră că fiecare nod a arborelui este o structură cu următoarele câmpuri:

- *cheie*: informația utilă a nodului;
- *culoare*: roșu / negru;
- *pred*: adresa nodului părinte (null pentru rădacină);
- *stg*: adresa fiului stâng;
- *drp*: adresa fiului drept.

Operația de inserare: algoritm

Procedure *inserare*(t, x)
begin
    insArbBinCautare(t, x)
    $x \to culoare \leftarrow$ rosu
    while $(x! = t \; and \; x \to pred \to culoare == rosu)$ **do**
        if $(x \to pred == x \to pred \to pred \to stg)$ **then**
            $y \leftarrow x \to pred \to pred \to drp$
            if $(y \to culoare == rosu)$ **then**
                Caz 1
            else
                if $(x == x \to pred \to drp)$ **then**
                    Caz 2
                Caz 3
        else
            similar cu ramura "then", doar că interschimbăm stg cu drp
    $t \to culoare \leftarrow$ negru
end

Operația de inserare: Caz 1

$x \rightarrow pred \rightarrow culoare \leftarrow$ negru
$y \rightarrow culoare \leftarrow$ negru
$x \rightarrow pred \rightarrow pred \rightarrow culoare \leftarrow$ roșu
$x \leftarrow x \rightarrow pred \rightarrow pred$

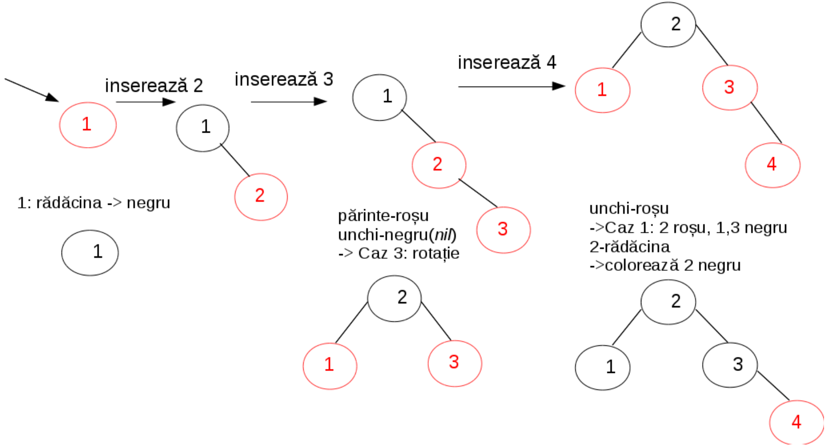
Operația de inserare: Caz 2

$x \leftarrow x \rightarrow pred$
rotatie-stânga(t, x)

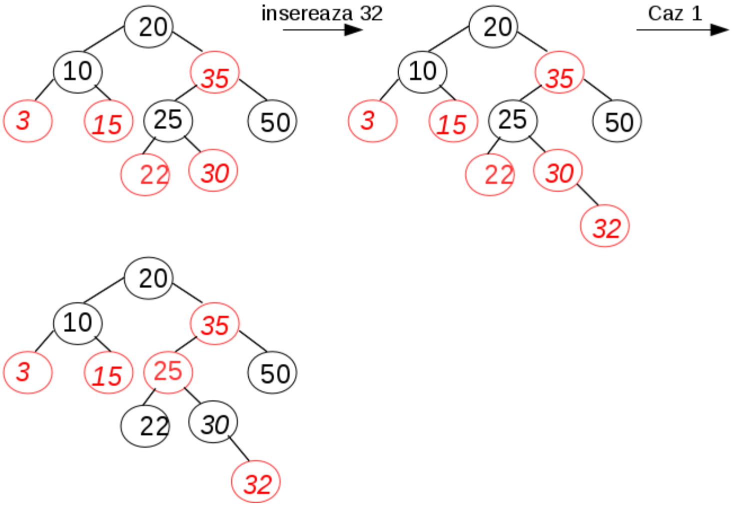
Operația de inserare: Caz 3

$x \rightarrow \textit{pred} \rightarrow \textit{culoare} \leftarrow \text{negru}$
$x \rightarrow \textit{pred} \rightarrow \textit{pred} \rightarrow \textit{culoare} \leftarrow \text{roșu}$
rotatie-dreapta$(t, x \rightarrow \textit{pred} \rightarrow \textit{pred})$

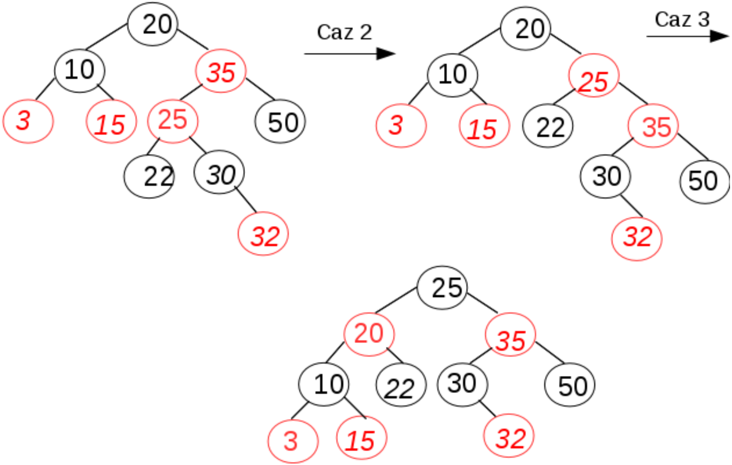
Operația de inserare - exemplul 2



Operația de inserare - exemplul 3



Operația de inserare - exemplul 3



Operația de ștergere

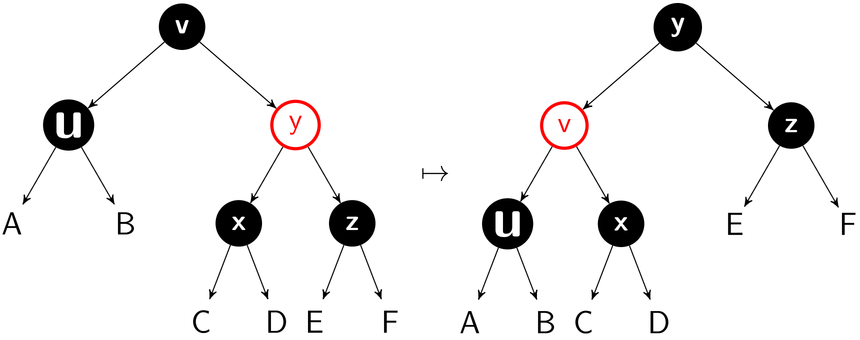
- Similară cu operația de ștergere de la arborii binari de căutare obișnuiți.
- Se va ține cont că nodurile “null” fac parte din structură.
- În urma ștergerii este posibil ca proprietatea 4 să nu mai fie respectată.
- Se restaurează proprietățile de arbore bicolor prin recolorare de noduri și aplicare de rotații simple.

Operația de ștergere: restaurarea proprietății 4

- **Caz 1:** Se transformă într-unul din cazurile 2), 3), 4) printr-o rotație.
- **Caz 2:** Nodul pentru care nu este satisfăcută proprietatea este deplasat spre rădăcină cu un nivel prin recolorarea unui nod.
- **Caz 3:** Se transformă în cazul 4) printr-o interschimbare de culori și o rotație.
- **Caz 4:** În acest caz se restabilește proprietatea de arbore bicolor pentru întreg arborele.

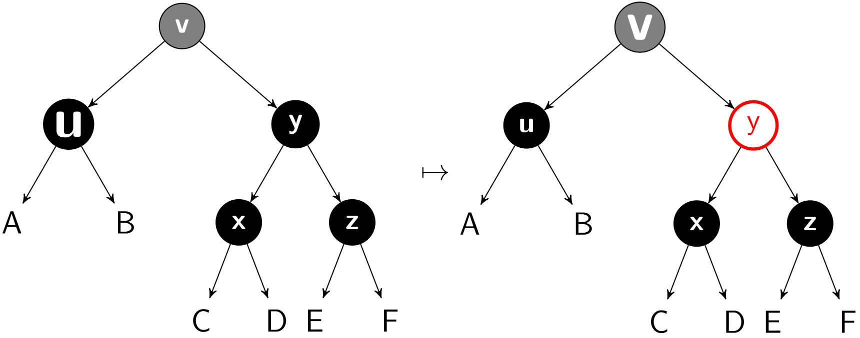
Ștergere – CAZUL 1

Caz 1: Se transformă într-unul din cazurile 2), 3), 4) printr-o rotație.



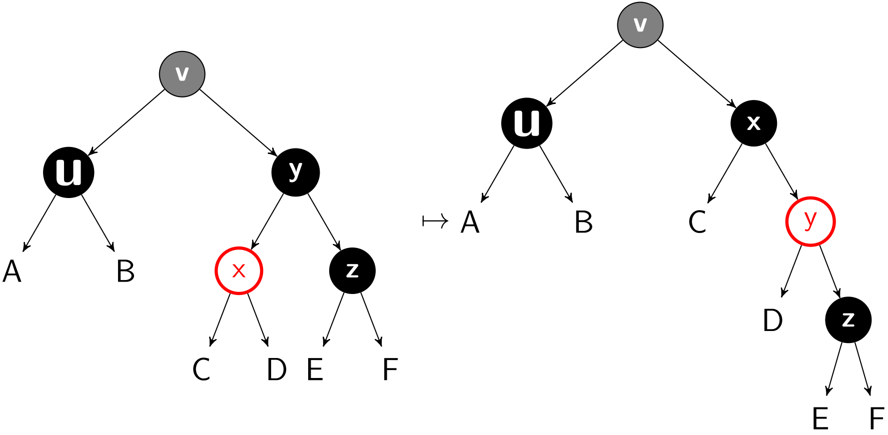
Ștergere – CAZUL 2

Caz 2: Nodul pentru care nu este satisfăcută proprietatea este deplasat spre rădăcină cu un nivel prin recolorarea unui nod.



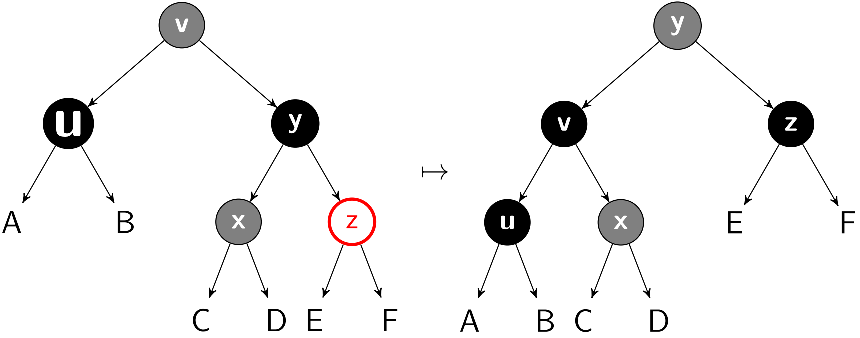
Ștergere – CAZUL 3

Caz 3: Se transformă în cazul 4) printr-o interschimbare de culori și o rotație.



Ștergere – CAZUL 4

Caz 4: În acest caz se restabilește proprietatea de arbore bicolor pentru întreg arborele.



Arbori bicolori

- Complexitatea algoritmilor de inserare / ștergere: $O(\log n)$.

Corolar:
Clasa arborilor bicolori este $O(\log n)$*–stabilă.*

Arbori bicolori

- Complexitatea algoritmilor de inserare / ștergere: $O(\log n)$.

Corolar:
Clasa arborilor bicolori este $O(\log n)$*–stabilă.*

- Utilizări:
  - System symbol tables.
  - Kernel Linux (Completely Fair Scheduler).