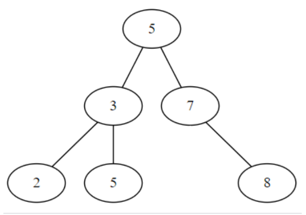
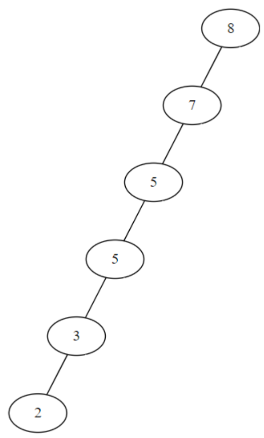
ARBORI DE CĂUTARE ECHILIBRAŢI
(BALANCED SEARCH TREES)

Analiza arborilor binari de căutare

- operaţiile specifice se execută în timp dependent de înălţimea arborelui (complexitate timp $O(h)$).
- în cel mai rău caz pentru n elemente înălţimea este $n-1$ (arbore degenerat) $\Rightarrow \theta(n)$ înălţime (complexitate în caz defavorabil pentru operaţii).
- cazul ideal: arbore echilibrat a cărui înălţime să fie $O(log_2 n)$.



(a) ABC echilibrat.



(b) ABC degenerat (lanţ).

  - ideea: la fiecare nod să păstrăm *echilibrarea.*
  - când un nod îşi pierde *echilibrul* $\Rightarrow$ **reechilibrare** (prin rotaţii specifice).

- sunt mai multe moduri de definire a echilibrării $\Rightarrow$ variante de arbori de căutare echilibraţi.
  - **arbori AVL**, arbori splay, arbori roşu-negru, B-arbori, etc.
  - caracteristică comună: înălţimea arborelui este $O(log_2 n)$.

ARBORI AVL

Definiţie 1 *Un **Arbore AVL** (Adelson Velski Landis) este un ABC care satisface următoarea* proprietate *(**invariant AVL**):*

- *dacă x este un nod al AVL, atunci:*
  - *înălţimea subarborelui stâng al lui x diferă de înălţimea subarborelui drept al lui x cu 0, 1 sau -1 (0, 1 sau -1 se numeşte **factor de echilibrare**).*

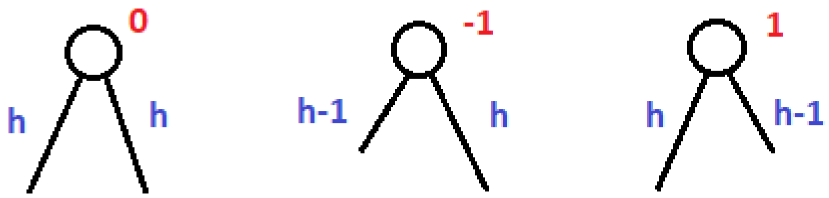
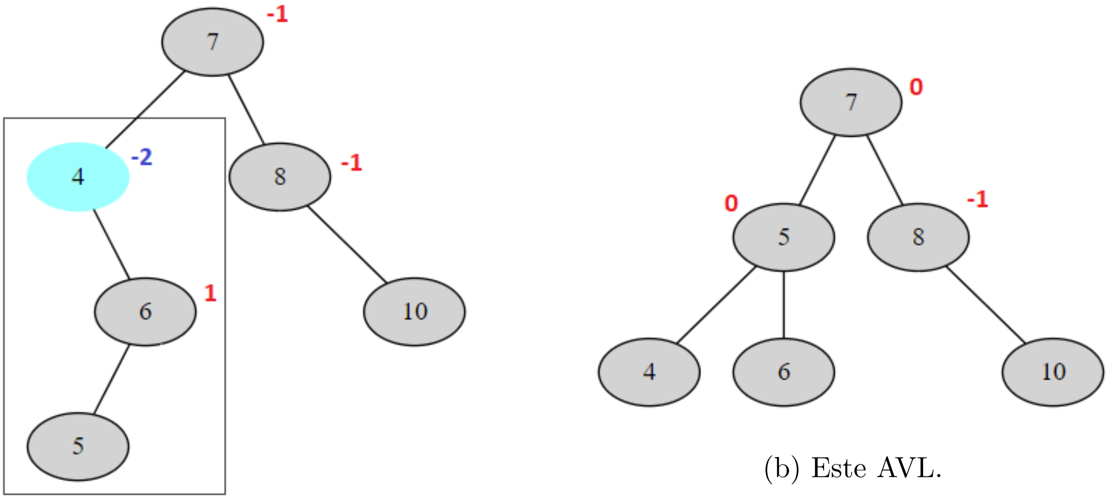


Figura 2: Factori de echilibrare posibili la orice nod al unui AVL.

- *în AVL cheile (elementele) memorate în noduri sunt distincte.*

Exemplu Presupunem că în container avem cheile 4 5 6 7 8 10 şi relaţia $\mathcal{R} = \leq$. În Figura **??** sunt indicaţi 2 ABC care conţin aceste chei. Cel din stânga nu este AVL, pe când cel din dreapta este AVL.

- la aborele din Figura **??**(a) va fi necesară o rotaţie în subarborele marcat, pentru a-l echilibra.



(a) Nu este AVL.

(b) Este AVL.

Figura 3: Arbori care conţin aceeaşi mulţime de chei: cel din stânga nu e echilibrat, cel din dreapta e echilibrat.

Proprietate. Înălţimea unui arbore AVL cu n noduri este $\theta(log_2 n)$.
În cazul în care arborele are număr maxim de noduri (n), acesta este **plin** (orice nod interior are factorul de echilibrare 0). $\Rightarrow h = \theta(log_2 n)$

- Notăm cu $N(h)$ numărul minim de noduri ale unui arbore AVL de înălţime h.
  - toate nodurile interioare au factor de echilibrare -1 sau 1.
  - înălţimea unui arbore (a cărui rădăcină este p) se poate defini recursiv ca fiind 1+ maximul dintre înăţimea subarborelui stâng şi înăţimea subarborelui drept $h(p) = 1 + max(h([p].st), h([p].dr))$.

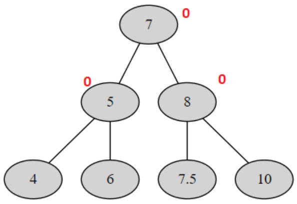
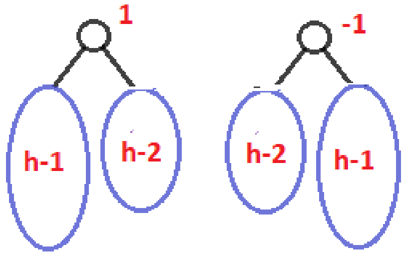


Figura 4: AVL cu număr maxim de noduri n.

(a) Înălţime $h = 0$.

(b) Înălţime $h = 1$



(c) Înălţime h

Figura 5: Număr minim de noduri în AVL

$$N(h) = \begin{cases} 1 & h = 0 \\ 2 & h = 1 \\ N(h-1) + N(h-2) + 1 & altfel \end{cases} \tag{1}$$

– se poate arăta că $N(h) \approx \phi^h$, unde $\phi = \frac{1+\sqrt{(5)}}{2}$ este numărul de aur (*golden ratio*), $\phi = 1.618....$

$$\Rightarrow h \approx log_{\phi} n \in \theta(log_2 n)$$

Rotaţii şi situaţii de reechilibrare în AVL

- 6 **situaţii de reechilibrare** (Knuth);
  - în cazul **adăugării** unui element
  - în cazul **ştergerii** unui element
- 4 tipuri de **rotaţii** pentru reechilibrare:
  1. o singură rotaţie spre stânga (**SRS**);
  2. dublă rotaţie spre stânga (**DRS**);
  3. o singură rotaţie spre dreapta (**SRD**);
  4. dublă rotaţie spre dreapta (**DRD**).

Situaţii de reechilibrare la adăugare

Caz I - rotaţii spre stânga

Caz Ia) - e necesară o SRS (Figura 6)

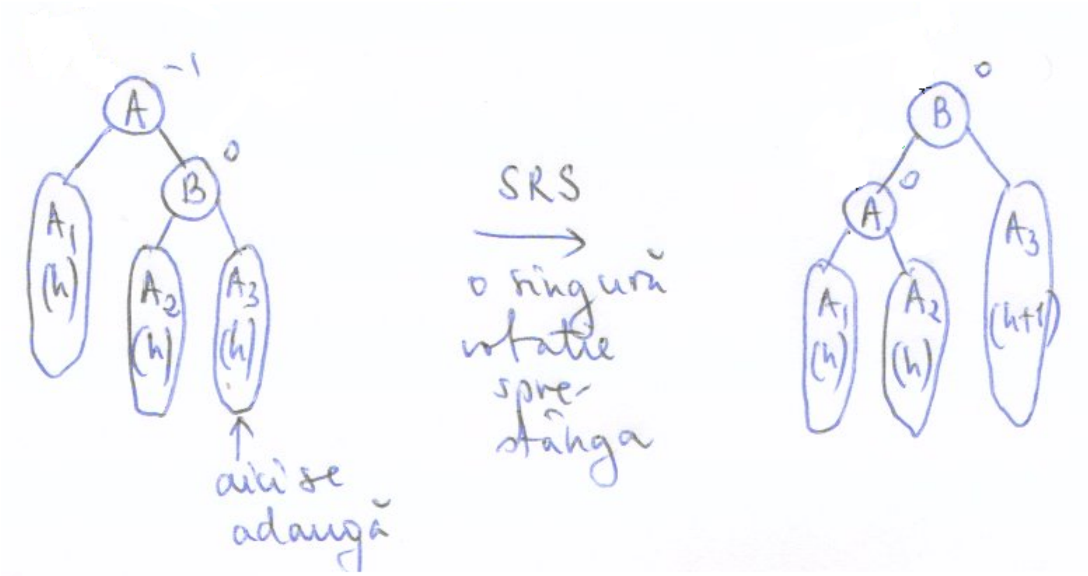


Figura 6: Caz Ia) la adăugare - e necesară o SRS pentru reechilibrare.

Exemplu caz Ia)

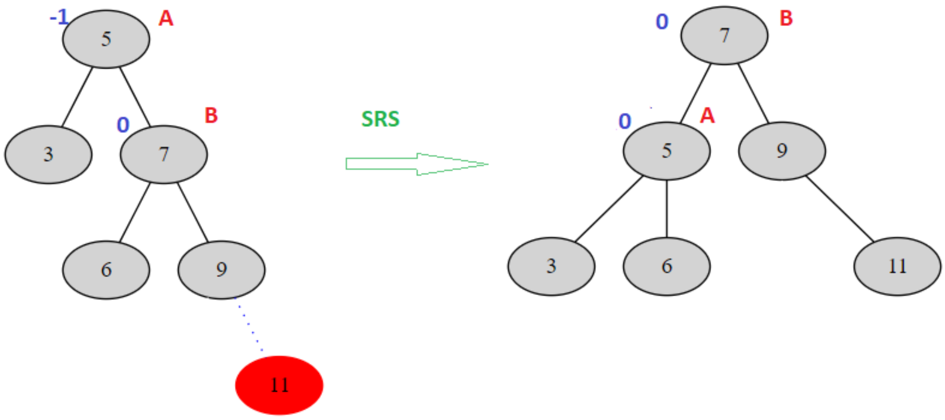


Figura 7: Exemplu caz Ia) la adăugare - e necesară o SRS pentru reechilibrare.

!!! Atenţie !!!

- **la inserarea unui element, rotaţiile se aplică în subarborele de înălţime minimă a cărui rădăcină şi-a pierdut echilibrul (de jos în sus - de la frunze spre rădăcină)**

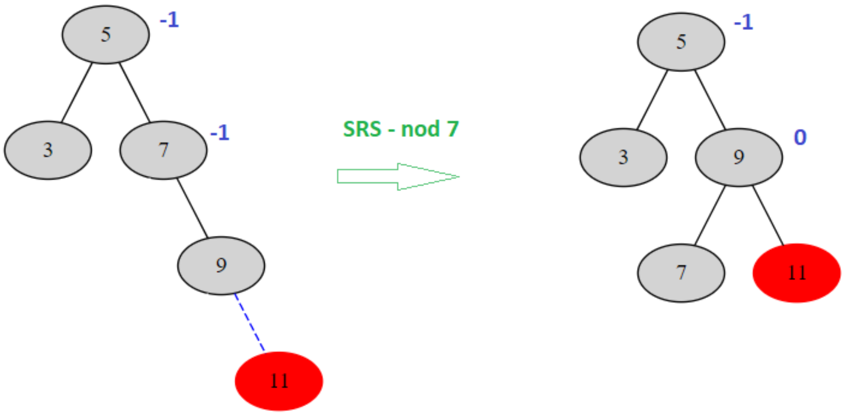


Figura 8: Echilibrul se strică la nodul 7. În subarborele de rădăcină 7 se aplică SRS pentru reechilibrare.

Implementare SRS

- pentru implementarea operaţiilor, pp. în cele ce urmează reprezentare înlănţuită folosind alocare dinamică.
- pp. că fiecare nod memorează:
  - informaţia utilă (e);
  - adresa celor doi subarbori (stâng st şi drept dr);
  - înălţimea nodului în arbore (h).

<u>Nod</u>
  e: TElement *//informaţia utilă nodului*
  st: $\uparrow$ Nod *//adresa la care e memorat descendentul stâng*
  dr: $\uparrow$ Nod *//adresa la care e memorat descendentul drept*
  h: $\uparrow$ Intreg *//înălţimea nodului*

```
  Functia h(p)
    {complexitate timp: θ(1)}
pre:    p :↑ Nod
post:    se returnează înălţimea lui p
    {dacă e subarbore vid}
    Daca p =NIL atunci
      h ← -1
    altfel
      h ← [p].h
    SfDaca
  SfFunctia

  Functia inaltime(p)
    {complexitate timp: θ(1)}
pre:    p :↑ Nod
post:    recalculează înălţimea lui p pe baza înălţimilor subarborilor lui p
    {dacă e subarbore vid}
    Daca p =NIL atunci
      inaltime ← −1
```

```
    altfel
      {se recalculează înălţimea lui p pe baza înălţimilor celor doi fii}
      inaltime ← max(h([p].st), h([p].dr))+1
    SfDaca
SfFunctia
```

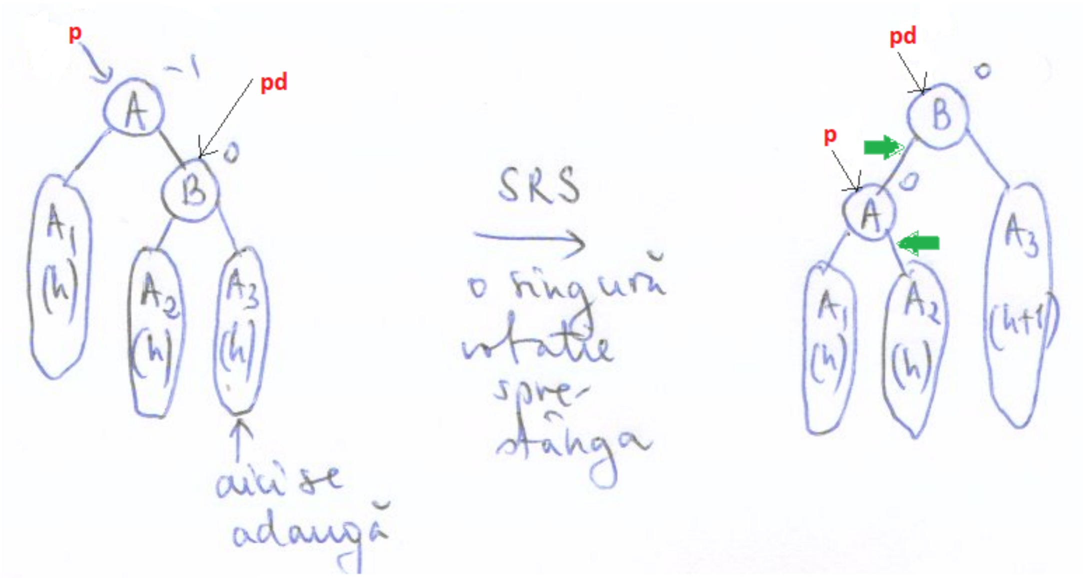


Figura 9: Situaţia de SRS pentru reechilibrare.

Functia SRS(p)

{complexitate timp: $\theta(1)$}

pre: p este adresa unui nod; $p :\uparrow Nod$ este rădăcina unui subarbore

post: se returnează rădăcina noului subarbore rezultat în urma unei SRS aplicate arborelui cu rădăcina p

{ $pd :\uparrow Nod$ e fiul drept }

$pd \leftarrow [p].dr$

{ se restabilesc legăturile între noduri conform SRS}

$[p].dr \leftarrow [pd].st$

$[pd].st \leftarrow p$

{se recalculează înălţimile conform SRS}

$[p].h \leftarrow$ inaltime(p)

$[pd].h \leftarrow$ inaltime(pd)

SRS $\leftarrow pd$

SfFunctia

Caz Ib) - **e necesară o DRS (Figura 10)**

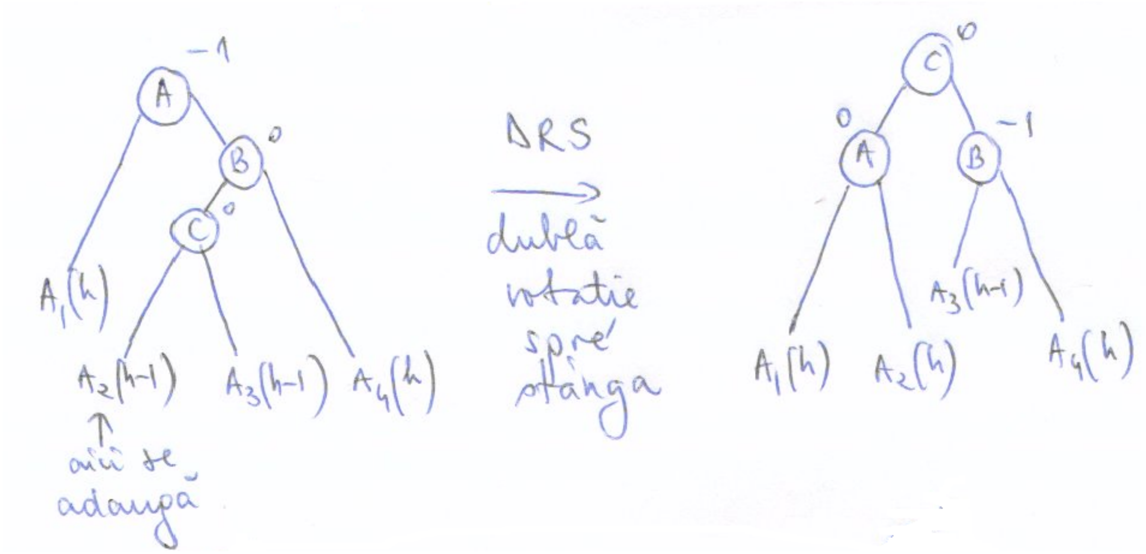


Figura 10: Caz Ib) la adăugare - e necesară o DRS pentru reechilibrare.

Exemplu caz Ib)

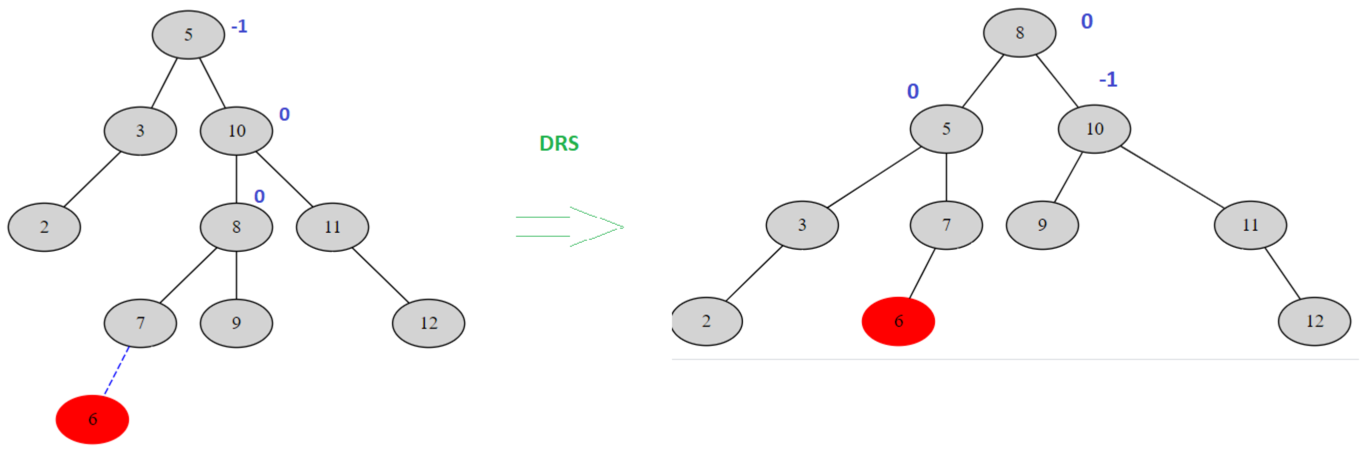


Figura 11: Exemplu caz Ib) la adăugare - e necesară o DRS pentru reechilibrare.

Caz Ic) **- e necesară o DRS (Figura 12)**

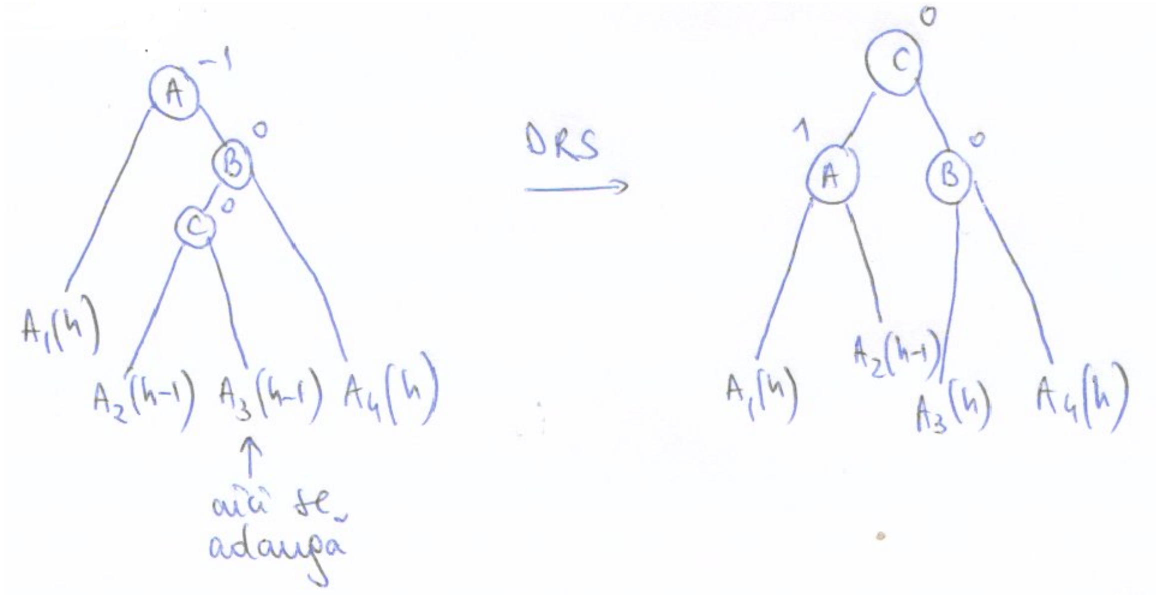


Figura 12: Caz Ic) la adăugare - e necesară o DRS pentru reechilibrare.

Exemplu caz Ic)

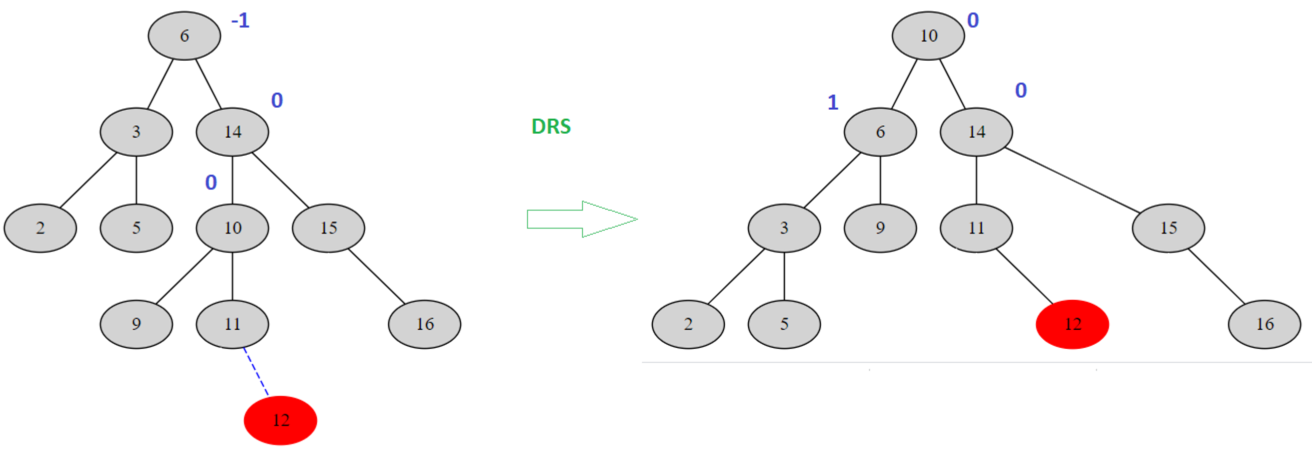


Figura 13: Exemplu caz Ic) la adăugare - e necesară o DRS pentru reechilibrare.

Caz II - rotaţii spre dreapta

- simetric cu cele trei situaţii de la cazul I
- **Caz IIa) - e necesară o SRD (caz simetric ca cel descris în Figura 6)**
- **Caz IIb) - e necesară o DRD (caz simetric ca cel descris în Figura 10)**
- **Caz IIc) - e necesară o DRD (caz simetric ca cel descris în Figura 12)**

Exemplu caz IIa)

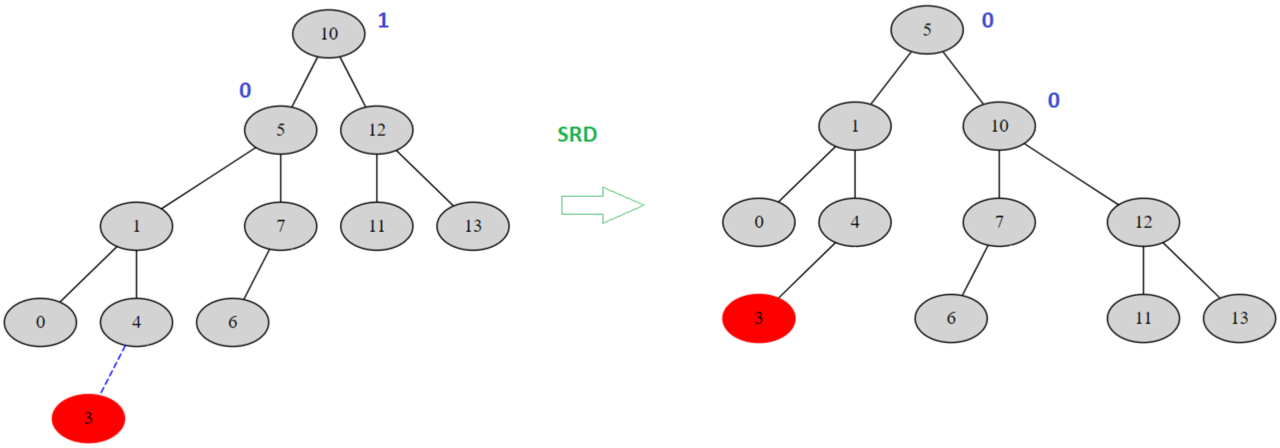


Figura 14: Exemplu caz IIa) la adăugare - e necesară o SRD pentru reechilibrare.

Exemplu caz IIc)

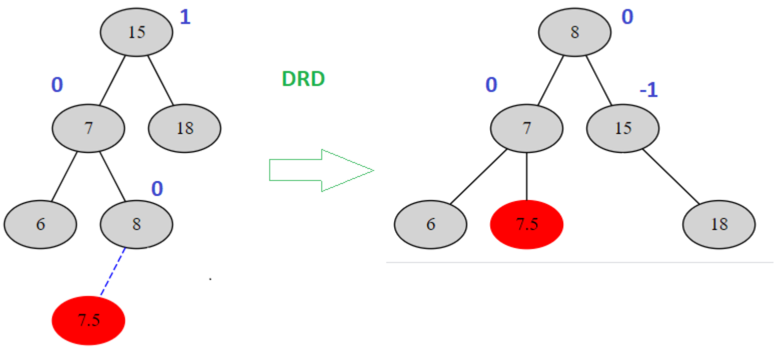


Figura 15: Exemplu caz IIc) la adăugare - e necesară o DRD pentru reechilibrare.

Operaţia de adăugare în arbore AVL

PP. reprezentare înlănţuită cu alocare dinamică, fiecare nod memorează şi înălţimea sa.

- un element identificat de o *cheie*

```
Functia creeazaNod(e) este  {θ(1)}
{creeaza un nod avand informatia utila 'e' si cei doi descendenti NIL}
        aloca(p)        {p:↑Nod}
        [p].e←e
        [p].st←NIL
        [p].dr←NIL
        [p].h←0
        creeazaNod←p
sf creeazaNod
```

```
Functia adauga_rec(p, e) este        { O(log₂ n) }
{se adauga informatia utila 'e' in subarborele de radacina 'p' si se returneaza noua
radacina a subarborelui }
        Daca p=NIL atunci
                p← creeazaNod(e)
          altfel
                daca e.c>[p].e.c atunci
                        [p].dr←adauga_rec([p].dr,e)
                        daca h([p].dr)-h([p].st)=2 atunci
                                daca e.c>[[p].dr].e.c atunci      {caz Ia}
                                        p←SRS (p)
                                   altfel                         {caz Ib, Ic}
                                        p←DRS (p)
                                sfDaca
                           altfel
                                [p].h←inaltime(p)
                        sfDaca
                    altfel
                        daca  e.c<[p].e.c atunci
                                @ simetric pe partea stanga – rotatii spre dreapta
                        altfel
                                @ cheie duplicat – nu e permisa in AVL
                        sfdaca
                sfDaca
        sfDaca
        adauga_rec←p
sf_adauga_rec

Subalgoritm adauga(ab, e) este { O(log₂ n) }
{se adauga informatia utila 'e' in arborele 'ab' si se returneaza arborele rezultat}
        ab.rad←adauga_rec(ab.rad, e)
sf_adauga
```

Observaţii

- alte reprezentări posibile pentru arborele AVL (ca şi pentru un ABC)
  - reprezentare înlănţuită cu reprezentare înlănţuiri pe tablou.
  - reprezentare secvenţială, folosind ca schemă de memorare un ansamblu.
- în locul înălţimii fiecărui nod, se poate memora *factorul de echilibrare* al acestuia

Situaţii de reechilibrare la ştergere

Caz I - rotaţii spre stânga

Caz Ia) şi Ib) - dacă prin ştergere din A1, înălţimea devine h-1 , e necesară o SRS (Figura 16)

- e posibil ca prin ştergere din A1, înălţimea să rămână $h \Rightarrow$ nu e necesară rotaţie

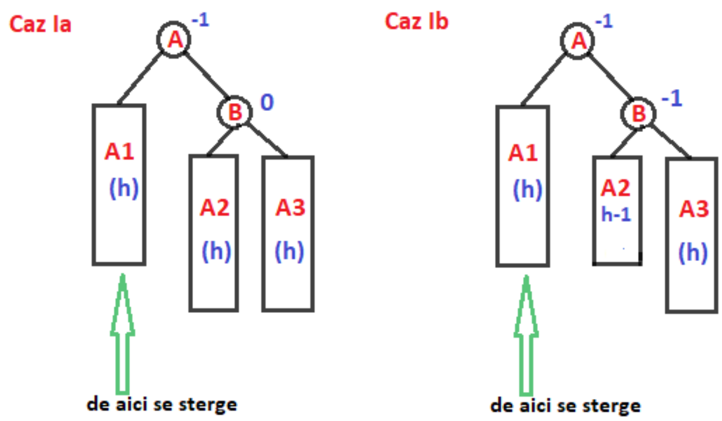


Figura 16: Caz Ia) şi Ib) la ştergere - e necesară o SRS pentru reechilibrare.

Exemple caz Ia) şi caz Ib) în care sunt necesare rotaţii

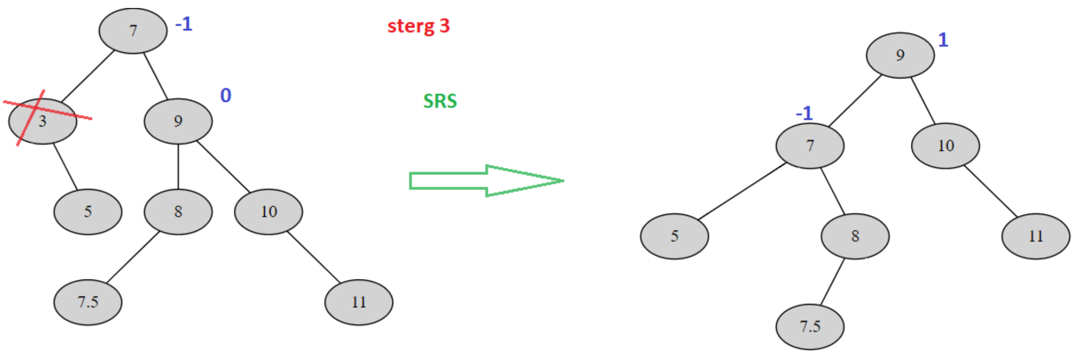


Figura 17: Exemplu caz Ia) la ştergere - e necesară o SRS pentru reechilibrare.

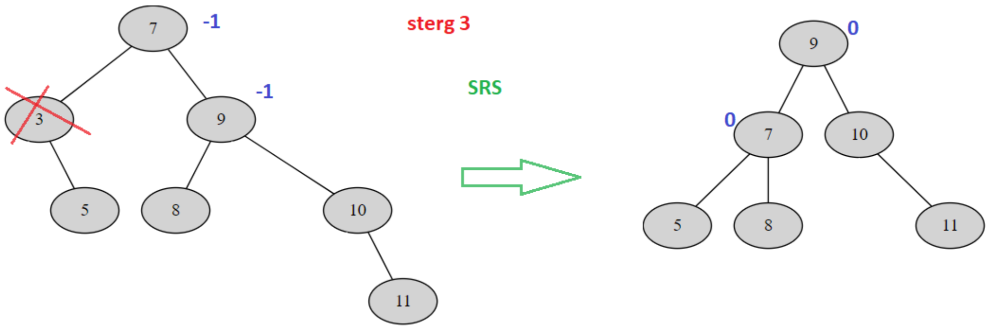


Figura 18: Exemplu caz Ib) la ştergere - e necesară o SRS pentru reechilibrare.

Exemplu caz Ia) în care nu e necesară rotaţie

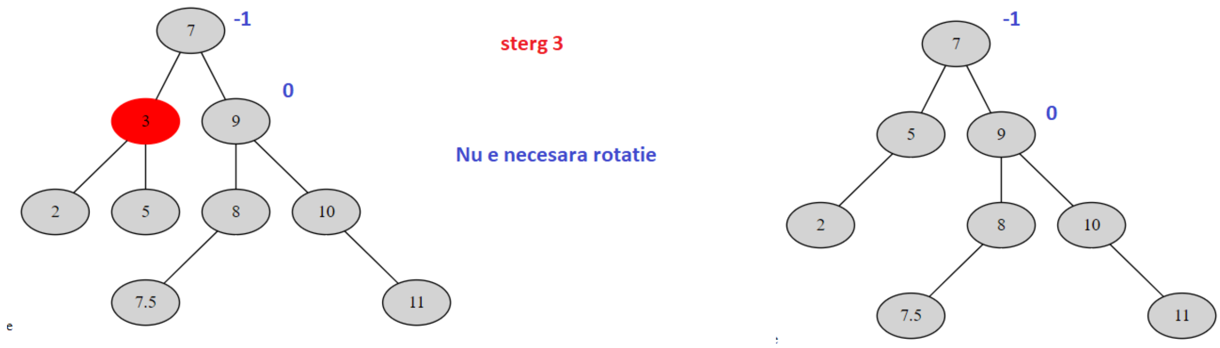


Figura 19: Exemplu caz Ia) la ştergere - nu e necesară rotaţie.

Caz Ic) - dacă prin ştergere din A1, înălţimea devine h-1, e necesară o DRS (Figura 20)

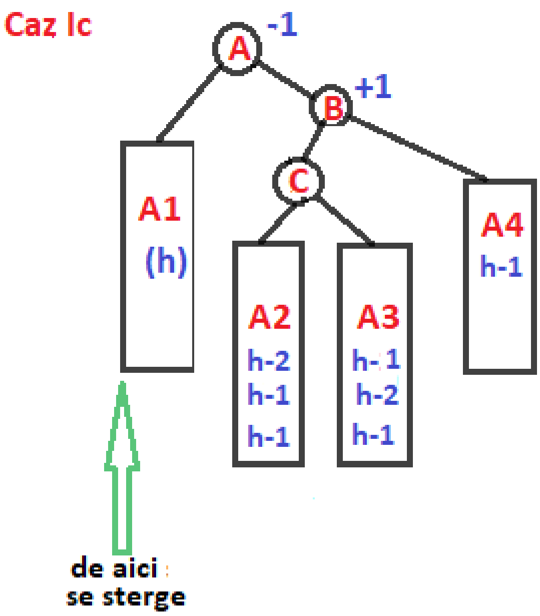


Figura 20: Caz Ic) la ştergere - e necesară o DRS pentru reechilibrare.

Exemplu caz Ic)

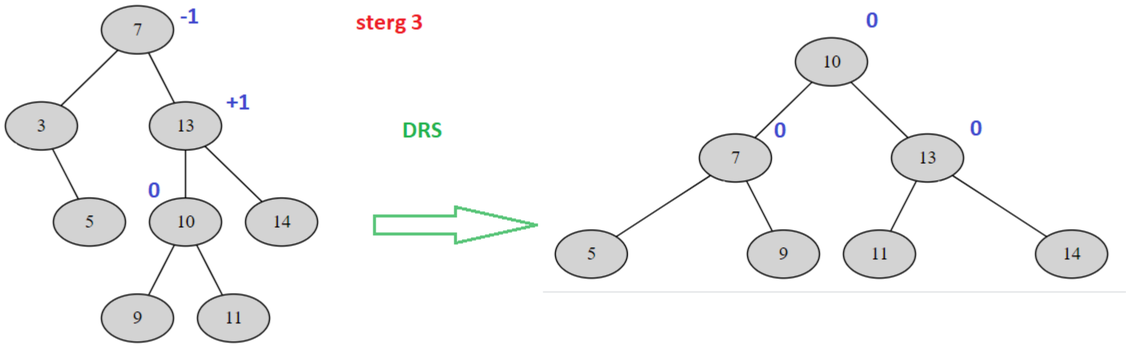


Figura 21: Exemplu caz Ic) la ştergere - e necesară o DRS pentru reechilibrare.

Caz II - rotaţii spre dreapta

- simetric cu Ia)

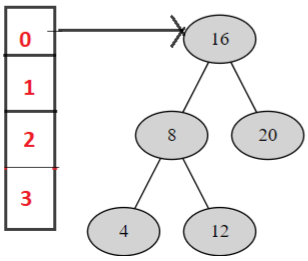
Observaţie

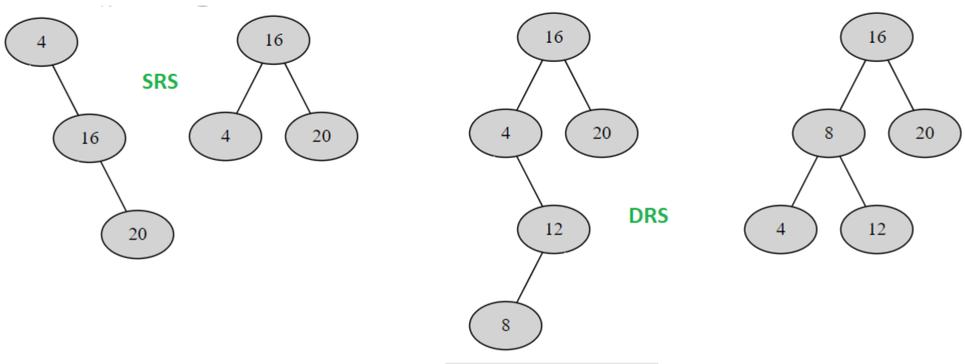
Tabelele de dispersie cu rezolvare coliziuni prin liste independente îşi pot memora listele folosind arbori AVL.

- se reduce complexitatea timp defavorabil la căutare de la $\theta(n)$ la $\theta(log_2 n)$.

Fie $m = 4$ şi funcţia de dispersie prin divizune.

c(**heie**)	4	5	16	9	20	7	12	8
$c \; mod \; m$	0	1	0	1	0	3	0	0





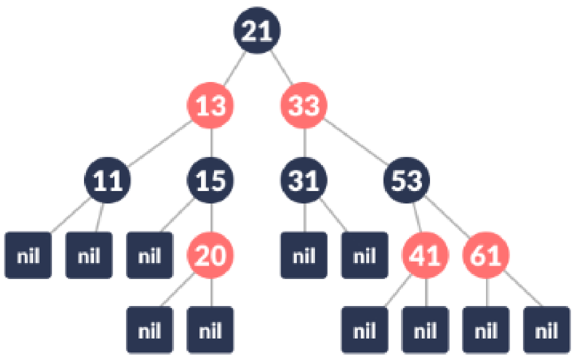
Probleme

1. Descrieţi în Pseudocod următoarele rotaţii: DRS, SRD, DRD.
2. Daţi exemple concrete în care apare necesitatea următoarelor tipuri de rotaţii la adăugare/ştergere: SRS, SRD, DRS, DRD.
3. Reprezentări alternative pentru AB, ABC, AVL folosind următoarele reprezentări:
   - reprezentare înlănţuită cu reprezentare înlănţuiri pe tablou.
   - reprezentare secvenţială, folosind ca schemă de memorare un ansamblu.

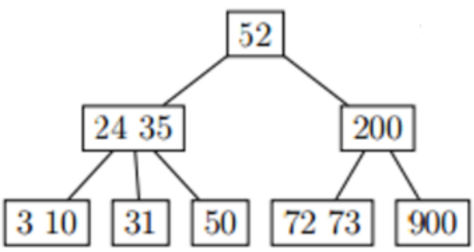
Alte tipuri de ABC echilibraţi

Înălţimea este $O(log_2 n)$

- arbori roşu-negru (*red-black trees*)
  - sunt ABC
  - nodurile au o culoare: *roşie* sau *neagră*
  - frunzele (**nil**) sunt negre
  - un nod roşu are cei doi fii de culoare neagră
  - pe orice drum de la rădăcină la o frunză, numărul nodurilor negre este acelaşi



- B-arbori



  - generalizare ABC
  - fiecare nod interior conţine mai multe chei
  - nodurile pot avea mai mult de 2 descendenţi
    - dacă un nod conţine 2 chei c_1 şi c_2, atunci are 3 descendenţi
  - folosiţi în *baze de date* şi *sisteme de fişiere*