ARBORI ECHILIBRAŢI
(BALANCED 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) $\Rightarrow$ $\theta(n)$ complexitate în caz defavorabil.
- cazul ideal: arbore echilibrat a cărui înălţime să fie $O(log_2 n)$.
  - 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 0.1 *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**).*

Proprietate. Înălţimea unui arbore AVL cu n noduri este $\theta(log_2 n)$.

- $N(h)$ - numarul minim de noduri ale unui arbore AVL de înălţime h.
- $N(0)$=1
- $N(h)$=$N(h-1)$+$N(h-2)$+1

- 6 situaţii de reechilibrare (Knuth);
- 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).

- pentru implementarea operaţiilor, pp. în cele ce urmează reprezentare înlănţuită folosind alocare dinamică.

- pp. că fiecare nod (Nod) memorează:
    - informaţia utilă (e);
    - adresa celor doi subarbori (stâng st şi drept dr);
    - înălţimea nodului în arbore (h).

Singură Rotaţie spre Stânga

```
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
```

SRS

```
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 :↑ Nod e fiul drept }
    pd ← [p].dr
    { se restabilesc legăturilerile între noduri conform SRS}
    [p].dr ← [pd].st
    [pd].st ← p
    {se recalculează înălţimile conform SRS}
    [p].h ←inaltime(p)
    [pd].h ←inaltime(pd)
    SRS ← pd
  SfFunctia
```

Observaţie

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

Probleme

1. Descrieţi în Pseudocod următoarele rotaţii: DRS, SRD, DRD.

2. Daţi exemple concrete în care apare necesitatea următoarelor tipuri de rotaţii: SRS, SRD, DRS, DRD.

3. Daţi exemple concrete în care apare necesitatea următoarelor tipuri de rotaţii: **SRS** (vezi la curs), SRD, DRS, DRD la adăugare.

4. Analizaţi ce se întâmplă la operaţia de ştergere dintr-un AVL: identificaţi situaţiile de reechilibrare (similar cu cele studiate la curs pentru adăugare).

5. Daţi exemple concrete în care apare necesitatea următoarelor tipuri de rotaţii: **SRS** (vezi la curs), SRD, DRS, DRD la ştergere.