Structură de arbore binar

Structura in memorie:

K	Data	$Pointer_{Left}$	$Pointer_{Right}$

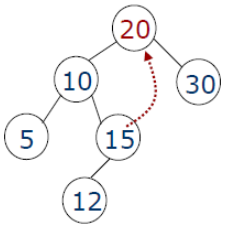
K = ID; Data = datele stocate; $Pointer_{Left/Right}$ = pointerii spre dreapta, respective stanga

Inserare:

- detectare poziţie înregistrare
- stocare înregistrare nouă într un nod liber
- legare nod la părinte

Eliminare:

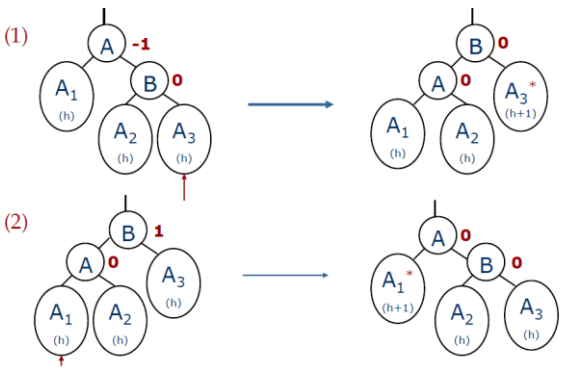
- căutare înregistrare
- 3 cazuri:
  - fără copii: pointer părinte= NULL
  - 1 copil: ataşează nod copil la părinte
  - 2 copii: înlocuieşte cu cel mai apropiat vecin
- adaugă nodul în lista de noduri libere



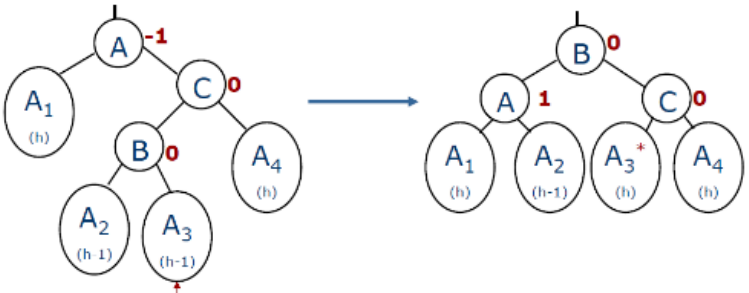
Arbori binari echilibraţi

- pentru fiecare nod diferenţa dintre înălţimile sub arborilor săi este 0, 1 sau 1 (înălţime arbore : dimensiunea celui mai lung drum de la rădăcină la frunze)

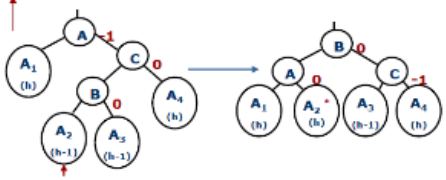
Cazuri de dezechilibrare:



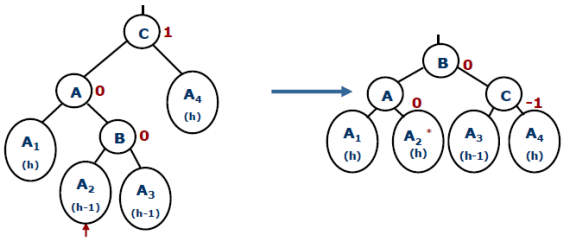
(3)



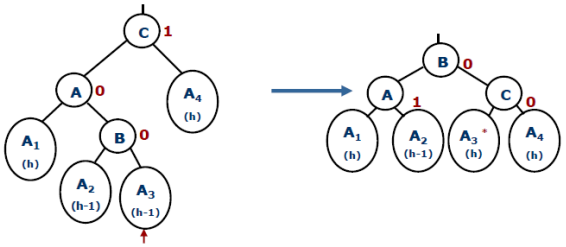
(4)



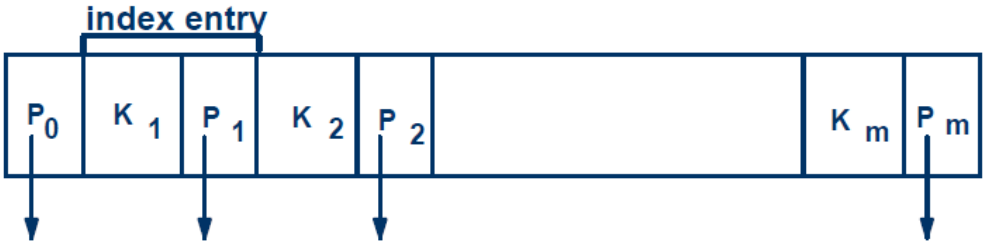
(5)



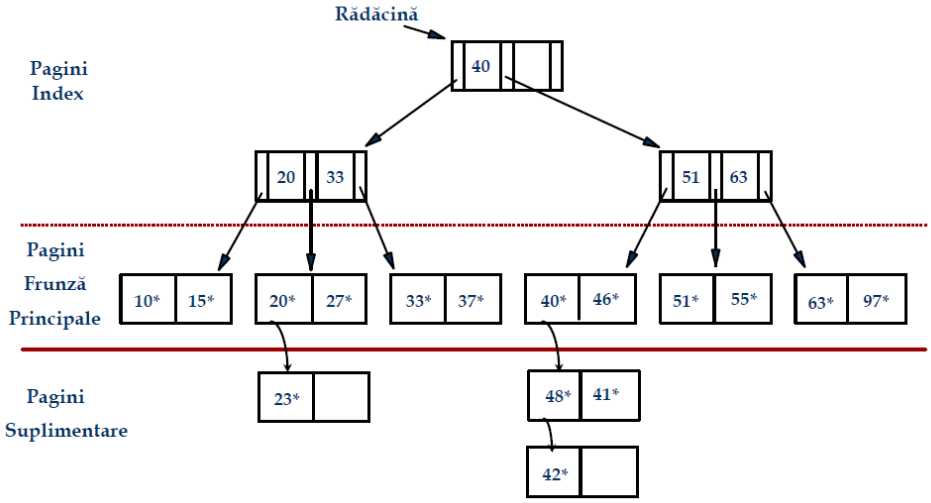
(6)



ISAM (Indexed Sequential Access Method)



Exemplu:

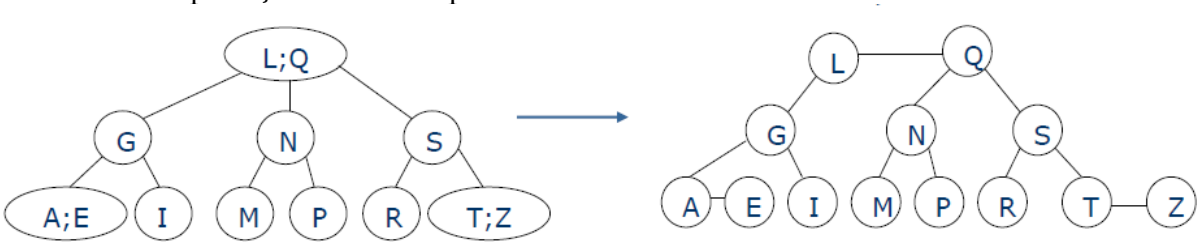


Organizare sub formă de Arbore-B (balanced/broad)

Proprietati:

Arbore-B de ordin *m*:

- Dacă nu e frunză, rădăcina va avea mereu cel puţin 2 subarbori
- Fiecare nod intern are cel puţin [*m*/2] sub-arbori (mai puţin rădăcina)
- Fiecare nod intern are cel mult *m* sub-arbori
- Toate frunzele sunt la acelaşi nivel
- Un nod cu p sub-arbori conţine
  - *p-1* chei ordonate(K1, K2, … Kp)
  - T1 conţine valori < K1
  - Ti conţine valori între Ki-1 si Ki
  - Tp conţine valori > Kp



Algoritmul procedurii de inserare:

1. Localizare nod pentru inserare
2. Inserare cheie
3. Daca nodul e plin (dimensiune depasita)
   a. Se creaza un nou nod in care se muta cheile mai mari decat valoarea cheii mediane
   b. Se insereaza cheia mediana in nodul parinte
   c. Pointerul din dreapta cheii va referii noul nod, iar cel stanga refera vechiul nod ce contine valorile mai mici
4. Daca nodul parinte e plin
   a. Daca nodul parinte e radacina atunci se creeaza o radacina noua
   b. Se repeta pasul 3 pentru nodul parinte

Algoritm de stergere:

1. Se cauta valoarea ce trebuie stearsa. Daca se afla intr-un nod intern se inlocuieste cu valoarea *vecina mai mare* (adica cu *cea mai din stanga valoare a celei mai din stanga frunze a subarborelui drept*)
2. Se repeta acest pas pana se ajunge in cazurile a. sau b.

a. Daca nodul contine valoarea de sters este radacina sau numarul valorilor ramase in nod este >= [m/2]:
   - se elimina valoarea
   - se re-aranjeaza *valorile si pointerii* din nod
   - se termina algoritmul

b. Daca numarul valorilor ramase in nod este < [m/2] unul dintre nodurile vecine contine > [m/2] valori → *redistribuire*
   - se ordoneaza valorile din ambele noduri + valoarea separator din parinte
   - se alege valoarea mediana si se adauga la nodul parinte, iar celelalte valori se insereaza in nodul stang, respective drept
   - algoritmul se termina

c. Daca suma valorilor din nodul din care s-a eliminat valoarea si a valorilor unuia din vecini este < m → *concatenare*
   - se insereaza valorile ambelor noduri + valoarea separator din nodul parinte intr-un singur nod
   - se repeta pasul 2. pentru nodul parinte (din care s-a eliminat valoarea separator)
   - daca nodul parinte este radacina si nu mai contine valori → nodul current devine radacina

Arbori B+ : combinatie intre Arbori-B si ISAM

- Căutarea pornește de la rădăcină, și va fi direcționată prin comparații către o frunză
- Într-un ArboreB+ toți pointerii către înregistrări din tabele se află doar la nivelul nodurilor frunză
- Un arbore B+ poate avea mai putine nivele (sau o capacitate mai mare pentru stocarea cheilor de cautare) decât arborele B correspondent