Laborator 6 – Arbori Binari de Cautare

Marti 16-18

- Un arbore este un graf neorientat, conex si fara cicluri – putem ajunge din orice nod in orice nod, dar nu se pot forma cicluri. Din acest motiv, se reprezinta sub forma arboresenta – un nod radacina, care are mai multi copii, fiecare din copiii sai putand avea alti copii la randul lor.
- Un arbore binar este un arbore in care fiecare nod are maxim doi fii – nodul stanga si nodul dreapta.
- Un arbore binar de cautare respecta urmatoarea proprietate – pentru orice nod, toate nodurile din stanga au valoarea tinuta in nod mai mica decat a nodului, iar toate nodurile din dreapta au valoarea mai mare decat a nodului. In acest fel, e foarte usor sa cautam daca o valoare exista sau nu in arbore.
- In general, arborii binari de cautare nu sunt echilibrati. Pentru a fi echilibrati, ar trebui ca pentru orice nod numarul de noduri ale subarborelui stang sa difere de numarul de noduri ale subarborelui drept cu maxim o unitate. Altfel spus, inaltimea arborelui trebuie sa fie O(log n). Arborii binari de cautare echilibrati sunt mai dificil de implementat si necesita operatii de rebalansare cand arborele nu mai echilibrat.
- Cautarea in ABC se face foarte usor -se incepe din radacina, daca valoarea cautata e mai mica mergem in stanga, daca e mai mare mergem in dreapta, pana gasim valoarea cautata sau ajungem intr-un nod frunza.
- Inserarea in ABC se face similar cu cautarea, se merge pana intr-un nod frunza (nu se tin duplicate in general in ABC) si se adauga la stanga sau la dreapta, dupa caz.
- Pentru stergere, avem mai multe cazuri:
  - pentru frunze, se sterg direct
  - pentru nodurile cu un singur fiu, se inlocuieste valoarea nodului cu valoarea copilului si se sterge fiul (copiii nodului vor deveni copiiii fiului)
  - pentru nodurile cu ambii fii, se cauta nodul precedent sau antecedent ca valoare (de exemplu, cel mai din dreapta copil al copilului stang), se inlocuieste valoarea nodului cu valoarea acestuia, apoi se sterge nodul respectiv.
- Pentru arbori, exista urmatoarele parcurgeri:
  - in-order: nod stanga, radacina, nod dreapta – obtinem arborele sortat
  - pre-order: radacina, nod stanga, nod dreapta (utila pentru copierea unui arbore)
  - post-order: nod stanga, nod dreapta, radacina (utila pentru eliberarea de memorie)

Aplicatii:

1. Avand urmatoarea structura:

```
typedef struct BSTree {
        int key;
        struct BSTree *left, * right;
} BSTree;
```

Sa se defineasca functii pentru initierea unui BSTree, search, insert, delete, free si height (inaltimea arborelui).

2. Sa se defineasca funcii care afiseaza arborele in inordine, preordine si postordine.

3. Din fisierul “abc.in” se citesc N linii, in felul urmator:

1 x = se adauga in ABC valoarea x
2 x = se sterge din ABC valoarea 3
3 x = se cauta in ABC valoarea x
0 = se afiseaza ABC in inordine

Sa se faca un program care citeste un astfel de fisier si afiseaza in “abc.out” rezultatele operatiei 4 la fiecare pas.