Arbori binari de căutare

Un arbore binar de cautare T este:

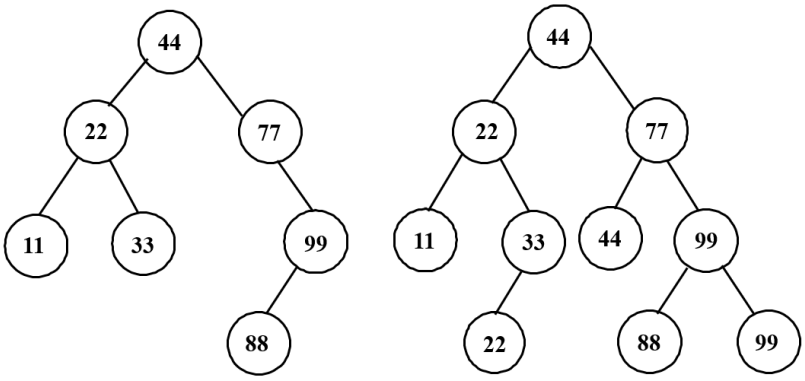
1. fie un arbore vid $T = \emptyset$
2. fie nevid şi atunci conține un nod numit rădăcină, cu info de un tip totul ordonat împreună cu doi subarbori binari de cautare disjuncţi (numiţi subarborele stâng, left, respectiv subarborele drept, right) si astfel incat:
   - *info[root(T)]>info[root(left[T])]*
   - *info[root(T)]<info[root(right[T])]*

```
struct Tree {
   int Data;
   Tree *Left;
   Tree *Right;
};

Tree *Root;
```

Numim arbore binar de căutare strict un arbore binar T cu proprietatea că în fiecare nod u al său avem relaţiile:

- info[u] > info[v], pentru orice v in left[u]
- info[u] < info[w], pentru orice w in right[u].



(a) Arbore binar de căutare strict.

(b) Arbore binar de căutare nestrict la dreapta. Cheile 22, 44 şi 99 sunt chei multiple.

Figura 1: Exemple arbori binari de cautare

1 Cautare cu inserare intr-un arbore binar de cautare

```
void SearchInsIterativ (int x,
          Tree *Root)
{
  Tree *p1, *p;
  int d;
  // iniţializarea pointerilor
  // pentru parcurgere
  p1 = NULL;
  p = Root;
  d = 1;
  while ((p != NULL) && (d != 0))
   if (x < p -> Data){
     p1 = p;
     p = p -> Left;
     d = -1;
   }
   else (if x > p -> Data) {
         p1 = p;
         p = p -> Right;
         d = 1;
   }
   else  //x = p -> info
         d = 0;
if (p != NULL)
// d = 0 şi am găsit x
//în arborele Root
else {
// p = NULL şi facem inserarea
 p = new Tree;
 p -> Data = x;
 p -> Left = NULL;
 p -> Right = NULL;
// legarea noului nod la tata
 if (p1 == NULL)
  //inserare într-un arbore vid
   Root = p1;
 else if (d < 0)
           p1 -> Left = p;
          else
           p1 -> right = p;
}
}
```

2 Aplicatii arbore binar de cautare

Exercițiul 1. Parcurgeti cheile unui arbore binar de cautare conform urmatoarelor strategii:

1. (0.5p) RSD (pre-ordine)
2. (0.5p) SRD (in-ordine)
3. (0.5p) SDR (post-ordine)

unde S - reprezinta arborele din stanga, R - nodul radacina, D - arborele din dreapta.

Exercițiul 2. (2.5p) Implementati un program pentru stergerea unui nod cu o cheie x data dintr-un arbore binar de cautare (pastrand proprietatea de arbore binar de cautare).

Exercitiul 3. (1p) Dat un arbore binar de cautare si doi intregi k1 si k2, sa se afiseze toate cheile x din arbore cu proprietatea k1 <= x <= k2.

Exercitiul 4. (2p) Să se ordoneze descrescător un şir de cuvinte date de la tastatură, folosind un arbore binar de căutare.

Exercitiul 5. (2p) Dat un arbore binar de cautare si doua noduri oarecare p si q, sa se gaseasca cel mai apropiat stramos comun (LCA - Lowest Common Ancestor).

3 Arbori binari de cautare echilibrati AVL

Se numeşte arbore binar de căutare echilibrat AVL (Adelson-Velskii-Landis) un arbore care în fiecare nod are proprietatea că înălţimile subarborilor stâng şi drept diferă cu cel mult 1.

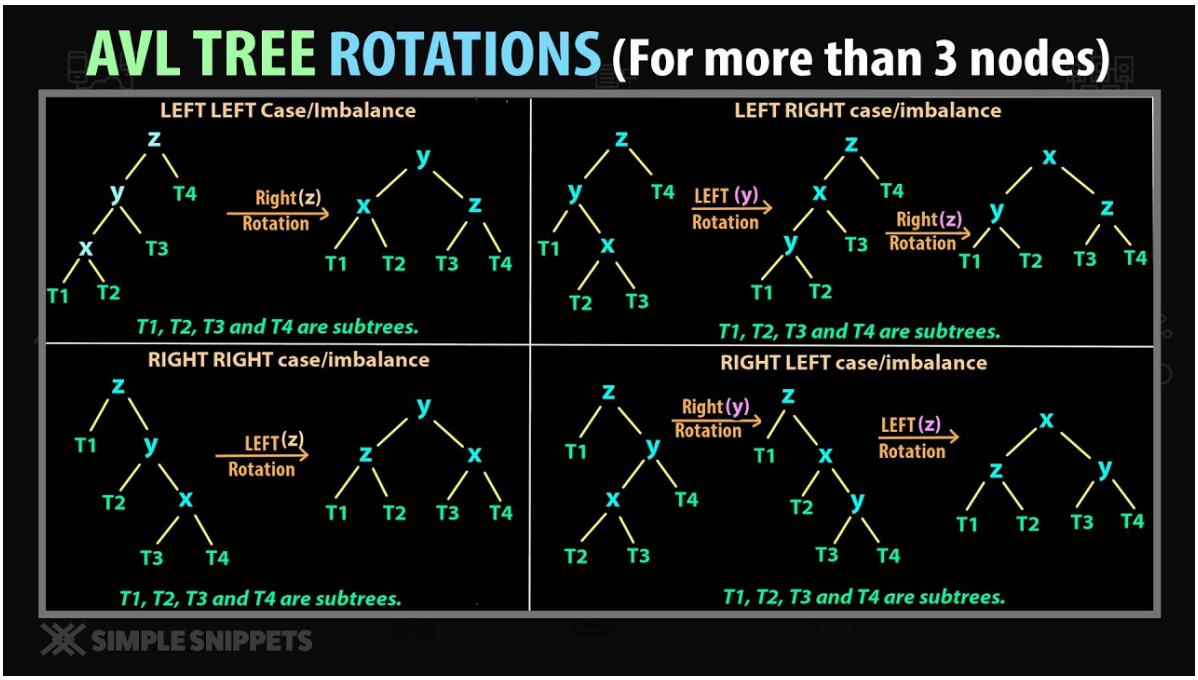


Figura 2: Rotatii[1]

[1] AVL Tree Rotations, https://www.youtube.com/watch?v=_nyt5QYel3Q

Exercitiul 6.(4p) Sa se implementeze o procedura pentru inserarea unui nod intr-un arbore binar de cautare echilibrat AVL. Sa se creeze folosind aceasta metoda un arbore binar de cautare echilibrat AVL din valorile existente intr-un fisier *arbore.in*. Sa se afiseze nodurile din arbore in pre-ordine pe ecran.

Exemplu:
arbore.in: 7 11 20 8 2 9 5 30 1 3 31 19
Output: 8 5 2 1 3 7 20 11 9 19 30 31

4 Observații

1. Prezentarea problemei (problemelor) se poate realiza în timpul laboratoarelor L și $L+1$ unde L este considerat laboratorul curent (în cazul de față laboratoarele 5 și 6).

2. Prezentarea se poate desfășura atât fizic în timpul laboratorului, cât și online printr-o programare stabilită anterior cu laborantul (pentru cazuri speciale: simptome COVID, probleme personale etc.)

3. Transmiterea laboratorului se realizează pe adresa de e-mail ruxandra.balucea@unibuc.ro pana în ziua laboratorului. Denumirea exercițiilor va fi de tipul x_grupa_Nume_Prenume (exemplu: 2_141_Pop_Ion). Toate aceste fișiere vor fi transmise într-o arhivă **grupa_Nume_Prenume.zip**.