Un gorun de 900 de ani care se află în satul Mercheaşa, comuna Homorod

Arbori

STEFAN I NITCHI † (26.04.2014)

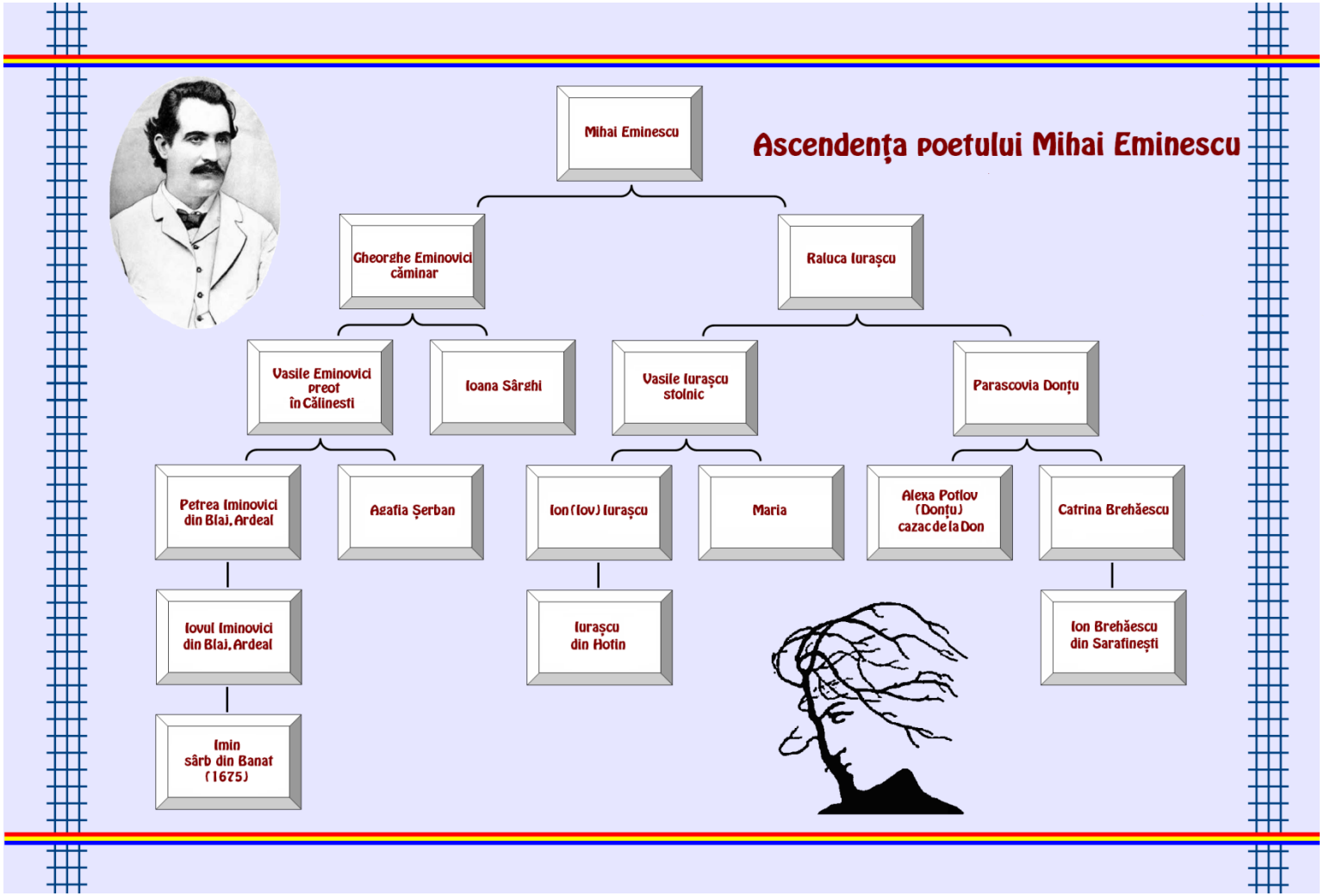
Concept de bază familiar în viaţa de zi cu zi.

Arborii sunt o abstractizare matematică care joacă un rol central în proiectarea şi analiza algoritmilor deoarece :

- utilizăm arbori pentru a descrie proprietăţile dinamice ale algoritmilor;
- construim şi utilizăm explicit structuri de date care sunt realizări concrete ale arborilor .

Exemple:

- Arborele genealogic (terminologia derivă în mare parte de aici)



❖Organizarea turneelor sportive (studiată şi de Lewis Carroll)

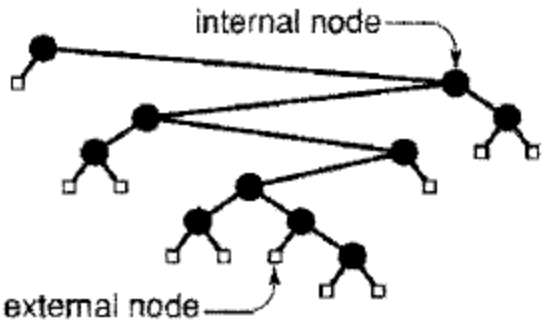
❖Organigrama unei corporaţii – această utilizare este sugestivă pentru descompunerea ierarhică cea care caracterizează algoritmii divide et impera.

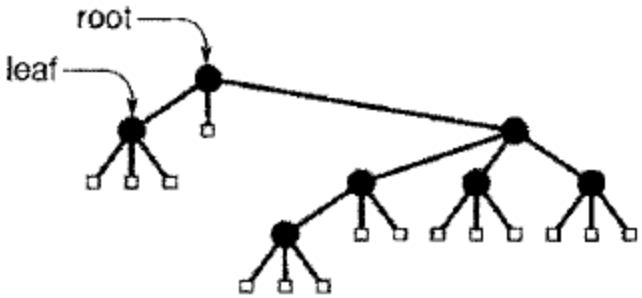
❖Arbore de analiză (sintactică) a unei propoziţii în părţile constituente, aceşti arbori sunt strâns legaţi de prelucrarea limbajelor pe calculator.

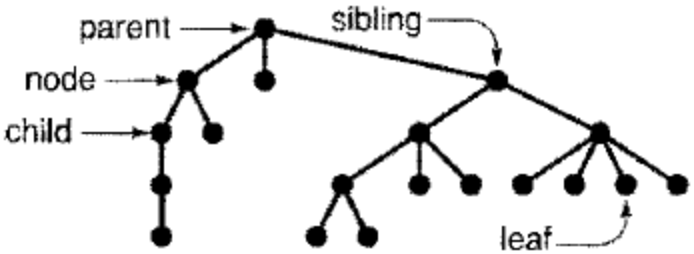
❖În aplicaţiile pe calculator una din cele mai cunoscute utilizări ale structurii de arbore este la organizarea sistemului de fişiere: fişierele sunt păstrate în *directori* (numiţi uneori şi *folder-e*) care sunt definiţi recursiv ca secvenţe de directori şi fişiere. Această definiţie recursivă reflectă o descompunere natural recursivă şi este identică cu definiţia unui anumit tip de arbore

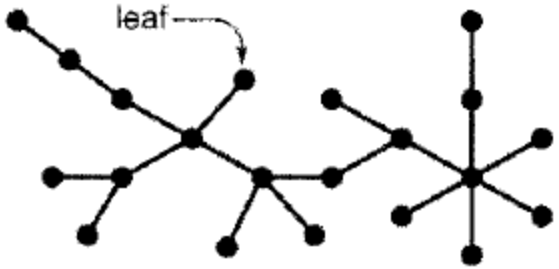
Există mai multe tipuri de arbori şi este important să înţelegem distincţia dintre abstractizare şi reprezentarea concretă cu care se lucrează într-o aplicaţie dată.

- Arbori liberi
- Arbori cu rădăcină
- Arbori ordonaţi
- Arbori binari şi arbori M-ari









Arbore binar
Arbore cu rădăcină

Arbore ternar
Arbore liber

(Sedgewick)

Un arbore este o colecţie nevidă de *vârfuri* şi *arce* care îndeplinesc anumite cerinţe.

Un vârf este un obiect simplu (numit şi nod) care poate avea un nume şi poate avea şi altă informaţie asociată.

Un arc este o conexiune între două vârfuri.

O cale într-un arbore este o listă de vârfuri distincte în care vârfuri succesive sunt legate prin arce în arbore.

Proprietatea definitorie pentru un arbore este că există o singură cale care uneşte *oricare* două noduri.

Dacă există mai multe căi sau nici una atunci vorbim despre un graf.

O mulţime disjunctă de arbori se numeşte pădure.
Un arbore cu rădăcină este un arbore în care am desemnat un nod ca rădăcina arborelui.

În informatică termenul de arbore este folosit pentru arbori cu rădăcină, iar termenul de arbori liberi pentru o structură mai generală.

Într-un arbore cu rădăcină, orice nod este rădăcină pentru un subarbore format din el şi din nodurile aflate sub el.

Există exact o cale între rădăcină şi fiecare alt nod din arbore. Definiţia nu presupune o direcţie pentru arce. Ne gândim la arce, în funcţie de aplicaţie, ca plecând toate de la rădăcină sau îndreptate toate spre rădăcină. De obicei reprezentăm arborii cu rădăcina sus, chiar dacă iniţial pare nenatural.

Spunem că un nod *y* se află *sub* nodul *x* (iar *x peste* nodul *y*) dacă x se află pe calea de la y la rădăcină.

Fiecare nod (exceptând rădăcina) are exact un nod deasupra lui. Acesta se numeşte *părinte*; nodurile aflate direct sub un nod se numesc *copii*. Uneori, în analogie cu arborele genealogic vorbim şi despre *străMoşul* unui nod.

Nodurile fără copii sunt numite *frunze* sau noduri *terminale*; nodurile cu cel puţin un copil sunt numite uneori *noduri interne (nonterminale)*.

În anumite aplicaţii ordinea în care sunt reprezentaţi copii este importantă, în altele nu.

Un arbore ordonat este un arbore cu rădăcină în care este specificată ordinea copiilor pentru fiecare nod.

Dacă fiecare nod *trebuie* să aibă un număr specificat de copii care apar într-o ordine anumită, atunci vorbim despre un arbore M-ar.

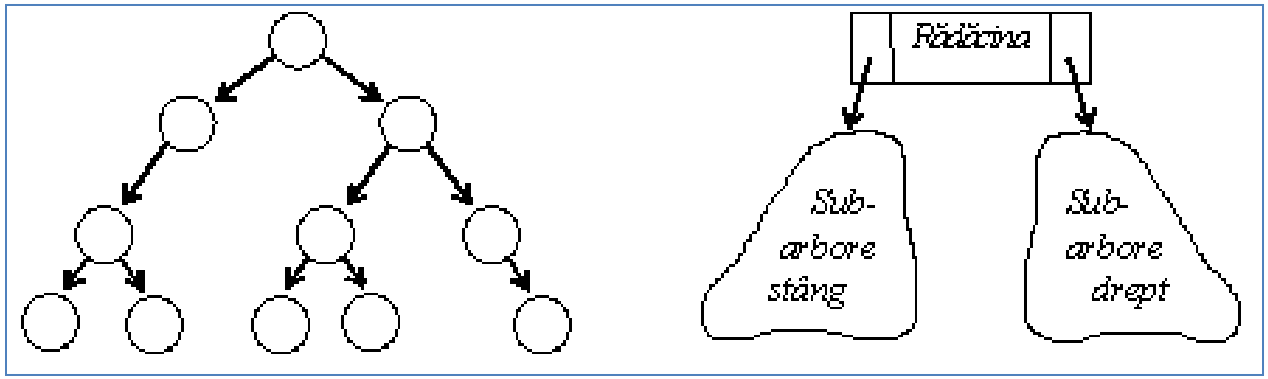
Într-un asemenea arbore adeseori se definesc *noduri externe speciale* care nu au copii. Acestea acţionează ca *pseudonoduri* pentru nodurile care nu au numărul specificat de copii.
Cel mai simplu tip de arbore M-ar este *arborele binar*.

Un arbore binar este un arbore ordonat având 2 tipuri de noduri: noduri externe care nu au copii şi noduri interne cu exact doi copii. Deoarece cei doi copii ai unui nod intern sunt ordonaţi, ne referim la *copilul stâng* şi la *copilul drept* al nodului intern. Orice nod intern trebuie să aibă atât un copil drept cât şi unul stâng chiar dacă unul sau ambii ar putea fi noduri externe.
O *frunză* într-un arbore M-ar este un nod intern ai cărui copii sunt toţi noduri externe.

Observaţie

Deci un *arbore binar* este un caz particular de *arbore ordonat*; un arbore ordonat este un caz particular de *arbore cu rădăcină*, iar un arbore cu rădăcină este un caz particular de *arbore liber*. Diferitele tipuri apar în mod normal în diverse aplicaţii şi este important să se cunoască deosebirile dintre ele atunci când se caută căi de reprezentare a arborilor cu structuri de date concrete.

Definiţie: Un arbore binar este fie un nod extern sau un nod intern legat de o pereche de arbori binari, care se numesc subarborele stâng şi subarborele drept ai acelui nod.

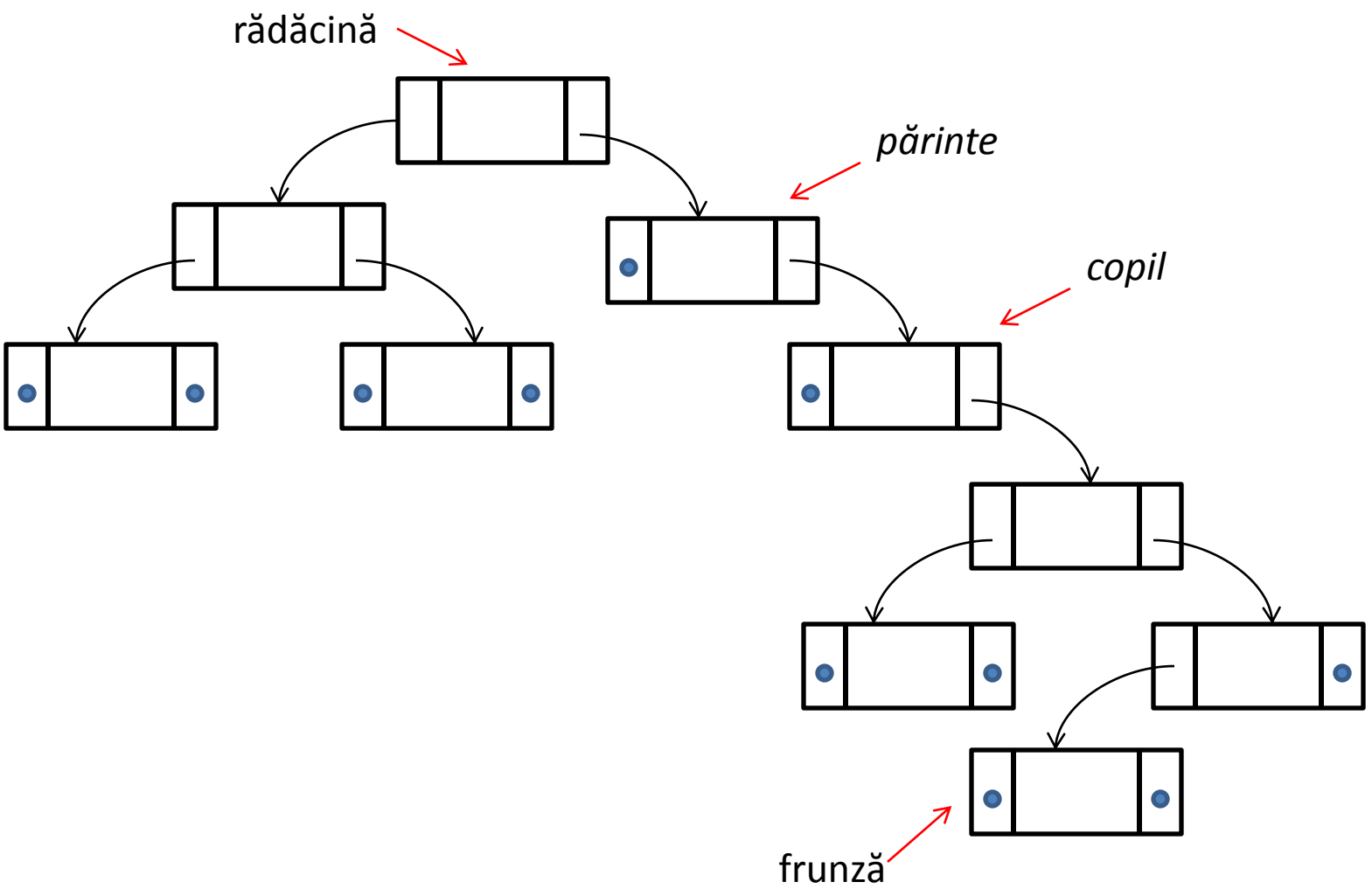


Cioban V.

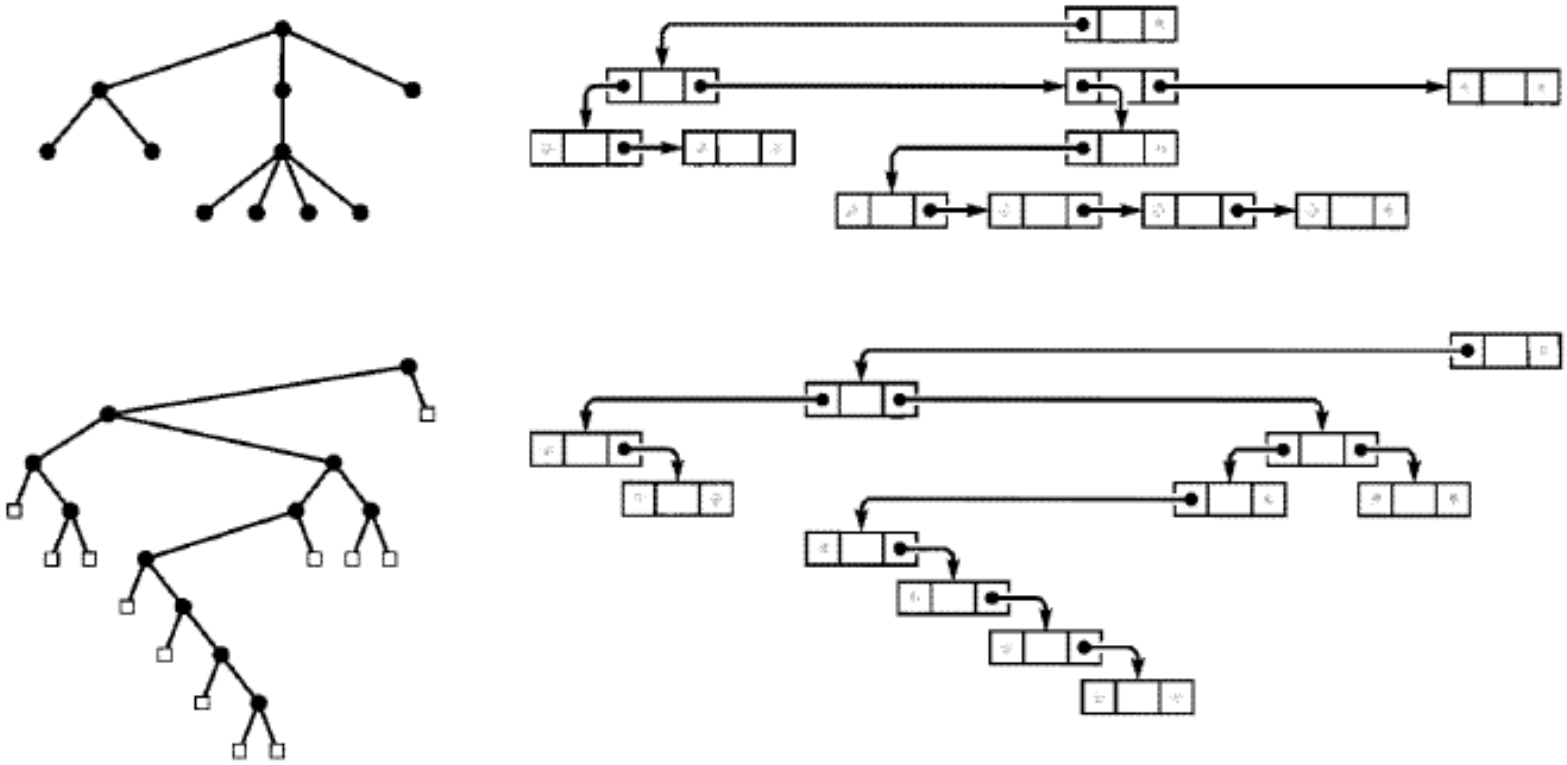
Reprezentarea concretă pe care o folosim cel mai adesea când se implementează programe care utilizează arbori binari, este o structură (record) cu două legături (pointeri) pentru nodurile interne: un pointer spre subarborele stâng şi un pointer spre subarborele drept. Structurile sunt asemănătoare cu cele de la liste înlănţuite cu deosebirea că au 2 link-uri pe un nod.

```
typedef struct node *link;
struct node { Item item; link l, r; };
```

Astfel implementăm operaţia abstractă *poziţionează-te pe arborele stâng* prin referinţa x=x->l



Reprezentarea unui arbore binar



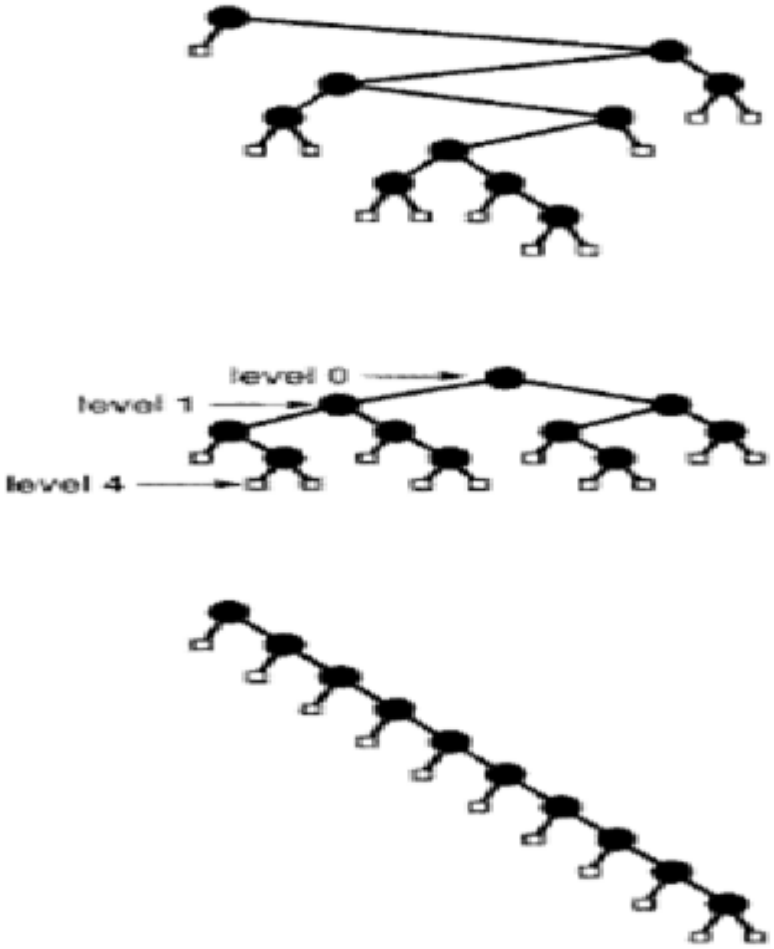
(Sedgewick)

Proprietate: Există o corespondenţă unu la unu între arbori binari şi arbori ordonaţi.

Proprietatea1. Un arbore binar cu N noduri interne are N+1 noduri externe.

Proprietatea2. Un arbore binar cu N noduri interne are 2N arce: N-1 spre noduri interne şi N+1 arce spre noduri externe.

Definiţie: Nivelul unui nod într-un arbore este cu 1 mai mare decât nivelul părintelui (cu rădăcina pe nivelul 0). Înălţimea unui arbore este maximul nivelurilor nodurilor arborelui. Lungimea drumurilor unui arbore este suma nivelurilor tuturor nodurilor arborelui. Lungimea drumurilor interne ale unui arbore binar este suma nivelurilor tuturor nodurilor interne ale arborelui. Lungimea drumurilor externe ale unui arbore binar este suma nivelurilor tuturor nodurilor externe ale arborelui.



Three binary trees with 10 internal nodes

(Sedgewick)

Proprietatea 3. Lungimea drumurilor externe ale oricărui arbore binar cu N noduri interne este cu 2N mai mare decât lungimea drumurilor interne.

Proprietatea 4. Înălţimea unui arbore binar cu N noduri interne este cel puţin lgN şi cel mult N-1.

Proprietatea 5. Lungimea drumurilor interne ale unui arbore binar cu N noduri interne este cel puţin Nlg(N/4) şi cel mult N(N-1)/2

Traversarea unui arbore

Dându-se un pointer spre un arbore, dorim să prelucrăm sistematic fiecare nod al arborelui. Acest proces îl vom analiza mai întâi pentru arbori binari.

Prin *traversarea* unui arbore binar vom înţelege parcurgerea tuturor nodurilor arborelui, trecând o singură dată prin fiecare nod.
În funcţie de ordinea (disciplina) de vizitare a nodurilor unui arbore binar, exista trei moduri de baza de traversare:

- *în preordine:* vizitează mai întâi nodul (rădăcină), apoi subarborele stâng şi după aceea subarborele drept
- *în inordine*: vizitează mai întâi subarborele stâng , apoi nodul (rădăcină) şi după aceea subarborele drept

- *în postordine*: vizitează mai întâi subarborele stâng , apoi subarborele drept şi după aceea nodul (rădăcină)

Evident, definiţiile sunt recursive, parcurgerea unui subarbore fiind facută după aceaşi regulă.

Traversare recursivă în preordine

```
void traverse(link h, void (*visit)(link))
{
if (h == NULL) return;
(*visit) (h);
traverse (h->l, visit);
traverse (h->r, visit);
```

Analog pentru celelalte tipuri de traversări – Temă!


„Nu există pustiu. Există doar incapacitatea noastră de a umple golul în care trăim.”
Octavian Paler