

Structuri de date – Curs 8

Conf. univ. dr. Cristian CIUREA
Departamentul de Informatica si Cibernetica Economica
Academia de Studii Economice din Bucuresti
cristian.ciurea@ie.ase.ro

Agenda

- ▶ Operații pe arbori binari de căutare
- ▶ Arbori echilibrați
- ▶ Arbori Roșu & Negru

Arbori binari de căutare

- ▶ Principala utilizare a arborilor binari de căutare este regăsirea rapidă a unor informații memorate în cheile nodurilor.
- ▶ Pentru orice nod al unui arbore de căutare, cheia acestuia are o valoare mai mare decât cheile tuturor nodurilor din subarborele stâng și mai mică decât cheile nodurilor ce compun subarborele drept.

Arbori binari de căutare

- ▶ Operațiile pe arbori binari de căutare se grupează în următoarele categorii:
 - operații de creare a arborilor;
 - operații cu elemente (noduri), precum operațiile de inserare și ștergere de noduri în și din arbore;
 - traversări de arbori:
 - pentru prelucrarea informației utile;
 - pentru căutarea de informație în arbore.
 - conversii și stocare în fișier.

Arbori binari de căutare

- ▶ Operația de creare:
 - presupune construirea în memorie a unui arbore binar prin preluarea de informații din mediul extern, sursele cele mai frecvente fiind introducerea de la tastatură sau fișierele;
 - algoritmi de creare a unui arbore binar presupun cunoașterea relațiilor în care se află un nod cu celelalte noduri din arbore.

Arbori binari de căutare

► Operația de creare:

- presupune adăugarea câte unui nod la un arbore inițial vid;
- după inserarea unui nod, arborele trebuie să rămână în continuare ordonat;
- pentru adăugarea unui nod se parcurge arborele începând cu rădăcina și continuând cu subarborele stâng sau drept în funcție de relația de ordine;
- traversarea se continuă până când se ajunge la un nod fără descendent;
- acestui nod îi va fi adăugat un nod fiu cu valoarea dorită a cheii.

Arbori binari de căutare

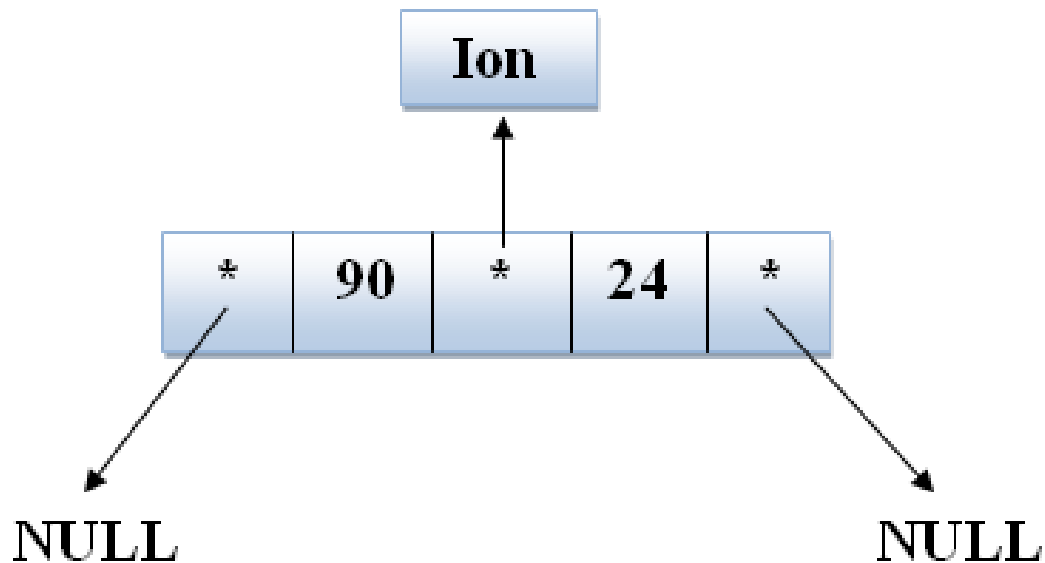
- ▶ Operația de creare a unui arbore binar de căutare, în care informația unui nod este de tip articol, compusă din câmpurile *cod*, *nume* și *vârsta*, asociate unui student.

```
struct student
{
    int cod;
    char* nume;
    int varsta;
};
```

```
struct bynarytreenode
{
    student info;
    bynarytreenode *left, *right;
};
```

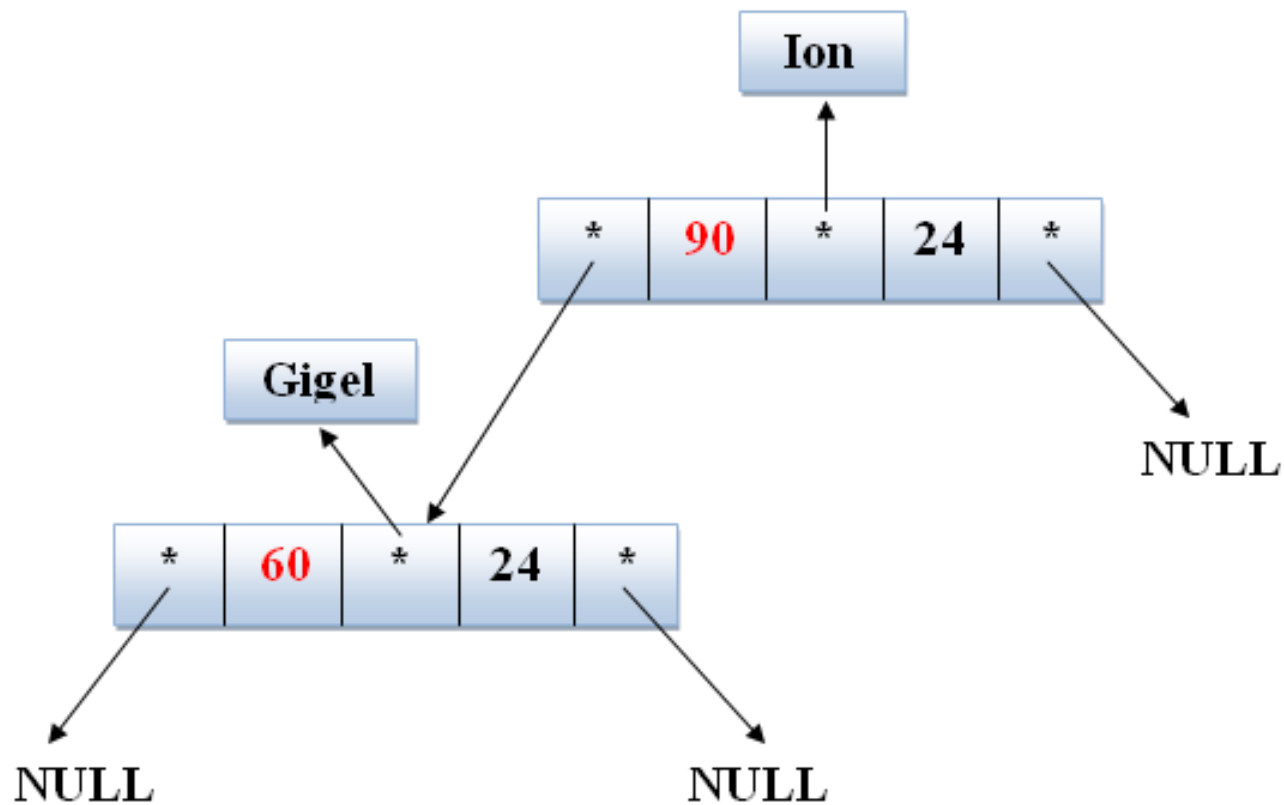
Arbori binari de căutare

- ▶ creare nod cu informația (90, "Ion", 24):



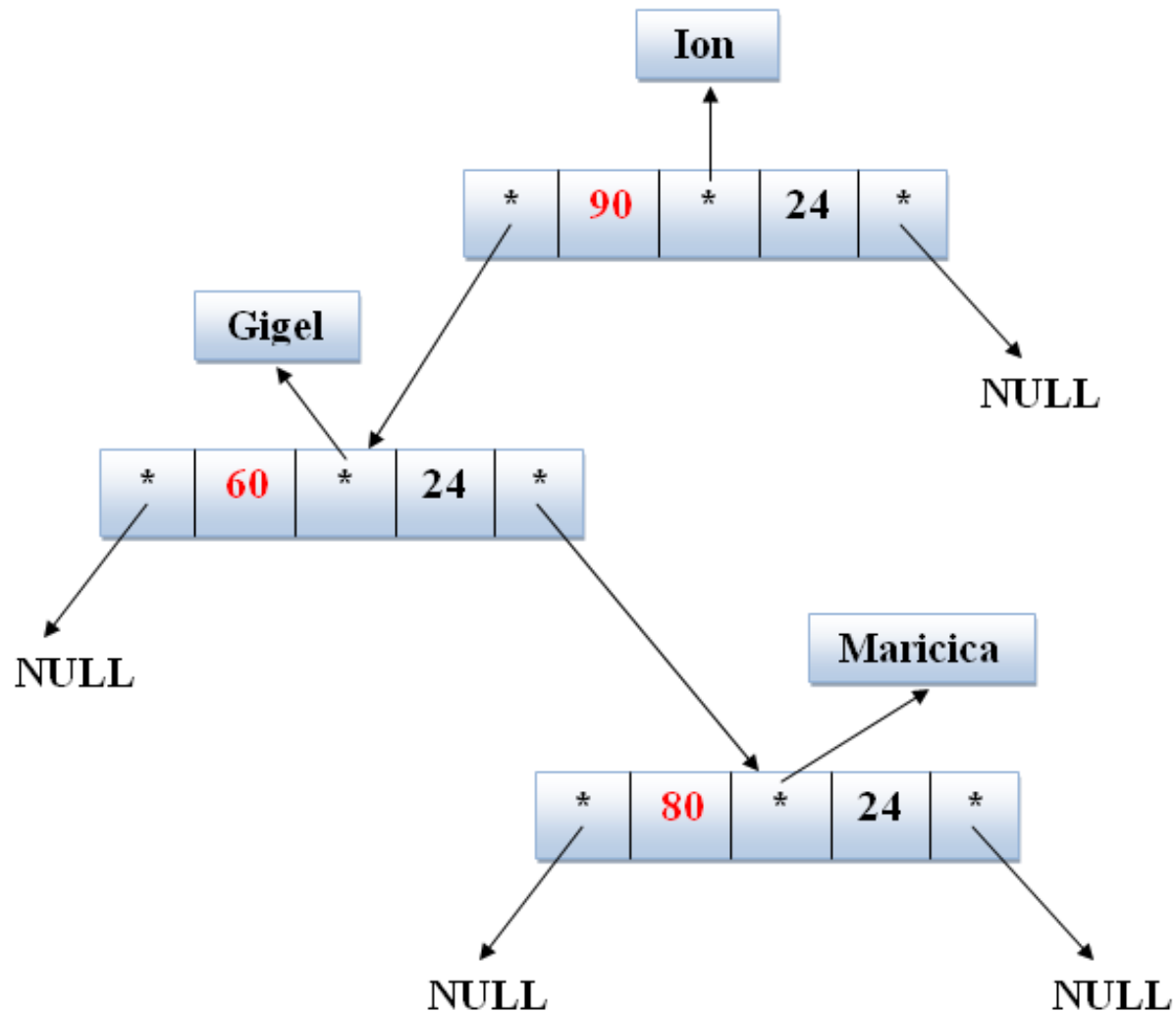
Arbori binari de căutare

- ▶ adăugare nod cu informația (60, "Gigel", 24):



Arbori binari de căutare

- ▶ adăugare nod cu informația (80, "Maricica", 24):



Arbori binari de căutare

- ▶ Operațiunile de inserare/ștergere:
 - operația de inserare a unui nod într-un arbore binar de căutare necesită specificarea relației în care se află nodul respectiv cu celelalte noduri din arbore;
 - operația de ștergere a unui nod implică formarea unor noi relații între nodurile rămase.

Arbori binari de căutare

- ▶ Operația de ștergere:
 - arborele trebuie să rămână arbore binar de căutare și după ștergerea unui nod;
 - nodul care va fi șters se va încadra într-una din variantele următoare:
 - nu are subarbori (fii);
 - are doar subarbore stâng;
 - are doar subarbore drept;
 - are atât subarbore stâng, cât și subarbore drept.

Arbori binari de căutare

- ▶ Operația de ștergere:
 - în cazul în care nodul nu are nici subarbore stâng, nici subarbore drept este necesară doar ștergerea nodului; nu sunt necesare alte operații de actualizare a arborelui;
 - în cazul în care nodul de șters are subarbore stâng sau drept, pe lângă ștergerea nodului este necesară și actualizarea legăturilor dintre nodurile arborelui, respectiv fiul nodului care va fi șters, dacă există, va deveni fiul tatălui acestuia.

Arbori binari de căutare

- ▶ Operația de ștergere:
 - cazul în care nodul de șters are atât subarbore stâng, cât și subarbore drept, necesită o tratare specială;
 - astfel, mai întâi se localizează fie cel mai din stânga fiu al subarborelui drept, fie cel mai din dreapta fiu al subarborelui stâng;
 - cheile acestor noduri reprezintă valoarea imediat următoare cheii nodului ce se dorește a fi șters.

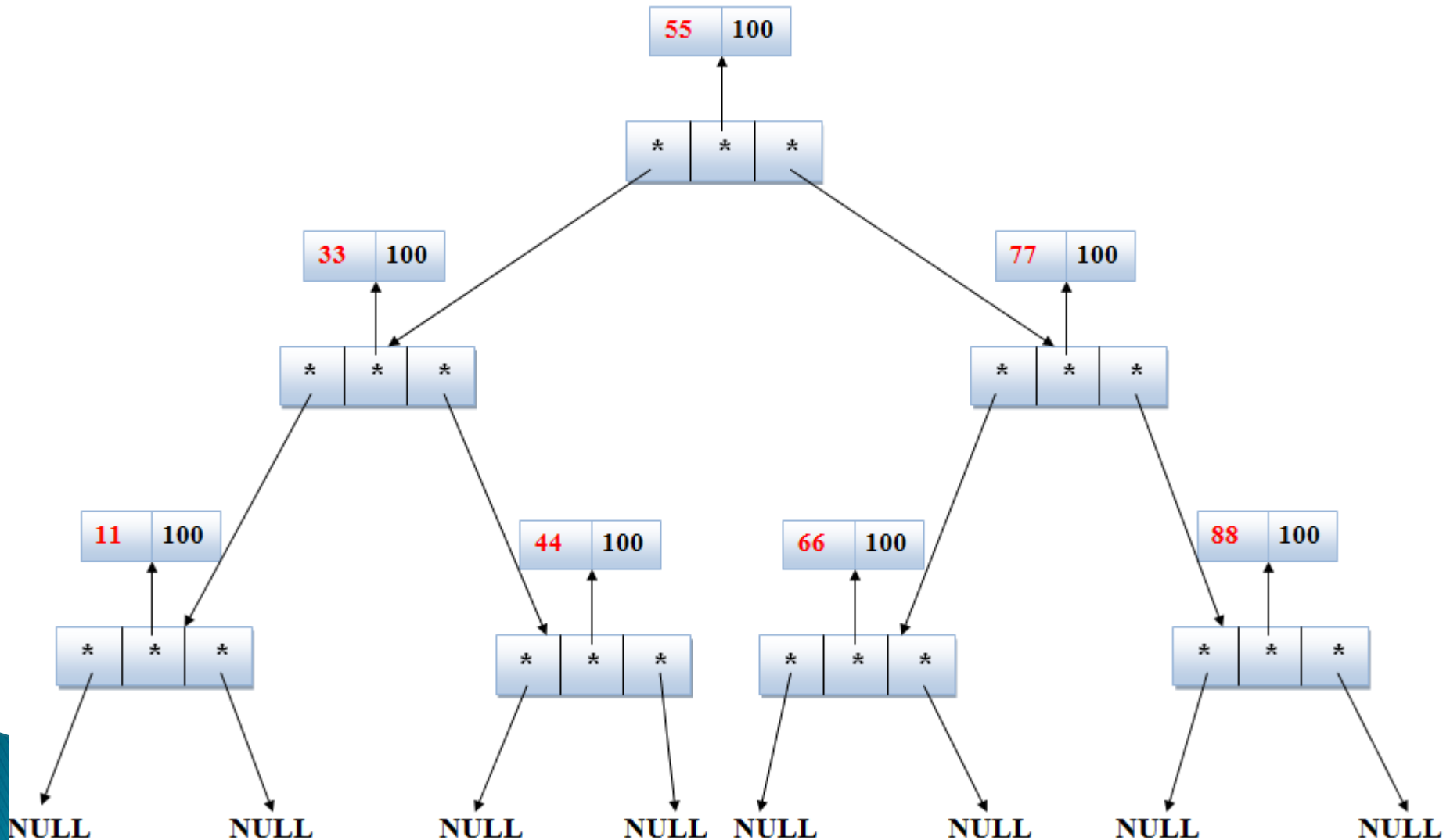
Arbori binari de căutare

- ▶ Operația de ștergere a unui nod dintr-un arbore binar de căutare, în care informația utilă este pointer la o structură "carte":

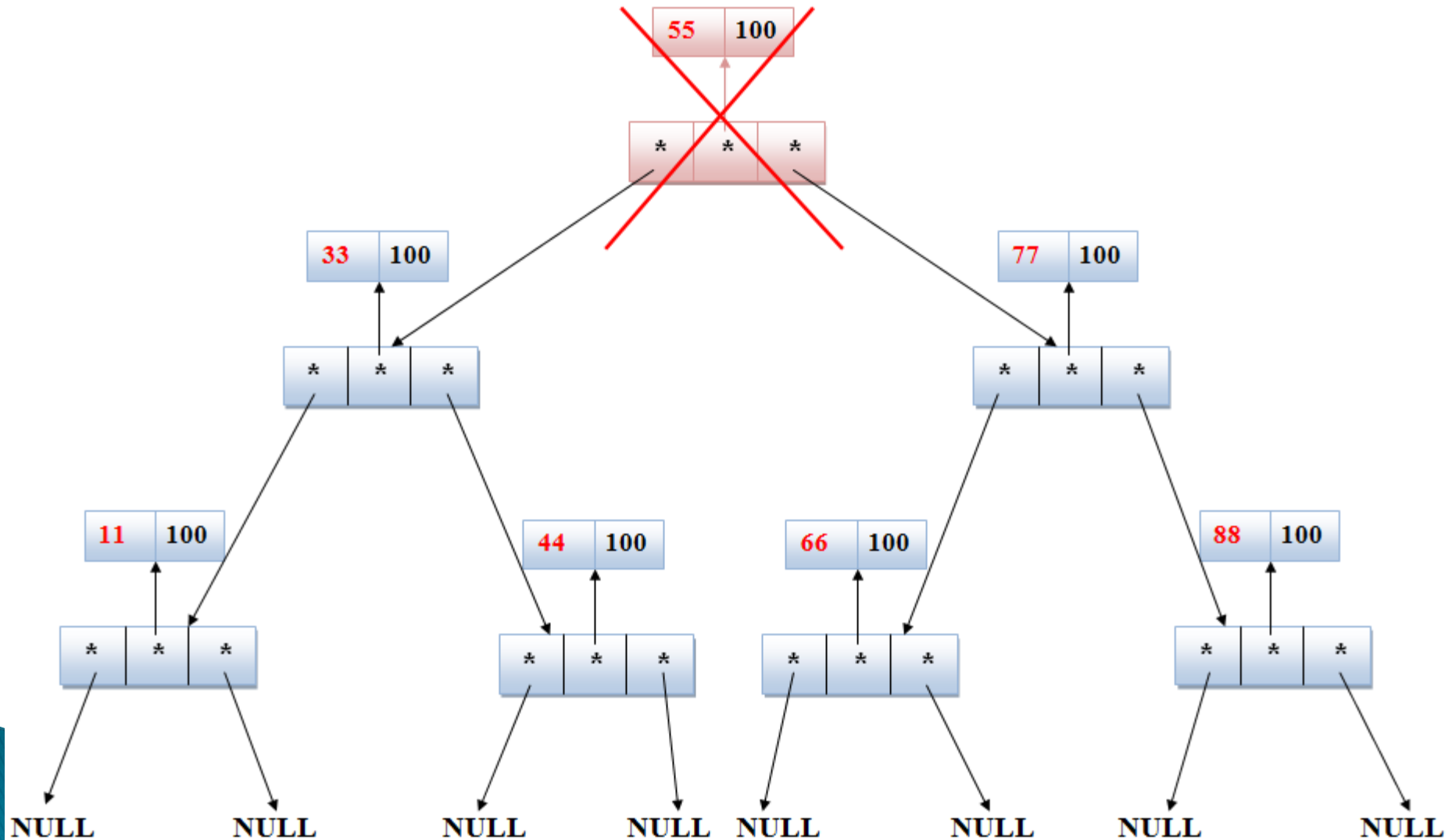
```
struct carte
{
    int ISBN;
    float pret;
};
```

```
struct bynarytreenode
{
    bynarytreenode *left;
    carte *inf;
    bynarytreenode *right;
};
```

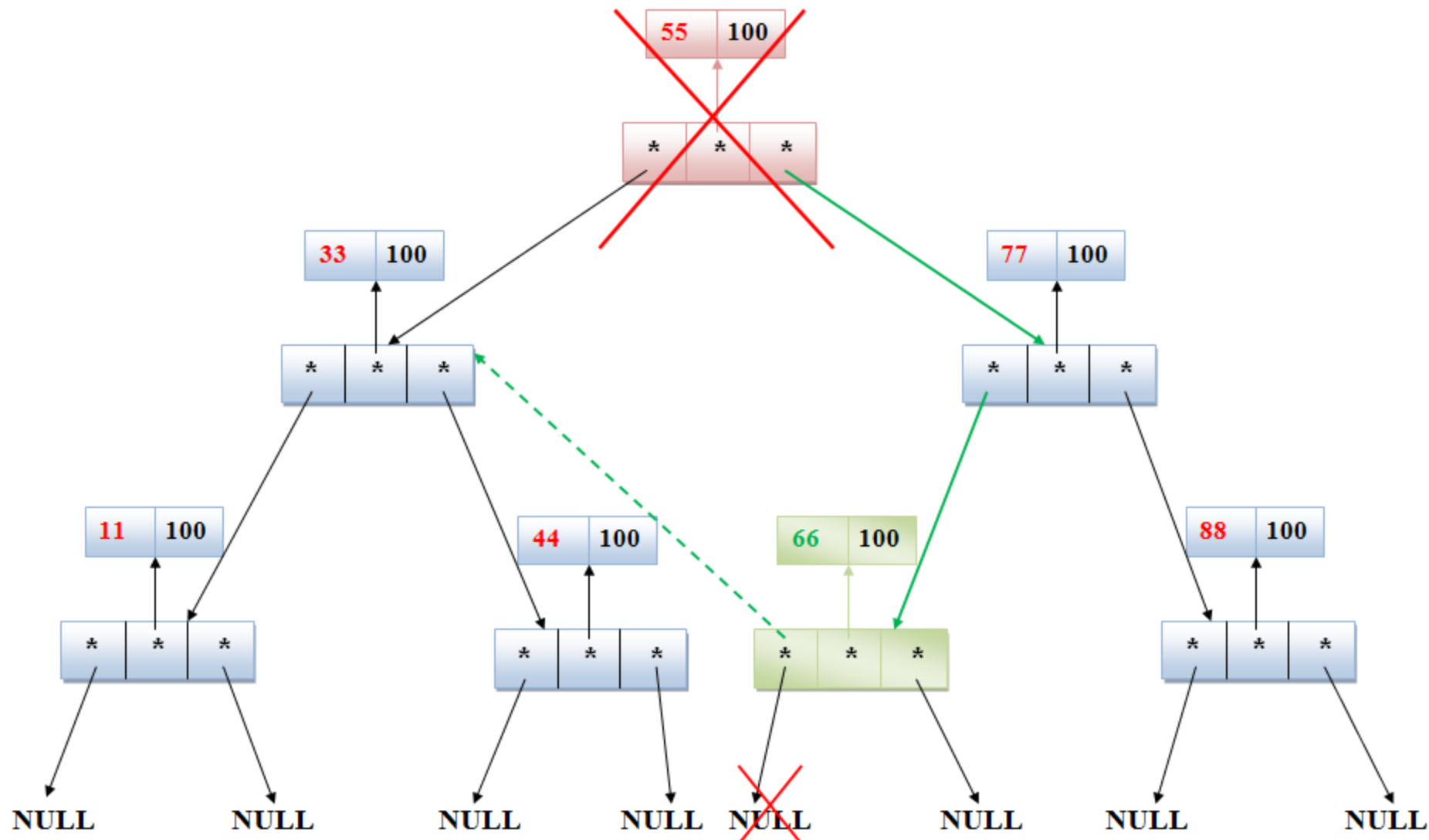
Arbori binari de căutare



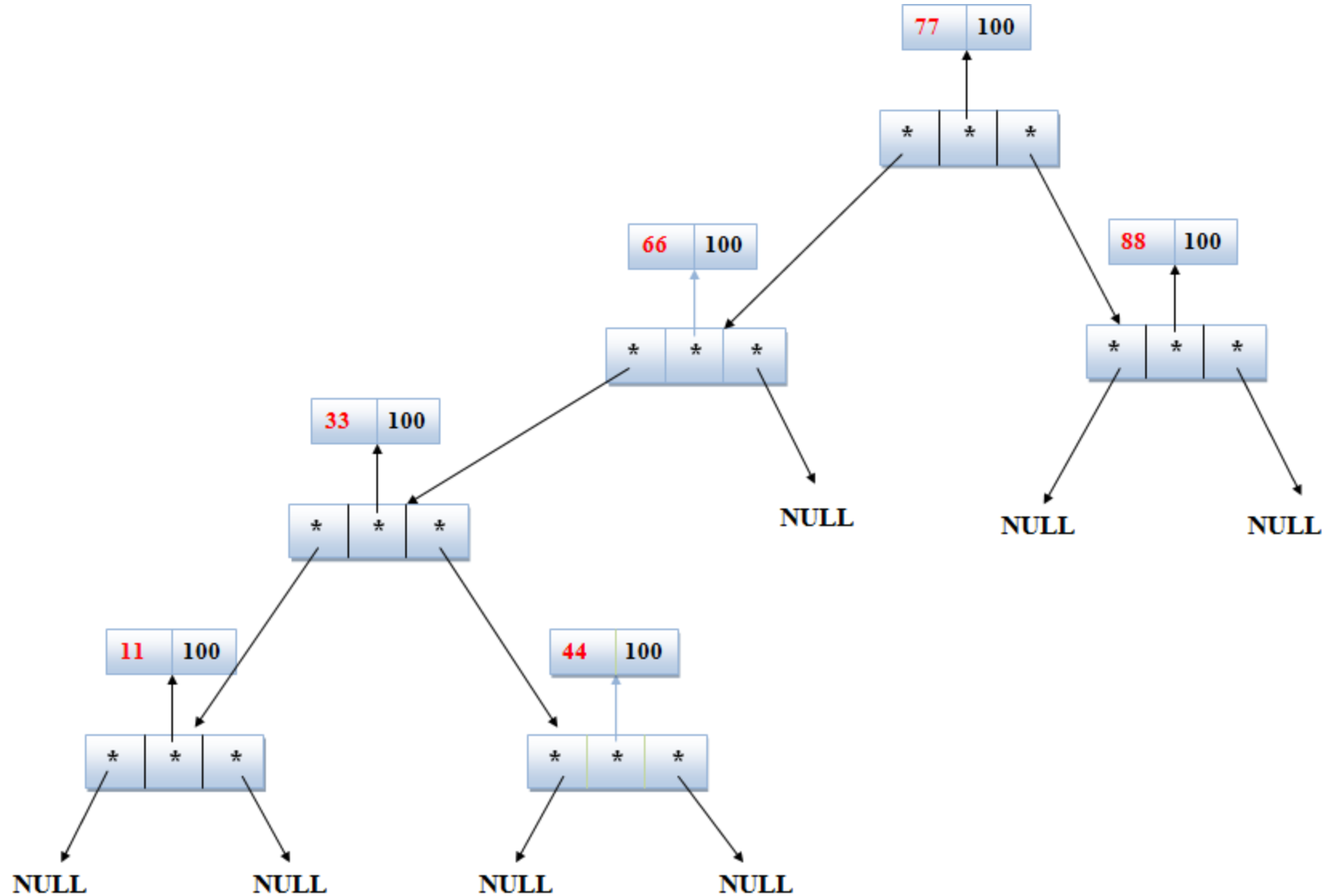
Arbori binari de căutare



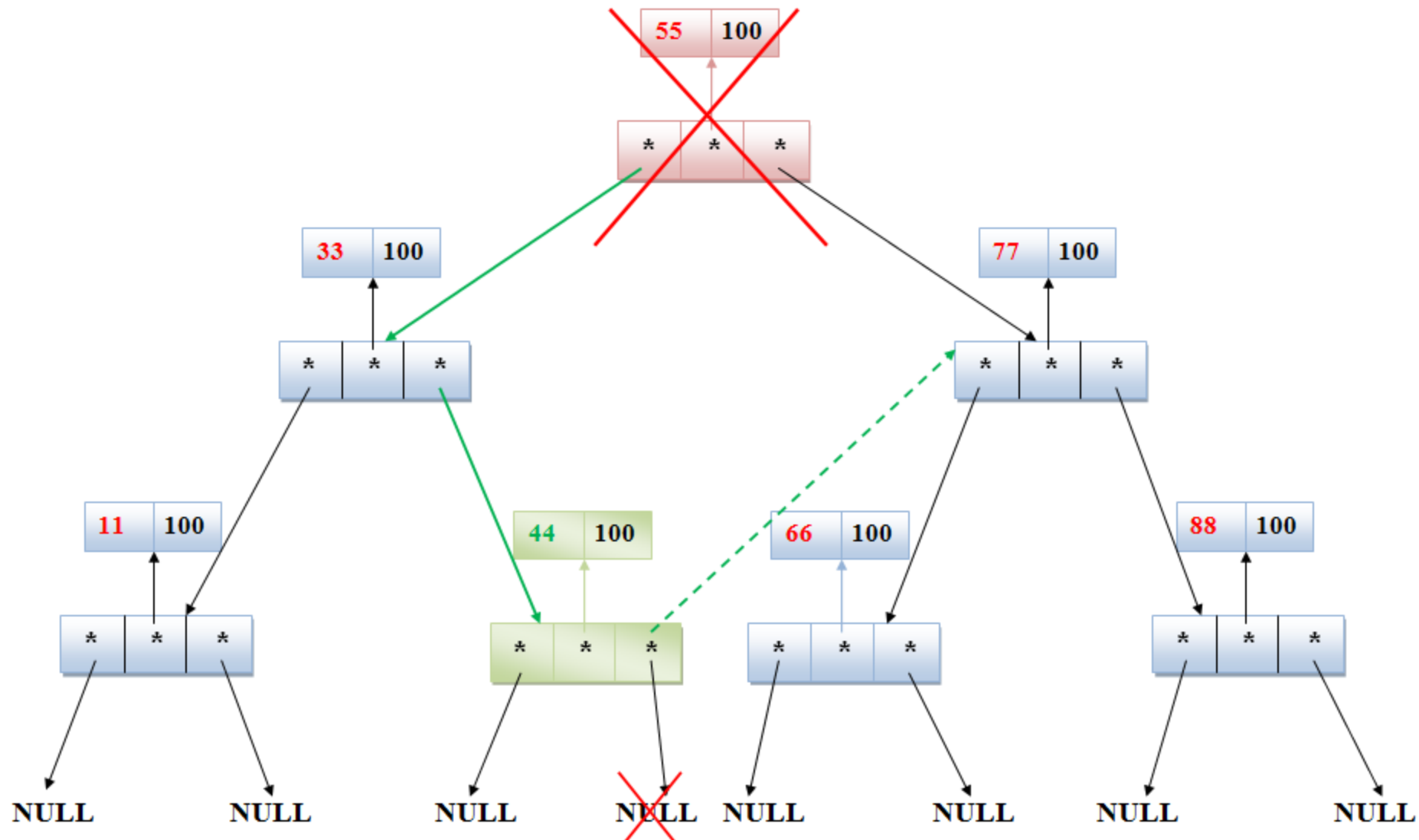
Arbori binari de căutare



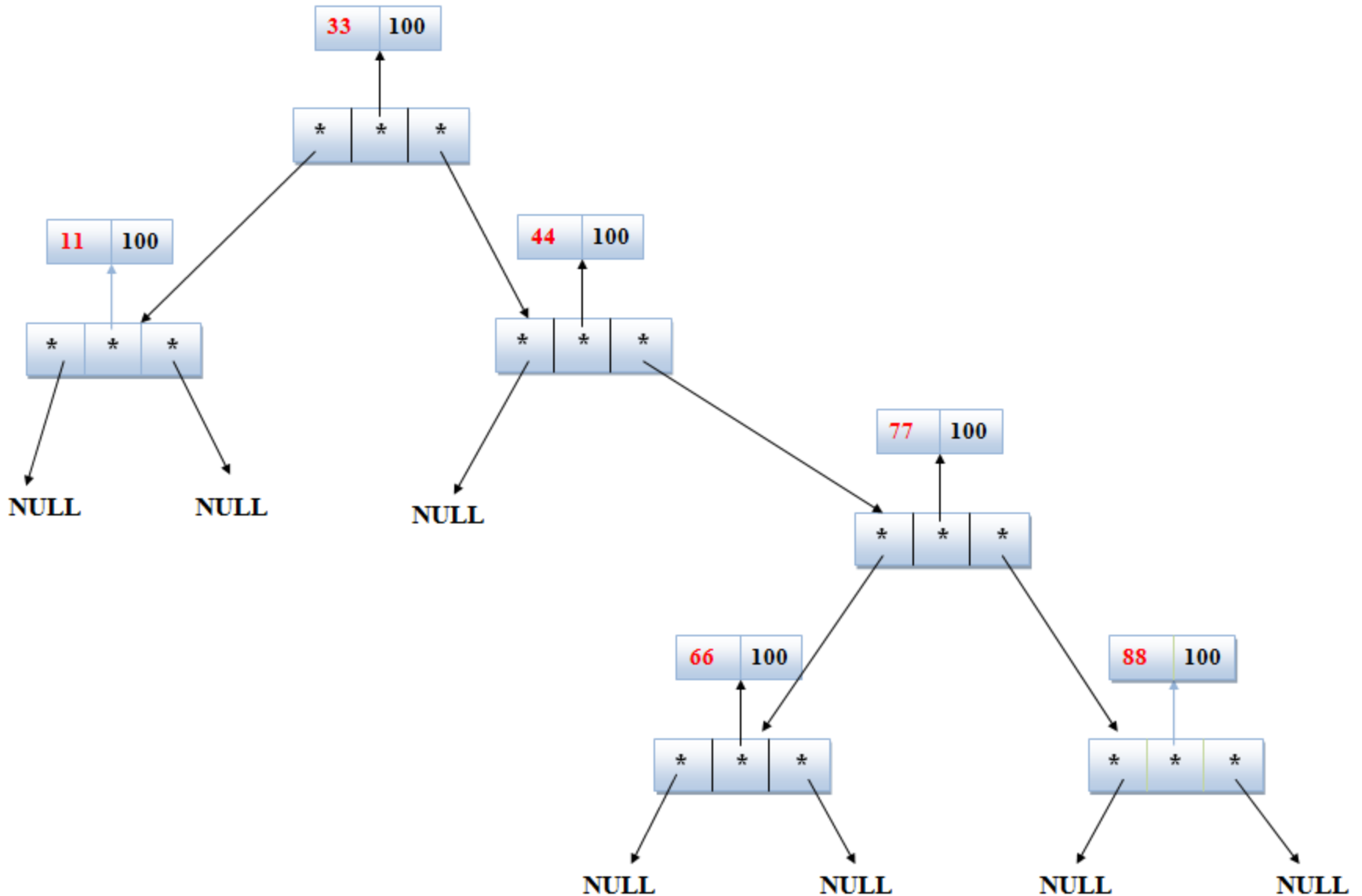
Arbori binari de căutare



Arbori binari de căutare



Arbori binari de căutare



Arbori binari de căutare

- ▶ Traversările arborilor binari de căutare:
 - pentru prelucrarea informației utile:
 - preordine (RSD);
 - inordine (SRD);
 - postordine (SDR);
 - pentru căutarea de informație în arbore:
 - în varianta recursivă;
 - în varianta iterativă.

Arbori binari de căutare

- ▶ Algoritmul pentru operația de căutare într-un arbore binar de căutare este următorul:
 - se compară cheia căutată cu cheia din rădăcină;
 - dacă sunt egale algoritmul se încheie;
 - dacă valoarea cheii căutate este mai mică decât valoarea din rădăcină, atunci se va relua algoritmul pentru subarborele stâng;
 - dacă nu există subarbore stâng, înseamnă că informația căutată nu se găsește în arbore;
 - altfel, dacă valoarea cheii căutate este mai mare decât valoarea cheii din rădăcină, se va relua algoritmul pentru subarborele drept;
 - dacă nu există subarbore drept, înseamnă că informația căutată nu se găsește în arbore;

Arbori binari de căutare

- ▶ căutare recursivă:

//căutare recursivă a cărții cu un anumit ISBN

```
bynarytreenode* cauta_recursiv(bynarytreenode* root, int  
    cheie)  
{  
    if (root)  
    {  
        if (cheie==root->inf->ISBN) return root;  
        else if (cheie<root->inf->ISBN)  
            cauta_recursiv(root->left, cheie);  
        else cauta_recursiv(root->right, cheie);  
    }  
    else return NULL;  
}
```


Arbori binari de căutare

- ▶ Operațiunile de conversie și stocare în fișier:
 - presupun traversarea arborilor binari de căutare și salvarea informațiilor în alte structuri de date, aflate în memorie, sau în fișiere, pe medii de stocare externe.

Arbori binari de căutare

- conversie arbore – listă:

//conversie arbore binar – listă simplă (varianta recursivă)

```
void bynarytree_to_list(bynarytreenode *root)
```

```
{
```

```
    if (root)
```

```
    {
```

```
        cap=inserare(cap, root->inf);
```

```
        bynarytree_to_list(root->left);
```

```
        bynarytree_to_list(root->right);
```

```
    }
```

```
}
```

Arbori binari de căutare

- conversie arbore – vector:

```
//conversie arbore binar – vector (varianta recursivă)
void bynarytree_to_array(bynarytreenode *root, book *v,
    int *k)
{
    if (root)
    {
        v[*k].ISBN=root->inf->ISBN;
        v[*k].price=root->inf->price;
        (*k)++;
        bynarytree_to_array(root->left, v, k);
        bynarytree_to_array(root->right, v, k);
    }
}
```

Arbori binari de căutare

- conversie arbore – fișier:

```
//conversie arbore – fisier text (varianta recursivă)
void bynarytree_to_file(FILE *f, bynarytreenode* root)
{
    if (root)
    {
        bynarytree_to_file(f, root->left);
        fprintf(f, "ISBN = %d, price = %5.2f",
                root->inf->ISBN, root->inf->price);
        bynarytree_to_file(f, root->right);
    }
}
```

Arbori echilibrați

- ▶ Un arbore binar de căutare este **echilibrat** dacă pentru orice nod, **înălțimile** celor doi subarbori diferă cu cel mult o unitate.
- ▶ Arborele **perfect echilibrat** este cel în care, pentru fiecare nod, **numărul de noduri** ale subarborelui stâng diferă de cel pentru subarboarele drept cu cel mult o unitate.

Arbori echilibrați

- ▶ Deoarece operațiunile de inserare și ștergere nu permit menținerea echilibrării perfecte a unui arbore, iar restructurarea arborelui după aceste operații este un proces complex, de obicei se preferă arbori imperfect echilibrați care vor fi numiți **arbori echilibrați**.

Arbori echilibrați

- ▶ Tipologii de arbori binari de căutare echilibrați:
 - arbori AVL (AVL Trees);
 - arbori Roșu & Negru (Red-Black Trees).

Arbori Roșu & Negru

Caracteristici:

- ▶ tipologie de arbori binari de căutare echilibrați;
- ▶ definiți de Rudolf Bayer în 1972 sub formă de arbori simetrici;
- ▶ nodurile sunt plasate în mod simetric în subarborii stânga sau dreapta;
- ▶ factorul cel mai important este dat de culoarea fiecărui nod.

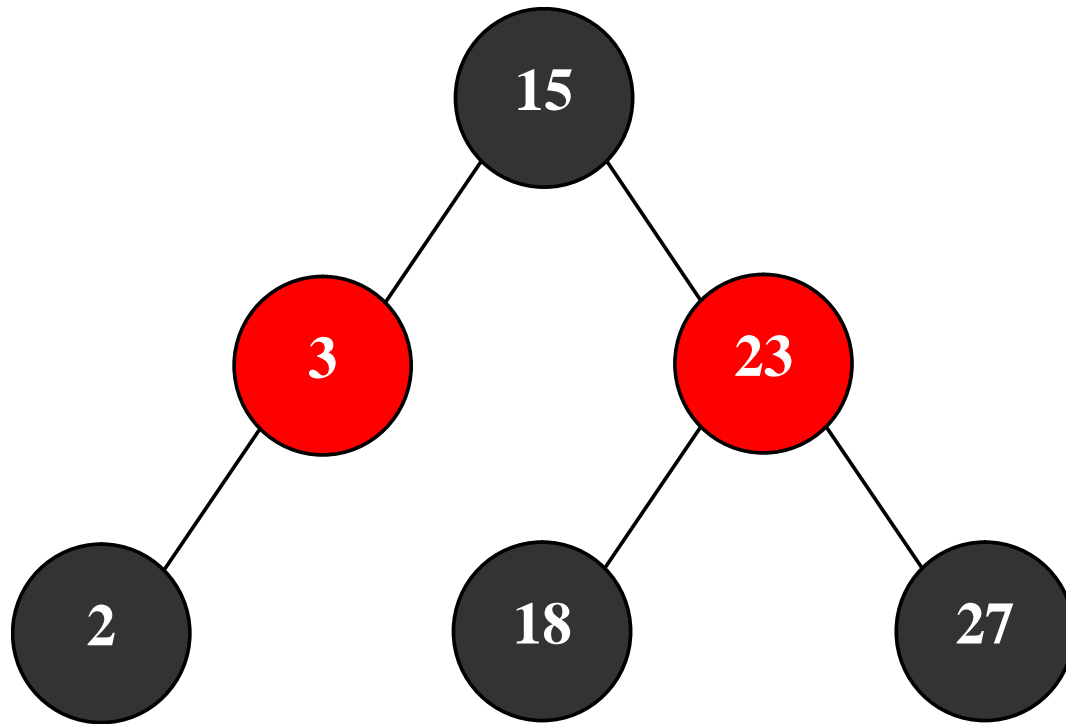
Arbori Roșu & Negru

Caracteristici:

- ▶ fiecare nod are una dintre cele două culori, roșu sau negru;
- ▶ nodul rădăcină este întotdeauna negru;
- ▶ ambele noduri fiu ale unui nod părinte roșu sunt negre;
- ▶ un nod roșu nu poate avea ca părinte decât un nod negru;
- ▶ toate drumurile de la rădăcină la oricare din nodurile frunză conțin același număr de noduri negre.

Arbori Roșu & Negru

- ▶ arbore roșu & negru:

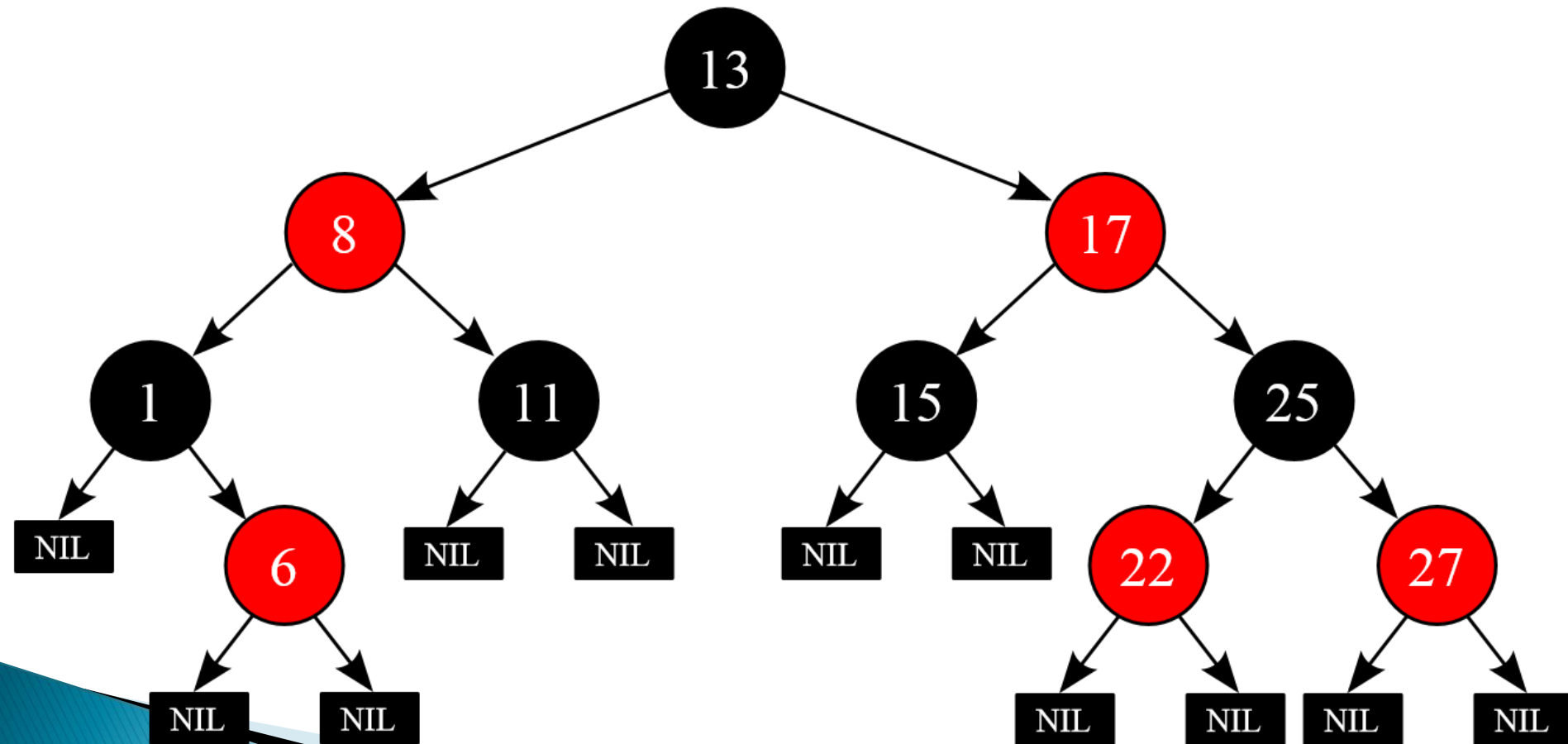


Arbori Roșu & Negru

- ▶ Eficiența arborelui Roșu & Negru față de arborele binar de căutare:
 - în arborele Roșu & Negru nu există pe un drum două noduri adiacente de culoare roșie, deoarece orice nod roșu are ambii fii de culoare neagră;
 - dacă cel mai scurt drum din arbore are numai noduri negre în număr de k , atunci cel mai lung drum din arbore are maxim $2*k$; demonstrația reiese din faptul că toate drumurile din arbore au același număr de noduri negre, ceea ce conduce la concluzia că drumul cel mai lung poate fi format doar din perechi de noduri adiacente de culori opuse.

Arbori Roșu & Negru

- ▶ arbore roșu & negru:



Arbori Roșu & Negru

- ▶ Pentru a facilita implementarea operațiilor cu arbori de tip Roșu & Negru se propune următoarea structură a nodului:

```
struct RNnod
{
    int info;
    bool culoare;
    RNnod *st;
    RNnod *dr;
    RNnod *parinte;
};
```

Bibliografie

- ▶ Ion Ivan, Marius Popa, Paul Pocatilu (coordonatori) – *Structuri de date*, Editura ASE, București, 2008.
 - Cap. 12. Arbori binari și arbori de căutare