UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Structuri de Date și Algoritmi

Sergiu Corlat, 2021

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Structuri de Date și Algoritmi

Lecția 6:

- Structuri de date neliniare – **arbori binari de căutare (ABC / BST)**
  - Proprietăți
  - Operații pe arborii ABC
  - Complexitatea operațiilor

Sergiu Corlat, 2021

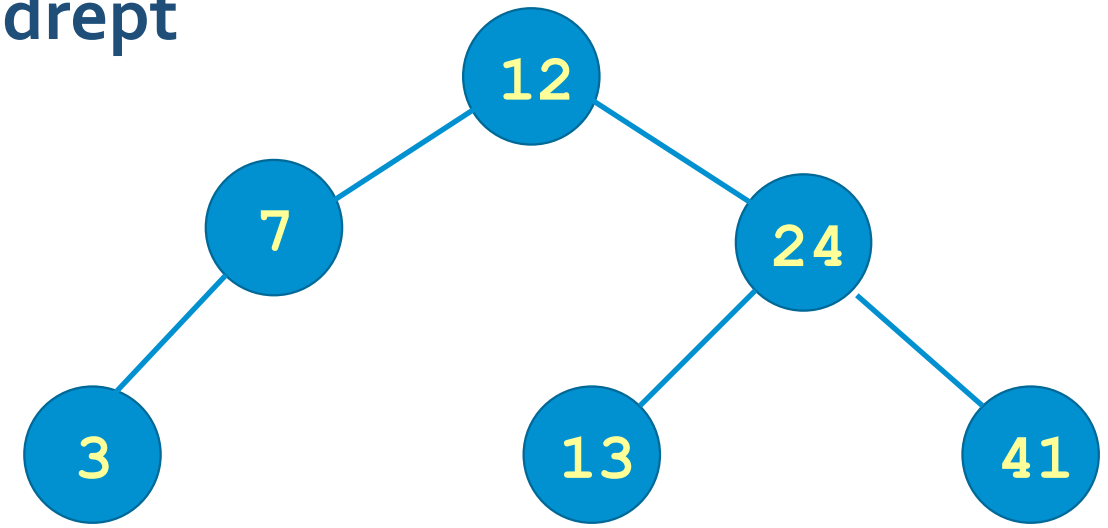
UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Definiții:

Definiții:

Arborii binari de căutare sunt arborii binari ordonați după cheia k a căror noduri au proprietatea:

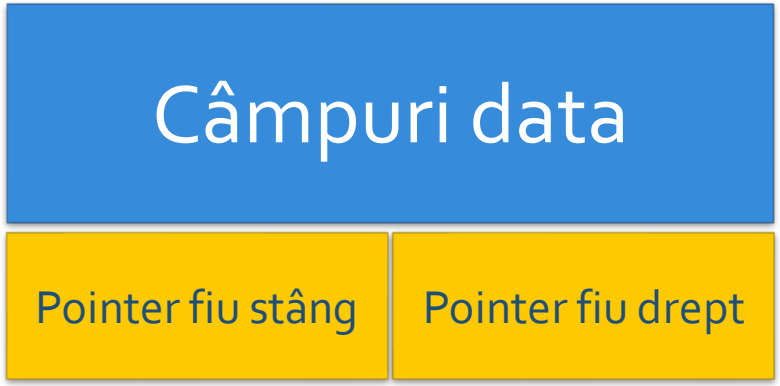
valoarea cheii k în nodul Z este mai mare decât valoarea aceleiași chei în oricare din nodurile subarborelui stâng și mai mică decât valoarea aceleiași chei în oricare din nodurile subarborelui drept



UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Structura nodurilor:

Structura clasică




```
struct node {
    int data;
    struct node *left;
    struct node *right;
};
```

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Operații pe arborii ABC:

Operații care implică modificarea structurii:

- Inserarea unui element (nod)
- Excluderea unui element (nod)

Adăugarea unui element

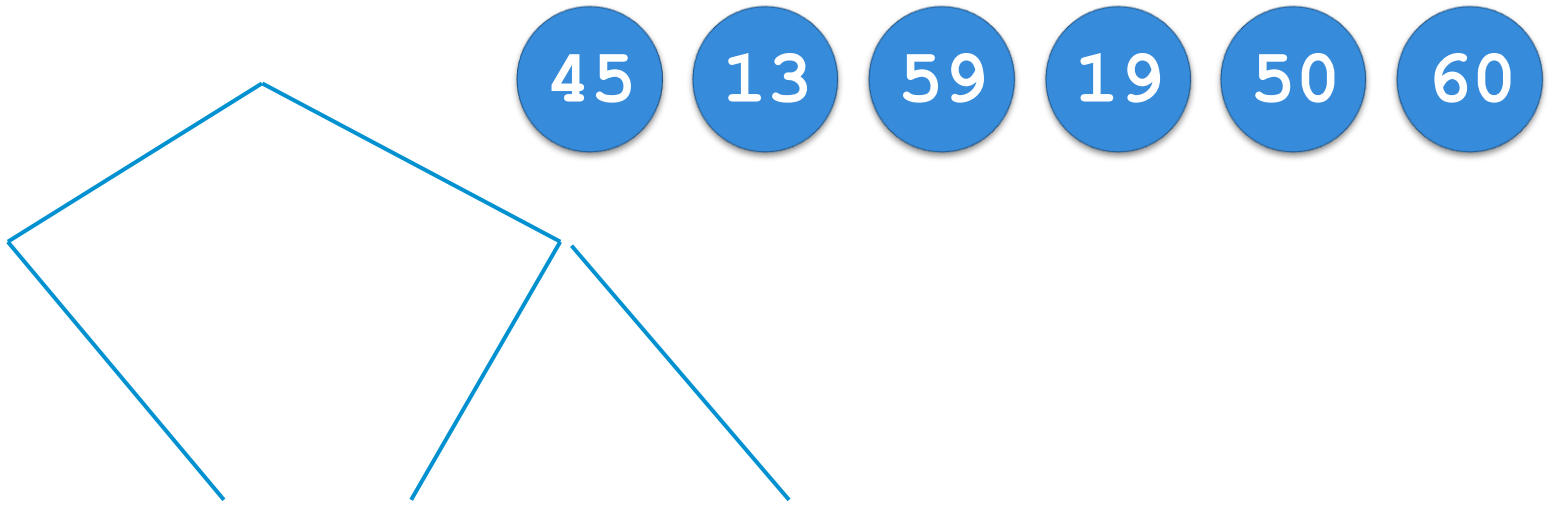
Un algoritm simplu, realizat recursiv:

Dacă nodul curent este NULL, creăm un nod nou, atribuim valoarea de adăugat, setăm descendenții nodului creat ca fiind NULL și conectăm nodul la nodul din care am venit,

În caz contrar:

- dacă valoarea de adăugat este mai mică sau egală cu valoarea nodului curent repetăm procedura din fiul stâng al acestuia
- dacă valoarea de adăugat este mai mare decât valoarea nodului curent repetăm procedura din fiul drept al acestuia

Adăugarea unui element



Adăugarea unui element

```
struct nod *insert(struct nod *nod, int val)
{
        if (nod == NULL)
                    return nod_nou(val);
          else
              {
                if(val > nod -> valoare )
                        nod -> dr = insert(nod - >dr, val);
                 else if( val < nod -> valoare )
                        nod -> st = insert(nod -> st, val);
              }
          // nimic de facut - valoarea deja exista
          return nod;
}
```

```
struct nod *nod_nou(int val)
{
    struct nod *tmp = (struct nod*)malloc(sizeof(struct nod));
    tmp -> valoare = val;
    tmp -> st = tmp -> dr = NULL;

    return tmp;
}
```

Adăugarea unui element – crearea elementului nou

Excluderea unui element

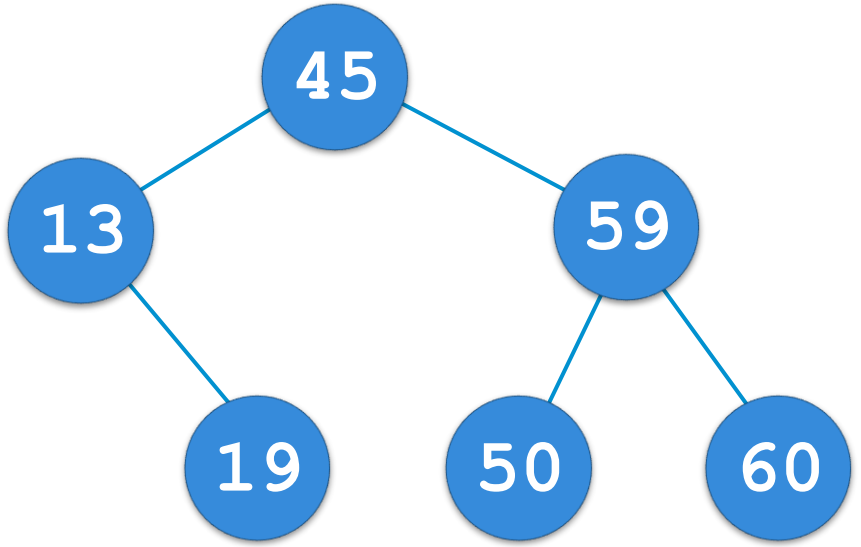
Arborii ABC sunt organizați conform unor principii, care urmează să rămână valide după excluderea nodului selectat.

Vom analiza cazurile posibile:

- Nodul exclus nu are fii
- Nodul exclus are doar fiul drept
- Nodul exclus are doar fiul stâng
- Nodul exclus are ambii fii

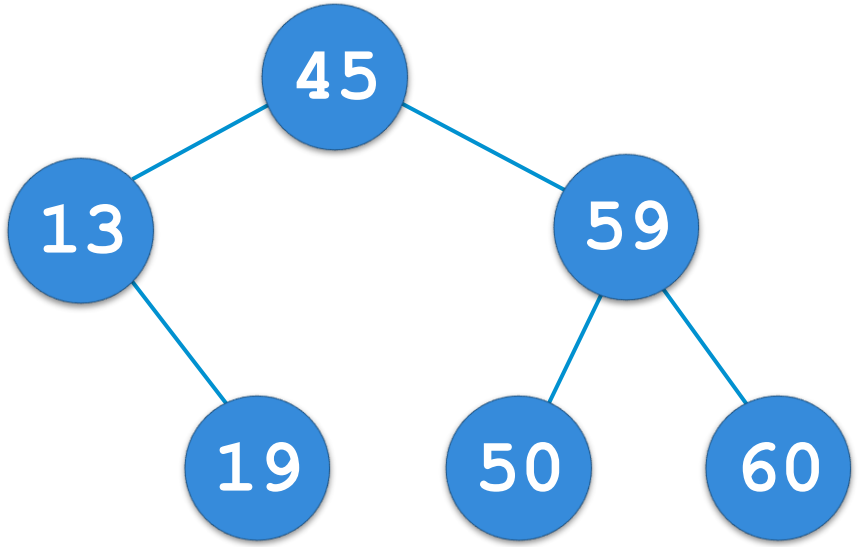
Excluderea unui element

Cazul A – fără fii



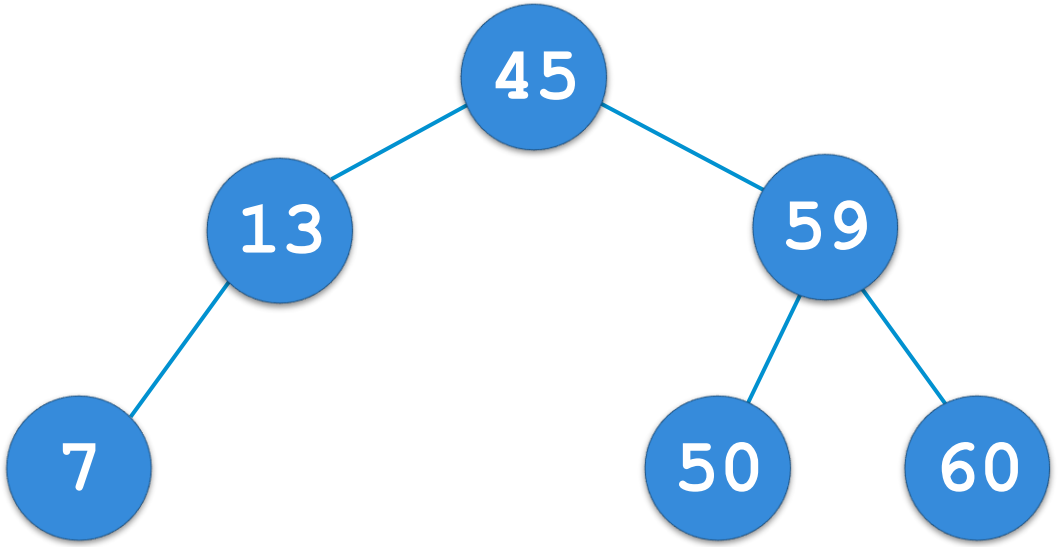
Excluderea unui element

Cazul B – nodul lichidat are doar fiul drept



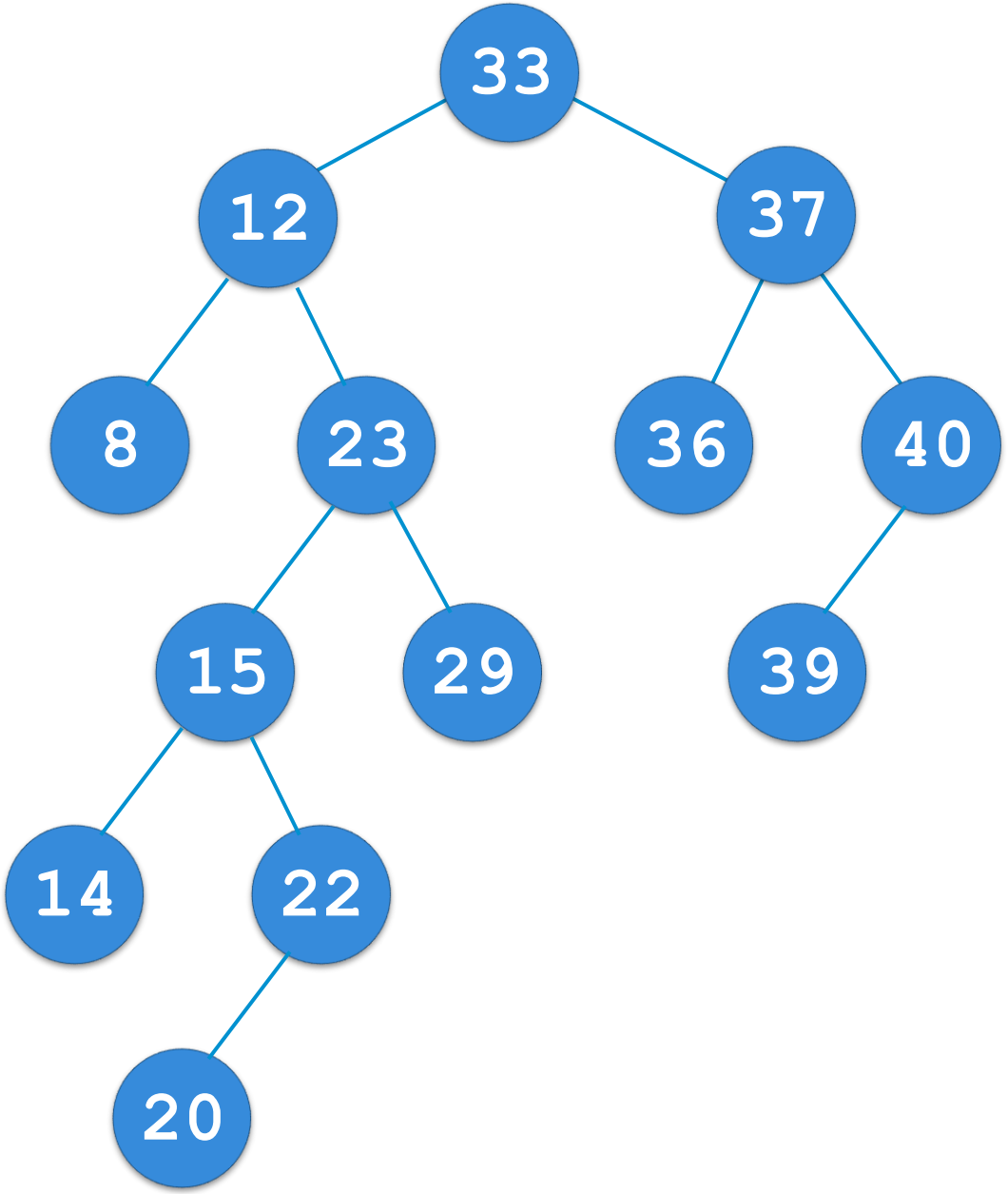
Excluderea unui element

Cazul C – nodul lichidat are doar fiul stâng



Excluderea unui element

Cazul D – nodul lichidat are ambii fii



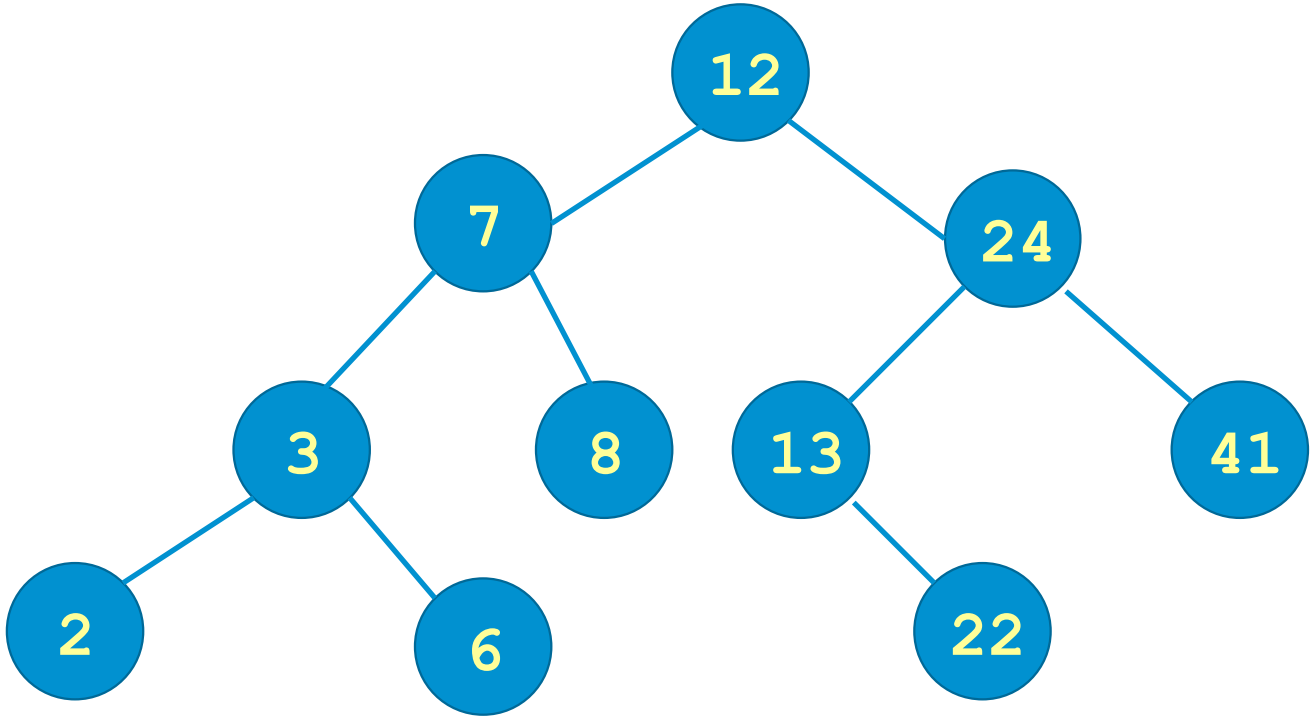
Operații care NU implică modificarea structurii:

- Parcurgerea arborelui
- Căutarea unui element

Parcurgerea arborilor ABC - după același algoritm ca parcurgerea arborilor binari oarecare

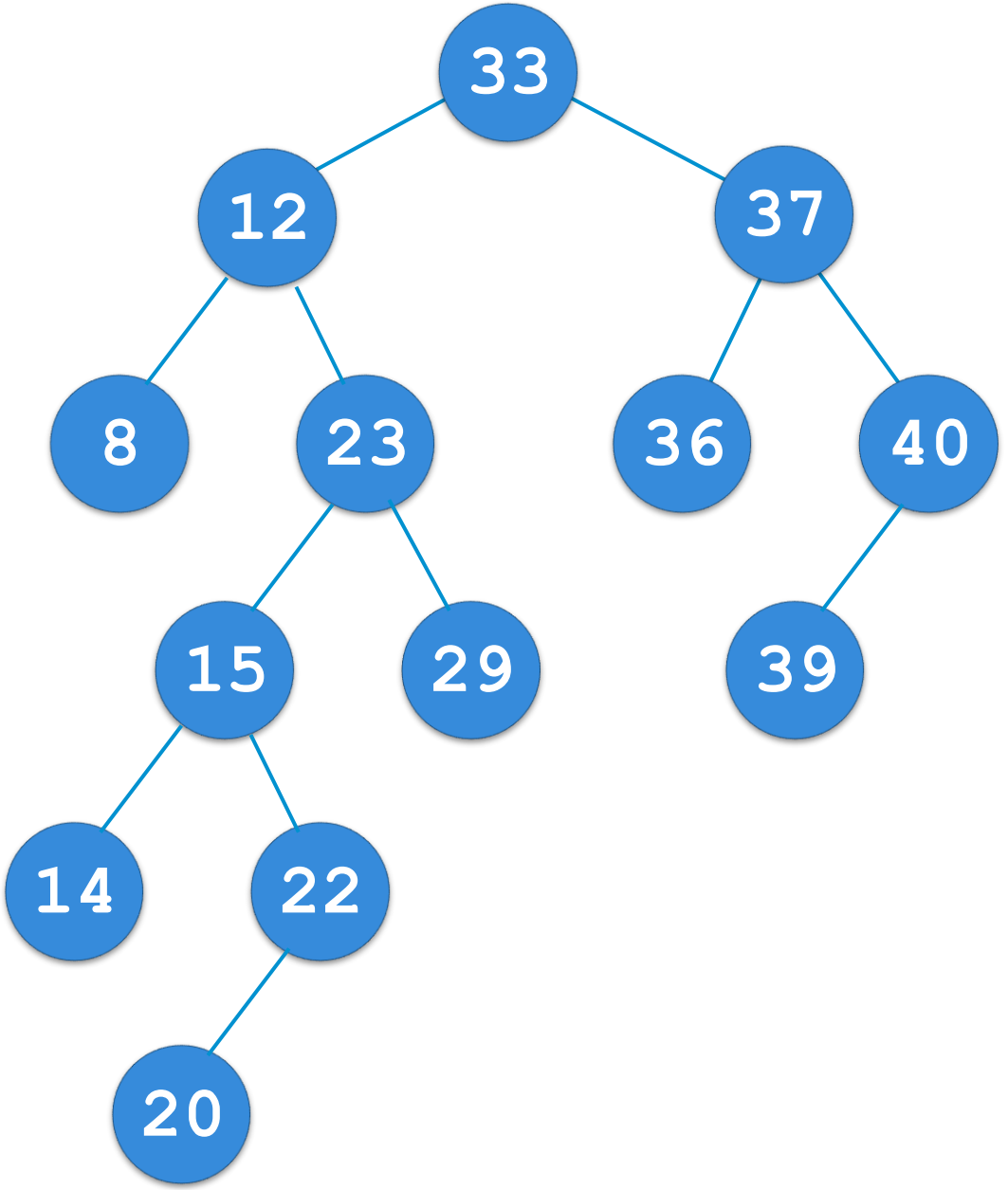
```
void parc_preordine(struct nod *root)
{
    if (root != NULL)
    {
        printf("%d ", root -> valoare);
        parc_preordine(root -> st);
        parc_preordine(root -> dr);
    }
}
```

Parcurgerea în INordine



2 3 6 7 8 12 13 22 24 41

Cautarea nodului



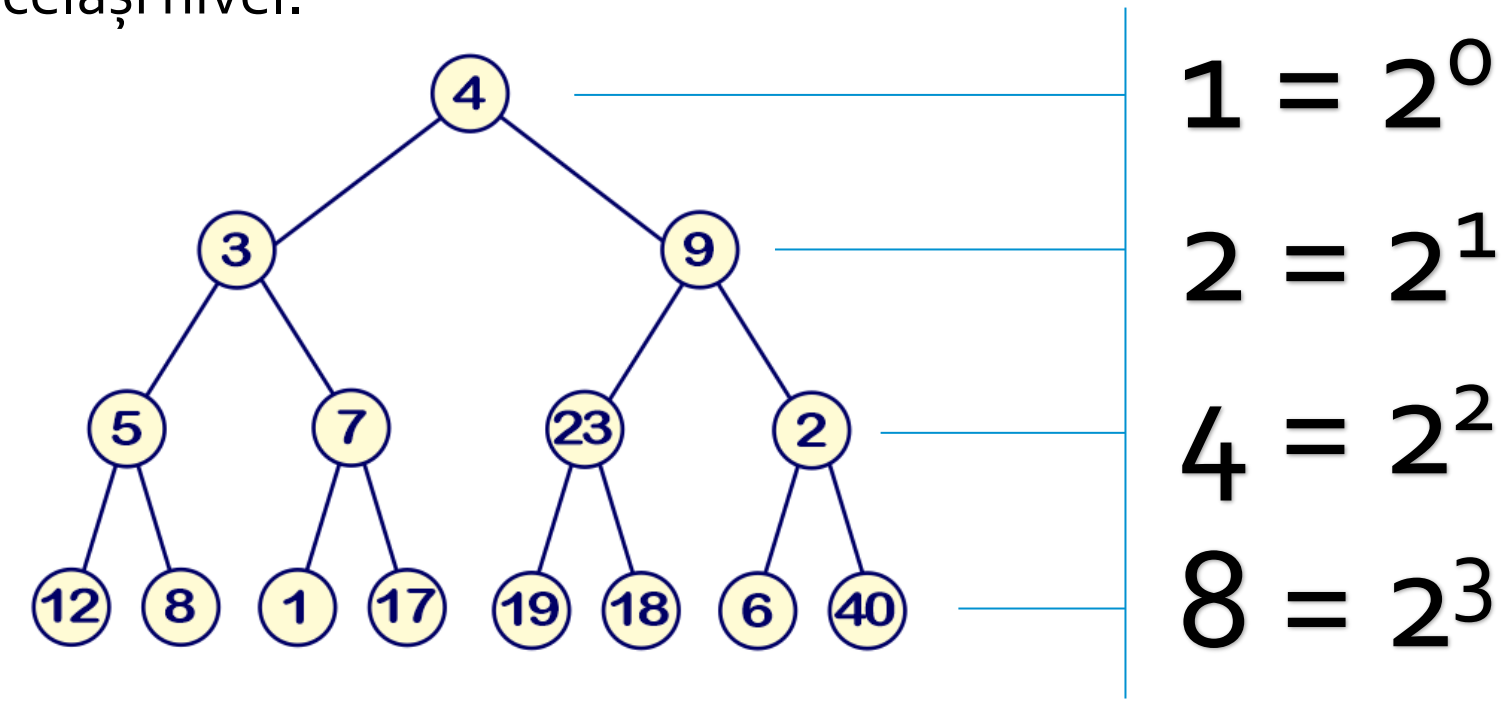
UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Complexitatea operațiilor pe ABC

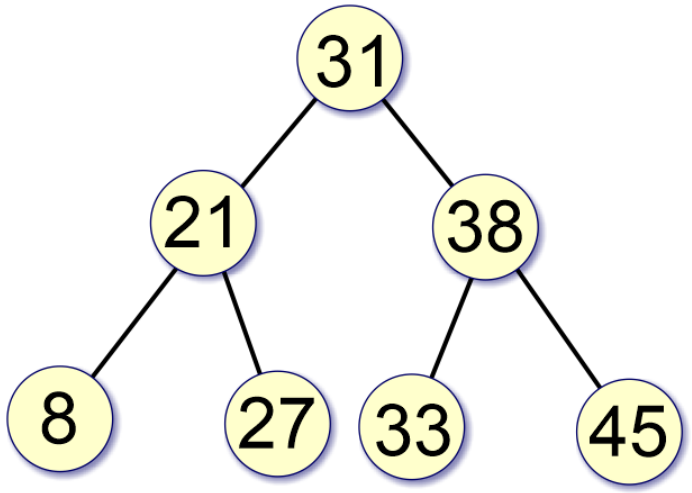
Sergiu Corlat, 2021

Definiții:

Un **arbore binar** este considerat **plin**, dacă orice nod al său are exact doi fii și toate nodurile – frunze sunt situate la același nivel.



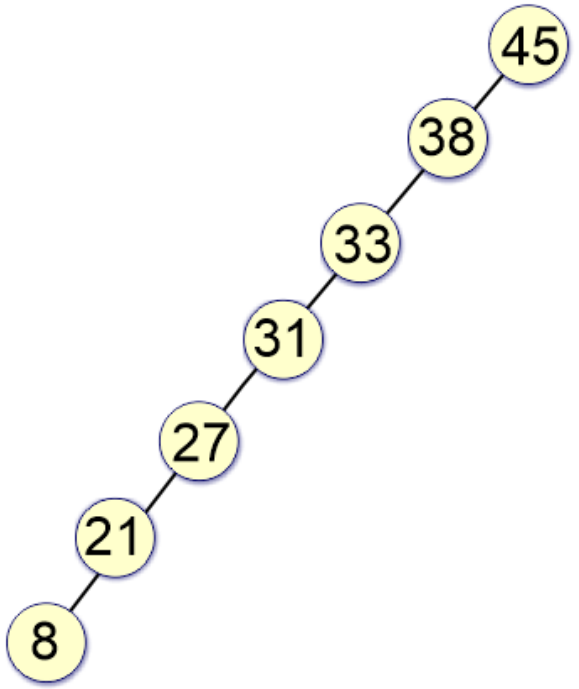
Estimări



31, 38, 33, 21, 27, 45, 8

Pentru un arbore binar cu n noduri înălțimea h poate varia de la $\log_2(n)$ – pentru un arbore de căutare balanțat până la

Estimări



45, 38, 33, 31, 27, 21, 8

Pentru un arbore binar cu n noduri înălțimea h poate varia până la **n** (în cel mai rău caz, când valorile adăugate sunt deja ordonate)

Estimări

Pentru ABC cu n noduri	Adăugarea unui nod	Excluderea unui nod	Căutarea unui nod	Parcurgerea
În cel mai bun caz	$O(log_2(n))$	$O(log_2(n))$	$O(log_2(n))$	$O(n)$
În cel mai rău caz	$O(n)$	$O(n)$	$O(n)$	$O(n)$

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Implementări!

Lucrul individual :

Modificați exemplul prezentat astfel încât la lichidare să fie folosit elementul din subarborele drept cu cea mai mică valoare

În următoarea sesiune:

- Heap-uri