Parcurgeri ale arborilor binari

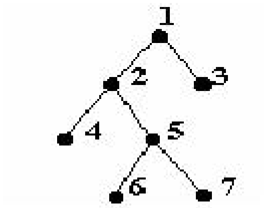


Figura 1: Arbore binar.

Parcurgere în preordine

Pentru a parcurge în *preordine* un arbore binar, se vizitează rădăcina, apoi se parcurge în preordine subarborele stâng, apoi se parcurge în preordine subarborele drept.

Spre exemplu, pentru arborele binar din Figura 1 parcurgerea în preordine este: $1, 2, 4, 5, 6, 7, 3$.

Parcurgere în inordine

Pentru a parcurge în *inordine* (*ordine simetrică*) un arbore binar, se parcurge în inordine subarborele stâng, se vizitează rădăcina, apoi se parcurge în inordine subarborele drept.

Spre exemplu, pentru arborele binar din Figura 1 parcurgerea în inordine este: $4, 2, 6, 5, 7, 1, 3$.

Parcurgere în postordine

Pentru a parcurge în *postordine* un arbore binar, se parcurge în postordine subarborele stâng, apoi se parcurge în postordine subarborele drept, după care se vizitează rădăcina.

Spre exemplu, pentru arborele binar din Figura 1 parcurgerea în postordine este: $4, 6, 7, 5, 2, 3, 1$.

Parcurgere în lăţime

Pentru a parcurge în *lăţime* un arbore binar, se vizitează nodurile pe niveluri, în ordine de la stânga la dreapta: nodurile de pe nivelul 0, apoi nodurile de pe nivelul 1, nodurile de pe nivelul 2, etc.

Spre exemplu, pentru arborele binar din Figura 1 parcurgerea în lăţime este: $1, 2, 3, 4, 5, 6, 7$.

TAD ArboreBinar
(BINARY TREE)

Observaţii

- Dăm în continuarea interfaţa minimală TAD ArboreBinar.
- Pe lângă operaţiile de mai jos, pot fi adăugate operaţii care:
  - să **şteargă** un subarbore;
  - să **modifice** informaţia utilă a rădăcinii unui subarbore;
  - să **caute** un element în arbore, etc.

TAD ArboreBinar
domeniu:

$\mathcal{AB} = \{ab \mid ab$ arbore binar cu noduri care conţin informaţii de tip $TElement\}$

operatii (interfaţa minimală):

- creeaza(ab)

  $pre:$ $adevarat$
  $post:$ $ab \in \mathcal{AB}, ab =$ arbore vid $(a = \Phi)$

- creeazaFrunza(ab, e)

  $pre:$ $e \in TElement$
  $post:$ $ab \in \mathcal{AB}, ab$ arbore având un singur nod şi informaţia din nodul rădăcină este egală cu e

- creeazaArb(ab, st, e, dr)

  $pre:$ $st, dr \in \mathcal{AB}, e \in TElement$
  $post:$ $ab \in \mathcal{AB}, ab$ arbore cu subarbore stâng $= st$, cu subarbore drept $= dr$ şi informaţia din nodul rădăcină este egală cu e

- adaugaSubStang(ab, st)

  $pre:$ $ab, st \in \mathcal{AB}$
  $post:$ $ab' \in \mathcal{AB}, ab'$ are subarborele stâng $= st$

- adaugaSubDrept(ab, dr)

  $pre:$ $ab, dr \in \mathcal{AB}$
  $post:$ $ab' \in \mathcal{AB}, ab'$ are subarborele drept $= dr$

- element(ab)

  $pre:$ $ab \in \mathcal{AB}, ab \neq \Phi$
  $post:$ $element = e, e \in TElement, e$ este informaţia din nodul rădăcină

- stang(ab)

  $pre: \quad ab \in \mathcal{AB}, ab \neq \Phi$

  $post: \quad stang = st, st \in \mathcal{AB}, st$ este subarborele stâng al lui ab

- drept(ab)

  $pre: \quad ab \in \mathcal{AB}, ab \neq \Phi$

  $post: \quad drept = dr, dr \in \mathcal{AB}, dr$ este subarborele stâng al lui ab

- vid(ab)

  $pre: \quad ab \in \mathcal{AB}$

  $$post: \quad vid \begin{cases} true & \text{dacă } ab = \Phi \\ false, & \text{altfel} \end{cases}$$

- iterator $(ab, ordine, i)$

  $pre: \quad ab \in \mathcal{AB}, ordine$ este ordinea de traversare a arborelui

  $post: \quad i \in \mathcal{I}, i$ este un iterator pe arborele ab în ordinea precizată de $ordine$

- distruge(ab) {destructor}

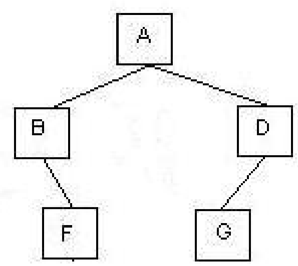
  $pre: \quad ab \in \mathcal{AB}$

  $post: \quad ab$ a fost 'distrus' (spaţiul de memorie alocat a fost eliberat)

Reprezentări posibile pentru arbori binari

A. Reprezentarea secvenţială pe tablou

- Se foloseşte ca schemă de memorare un ansamblu ($A[1..Max]$):
  - A_1 este elementul din nodul rădăcină.
  - fiul stâng al lui A_i este $A_{2\cdot i}$.
  - fiul drept al lui A_i este $A_{2\cdot i+1}$.
  - părintele lui A_i este $A_{[i/2]}$.



Indice	Tablou
[1]	A
[2]	B
[3]	D
[4]	-1
[5]	F
[6]	G
[7]	-1

B. Reprezentarea înlănţuită

Într-o astfel de reprezentare, în fiecare nod reprezentat al arborelui sunt memorate:

- informaţia utilă a nodului din arbore;
- două referinţe spre cei doi fii – stâng şi drept.

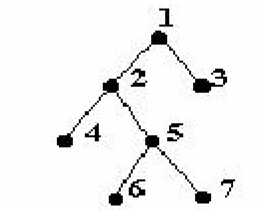
Arborele va fi identificat prin referinţa spre nodul rădăcină.

B.1 Reprezentarea înlănţuirilor folosind alocarea dinamică a memoriei.

- Referinţele sunt în acest caz pointeri (adrese de memorie).
- Pointerul NIL indică un arbore vid.

B.2. Reprezentarea înlănţuirilor folosind alocarea statică a memoriei.

Pentru arborele



reprezentarea (B.2) este

Indice	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Element	3	-	5	1	-	2	6	-	4	7
Stanga	0	8	7	6	0	9	0	5	0	0
Dreapta	0		10	1		3	0		0	0

Tabela 1: *radacina*=4, *primLiber*=2

Observaţie:

1. Capacitatea vectorilor poate fi mărită dinamic, dacă este necesar - numărul de elemente din arbore depăşeşte numărul de locaţii alocat iniţial (vezi vectorul dinamic).

Parcurgeri nerecursive ale arborilor binari

Observaţii:

1. Variantele recursive de parcurgere sunt simplu de implementat (datorită definiţiei recursive a arborelui binar).
2. Dezavantajul procedurilor recursive: supraîncărcarea stivei de execuţie.

Presupunem (în cele ce urmează) reprezentare înlănţuită cu alocare dinamică.

PREORDINE

Se va folosi o **STIVĂ** auxiliară.

`Subalgoritm preordine`(ab)
  $\{$complexitate timp: $\theta(n)\}$
pre: ab este un arbore binar
post: se afişează în preordine elementele din arbore
  $\{$stiva va conţine adrese de noduri$\}$
  `creeaza(S)`
  `Daca` $ab.rad \neq NIL$ `atunci`
    $\{$arborele nu e vid$\}$
    $adauga(S, ab.rad)$
    $\{$se adauga radacina in stiva$\}$
  `SfDaca`
  `CatTimp` $\neg vida(S)$ `executa`
    $sterge(S, p)$
    $\{$se sterge nodul din varful stiva$\}$
    $@tipareste[p].e$
    `Daca` $[p].dr \neq NIL$ `atunci`
      $\{$exista legatura dreapta$\}$
      $adauga(S, [p].dr)$
      $\{$se adauga descendentul drept in stiva$\}$
    `SfDaca`
    `Daca` $[p].st \neq NIL$ `atunci`
      $\{$exista legatura stanga$\}$
      $adauga(S, [p].st)$
      $\{$se adauga descendentul stang in stiva$\}$
    `SfDaca`
  `SfCatTimp`
`SfSubalgoritm`

LĂŢIME (parcurgere pe niveluri)

Se va folosi o **COADĂ** auxiliară.

```
Subalgoritm niveluri(ab)
    {complexitate timp: θ(n)}
pre:    ab este un arbore binar
post:   se afişează în lăţime elementele din arbore
    {coada va conţine adrese de noduri}
    creeaza(C)
    Daca ab.rad ≠ NIL  atunci
        {arborele nu e vid}
        adauga(C, ab.rad)
        {se adauga radacina in coada}
    SfDaca
    CatTimp ¬vida(C) executa
        sterge(C, p)
        {se sterge nodul din coada}
        @tipareste[p].e
        Daca [p].st ≠ NIL  atunci
            {exista legatura stanga}
            adauga(C, [p].st)
            {se adauga descendentul stang in coada}
        SfDaca
        Daca [p].dr ≠ NIL  atunci
            {exista legatura dreapta}
            adauga(C, [p].dr)
            {se adauga descendentul drept in coada}
        SfDaca
    SfCatTimp
SfSubalgoritm
```

INORDINE

Se va folosi o **STIVĂ** auxiliară.

```
Subalgoritm inordine(ab)
    {complexitate timp: θ(n)}
pre:    ab este un arbore binar
post:   se afişează în inordine elementele din arbore
    {stiva va conţine adrese de noduri}
    creeaza(S)
    p ← ab.rad
    {nodul curent}
    CatTimp ¬vida(S) ∨ (p ≠ NIL) executa
        CatTimp p ≠ NIL executa
            {se adauga in stiva ramura stanga a lui p}
            adauga(S, p)
            p ← [p].st
        SfCatTimp
        sterge(S, p)
        {se sterge nodul din varful stivei}
        @tipareste[p].e
        p ← [p].dr
```

```
    SfCatTimp
  SfSubalgoritm
```

POSTORDINE

Se va folosi o **STIVĂ** auxiliară. Un element din stivă va fi de forma $[p, k]$ unde:

- p e adresa nodului;
- k este 0 (dacă nu s-a trecut la partea dreaptă a lui p) sau 1 (s-a trecut la parcurgerea subarborelui drept al lui p).

Subalgoritm postordine(ab)
  {complexitate timp: $\theta(n)$}
pre: ab este un arbore binar
post: se afişează în postordine elementele din arbore
  creeaza(S)
  $p \leftarrow ab.rad$
  {nodul curent}
  CatTimp $\neg vida(S) \vee (p \neq NIL)$ executa
    CatTimp $p \neq NIL$ executa
      {se adauga in stiva ramura stanga a lui p}
      $adauga(S, [p, 0])$
      $p \leftarrow [p].st$
    SfCatTimp
    $sterge(S, [p, k])$
    {se sterge nodul din varful stivei}
    Daca $k = 0$ atunci
      {nu s-a traversat subarborele drept al lui p}
      $adauga(S, [p, 1])$
      $p \leftarrow [p].dr$
    altfel
      $@tipareste[p].e$
      $p \leftarrow NIL$
      {trebuie extras un nou nod din stiva - al doilea ciclu CatTimp nu trebuie sa se mai execute}
    SfDaca
  SfCatTimp
SfSubalgoritm

Probleme

1. Implementaţi iteratori cu parcurgere în preordine, postordine şi lăţime a nodurilor unui arbore binar.
2. Descrieţi în Pseudocod operaţia pentru căutarea într-un arbore binar a unei informaţii date. Se va folosi o implementare iterativă.
3. Descrieţi în Pseudocod operaţia pentru determinarea înălţimii unui arbore binar. Se va folosi o operaţie iterativă.

4. Să se scrie o procedură iterativă pentru determinarea nivelului pe care apare într-un arbore binar o informaţie dată.

5. Scrieţi o operaţie nerecursivă care determină părintele unui nod p dintr-un AB.

6. Translataţi o expresie aritmetică din forma infixată în forma prefixată/postfixată. Folosiţi un AB intermediar.

   Observaţie. O expresie aritmetică se poate reprezenta printr-un arbore binar ale cărui noduri terminale sunt asociate cu variabile sau constante şi ale cărui noduri interne sunt asociate cu unul dintre operatorii: **+**, **-**, $\times$, şi **/**. (Vezi Figura 2.)

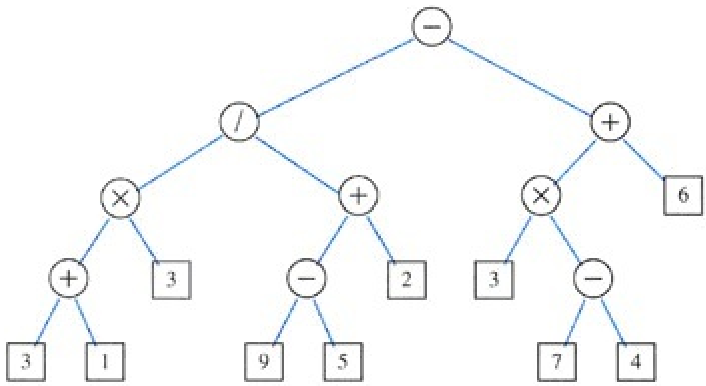


Figura 2: Arbore corespunzător expresiei $((((3+1)\times 3)/((9-5)+2))-((3\times(7-4))+6))$.

   Fiecare nod dintr-un asemenea arbore are o valoare asociată:

   - Dacă nodul este terminal valoarea sa este cea a variabilei sau constantei asociate.
   - Dacă nodul este neterminal valoarea sa este definită prin aplicarea operaţiei asociate asupra fiilor lui.

7. Evaluaţi o expresie aritmetică în forma infixată folosind un AB intermediar.

8. Cunoscand preordinea si inordinea nodurilor unui arbore binar, sa se descrie o operatie care construieste arborele. (vezi SEMINAR 7)

9. Cunoscand postordinea si inordinea nodurilor unui arbore binar, sa se descrie o operatie care construieste arborele. (vezi SEMINAR 7)