Lecture 10/14/16

Lecturer: Xiaodi Wu

Reading: Chapter 2.3
Traversals of Trees

A traversal of a tree $T$ is a systematical way of “visiting” all nodes in $T$. 

- **Pre-order**: Root first and then visit each sub-tree in order.
- **Post-order**: Visit each sub-tree first and then the root.
A **traversal** of a tree \( T \) is a systematical way of "visiting" all nodes in \( T \).

- **Pre-order:** Root first and then visit each sub-tree in order.
- **Post-order:** Visit each sub-tree first and then the root.
Traversals of Trees

**Algorithm preorder(} T, v)\
"visit" the node v\
for each child w of v do\
    preorder(} T, w)\
end for

**Algorithm postorder(} T, v)\
for each child w of v do\
    post-order(} T, w)\
end for\
"visit" the node v

Complexity $O(n)$: similar counting as the analysis in height().
Traversal of Trees

**Algorithm preorder**($T, v$)
"visit" the node $v$
for each child $w$ of $v$ do
    preorder($T, w$)
end for

**Algorithm postorder**($T, v$)
for each child $w$ of $v$ do
    post-order($T, w$)
end for
"visit" the node $v$

**Complexity**

$O(n)$: similar counting as the analysis in height().
Trees: Pre-order

Pre-order: 33, 15, 10, 5, 12, 20, 18, 47, 38, 36, 39, 51, 49
Pre-order: 33, 15, 10, 5, 12, 20, 18, 47, 38, 36, 39, 51, 49
Trees: Post-order

Post-order: 5, 12, 10, 18, 20, 15, 36, 39, 38, 49, 51, 47, 33
Post-order: 5, 12, 10, 18, 20, 15, 36, 39, 38, 49, 51, 47, 33
A binary tree is an ordered tree in which each node has at most two children. It is called proper if each internal node has two children (left and right child).
A binary tree is an ordered tree in which each node has at most two children. It is called proper if each internal node has two children (left and right child).

Methods

- leftChild(v): return the left child of v if v is internal.
- rightChild(v): return the right child of v if v is internal.
A **binary tree** is an ordered tree in which each node has at most two children. It is called **proper** if each internal node has two children (**left** and **right child**).

**Methods**

- `leftChild(v)`: return the left child of `v` if `v` is internal.
- `rightChild(v)`: return the right child of `v` if `v` is internal.

A third traversal order: **inorder**.
Traversals of Binary Trees

**Algorithm** \( b\text{Preorder}(T, v) \)

"visit" the node \( v \)

if \( v \) is internal

\[ b\text{Preorder}(T, T\text{.leftChild}(v)) \]

\[ b\text{Preorder}(T, T\text{.rightChild}(v)) \]

end if

**Algorithm** \( b\text{Postorder}(T, v) \)

if \( v \) is internal

\[ b\text{Postorder}(T, T\text{.leftChild}(v)) \]

\[ b\text{Postorder}(T, T\text{.rightChild}(v)) \]

end if

"visit" the node \( v \)
**Algorithm** bInorder\((T, v)\)

**if** \(v\) is internal **then**

\[
bPostorder(T, T.leftChild(v))
\]

**end if**

"visit" the node \(v\)

**if** \(v\) is internal **then**

\[
bPostorder(T, T.rightChild(v))
\]

**end if**
Trees: In-order

In-order: 5, 10, 12, 15, 18, 20, 33, 36, 38, 39, 47, 49, 51
Identify Trees from preorder, inorder, postorder visits

- Pre-order: 33, 15, 10, 5, 12, 20, 18, 47, 38, 36, 39, 51, 49
- Post-order: 5, 12, 10, 18, 20, 15, 36, 39, 38, 49, 51, 47, 33
- In-order: 5, 10, 12, 15, 18, 20, 33, 36, 38, 39, 47, 49, 51
Identify Trees from preorder, inorder, postorder visits

- Pre-order: 33, 15, 10, 5, 12, 20, 18, 47, 38, 36, 39, 51, 49
- Post-order: 5, 12, 10, 18, 20, 15, 36, 39, 38, 49, 51, 47, 33
- In-order: 5, 10, 12, 15, 18, 20, 33, 36, 38, 39, 47, 49, 51

How?

- Use Pre-order or Post-order to identify the root.
Identify Trees from preorder, inorder, postorder visits

- Pre-order: 33, 15, 10, 5, 12, 20, 18, 47, 38, 36, 39, 51, 49
- Post-order: 5, 12, 10, 18, 20, 15, 36, 39, 38, 49, 51, 47, 33
- In-order: 5, 10, 12, 15, 18, 20, 33, 36, 38, 39, 47, 49, 51

How?

- Use Pre-order or Post-order to identify the root.
- Use In-order to identify both sub-trees.
Identify Trees from preorder, inorder, postorder visits

- Pre-order: 33, 15, 10, 5, 12, 20, 18, 47, 38, 36, 39, 51, 49
- Post-order: 5, 12, 10, 18, 20, 15, 36, 39, 38, 49, 51, 47, 33
- In-order: 5, 10, 12, 15, 18, 20, 33, 36, 38, 39, 47, 49, 51

How?

- Use Pre-order or Post-order to identify the root.
- Use In-order to identify both sub-trees.
- Apply the above procedure recursively.
Arithmetic Expression

\[
- \left( \frac{(3 + 1) \times 3}{(9 - 5) + 2} \right) - (3 \times (7 - 4)) + 6
\]
((((3 + 1) \times 3)/((9 - 5) + 2)) - ((3 \times (7 - 4)) + 6))
Arithmetic Expression: Preorder

```
- 
/ 
× +
+ × -
+ × -
+ 
+ -
```

```
3 1 9 5 2 3 7 4 6
```
Arithmetic Expression: Preorder

\[- \left( \frac{\times (3) (1)}{3} \right) + (-(9) (5) (2)) + (\times (3) (-7) (4)) (6) \]
Arithmetic Expression: Postorder

- / × + 3 1
  / + × + 6
  + 3 9 − 5
  + 7 4
  + 3 1

- / × + 3 1
  / + × + 6
  + 3 9 − 5
  + 7 4
  + 3 1
Arithmetic Expression: Postorder

\(((3)(1+)(3\times)\((((9)(5)-(2)+)/)\)((3)((7)(4)-(\times))(6)+)-)\)
Arithmetic Expression: Inorder

- 
  / 
  / 
  \ 
  /

+ 
+ 
-

6

3
+

1

9

5
-

2

3
-

7

4
Arithmetic Expression: Inorder

$$(((3 + 1) \times 3)/((9 - 5) + 2)) - ((3 \times (7 - 4)) + 6)$$
Properties about Binary Trees

**Theorem**

Let $T$ be a (proper) binary tree with $n$ nodes, $h$ the height of $T$. We have

- The number of external nodes of $T$ is between $h + 1$ and $2^h$.
- The number of internal nodes of $T$ is between $h$ and $2^h - 1$.
- The height of $T$ is between $\log(n + 1) - 1$ and $(n - 1)/2$. 
Properties about Binary Trees

Theorem (Theorem 2.9, on page 85)

In a (proper) binary tree $T$, the number of external nodes is 1 more than the number of internal nodes.

Proof.

By induction,

- If $T$ only has one node, it must be external. Thus, no internal node. The statement holds.

- Otherwise, $T$ has at least one external node with its parent. Remove any external node $w$ and its parent $v$, then connect $w$'s sibling to $v$'s parent. The tree remains proper and binary, but smaller.
Properties about Binary Trees

Theorem (Theorem 2.9, on page 85)

In a (proper) binary tree $T$, the number of external nodes is 1 more than the number of internal nodes.

Proof.

By induction,

- If $T$ only has one node, it must be external. Thus, no internal node. The statement holds.
Properties about Binary Trees

Theorem (Theorem 2.9, on page 85)

In a (proper) binary tree $T$, the number of external nodes is 1 more than the number of internal nodes.

Proof.

By induction,

- If $T$ only has one node, it must be external. Thus, no internal node. The statement holds.

- Otherwise, $T$ has at least one external node with its parent. Remove any external node $w$ and its parent $v$, then connect $w$’s sibling to $v$’s parent. The tree remains proper and binary, but smaller.