CSE 373 Lecture 6: Trees

✦ Today's agenda:
   ➤ Trees: Definition and terminology
   ➤ Traversing trees
   ➤ Binary search trees
   ➤ Inserting into and deleting from trees
✦ Covered in Chapter 4 of the text

Why Do We Need Trees?

✦ Lists, Stacks, and Queues represent linear sequences
✦ Data often contain hierarchical relationships that cannot be expressed as a linear ordering
   ➤ File directories or folders on your computer
   ➤ Moves in a game
   ➤ Employee hierarchies in organizations and companies
   ➤ Family trees
   ➤ Classification hierarchies (e.g. phylum, family, genus, species)

Tree Jargon

✦ Basic terminology:
   • nodes and edges
   • root
   • subtrees
   • parent
   • children, siblings
   • leaves
   • path
   • ancestors
   • descendants
   • path length

Note: Arrows denote directed edges
Trees always contain directed edges but arrows are often omitted.

More Tree Jargon

✦ Length of a path = number of edges
✦ Depth of a node N = length of path from root to N
✦ Height of node N = length of longest path from N to a leaf
✦ Depth and height of tree = ?
Definition and Tree Trivia

- Recursive Definition of a Tree:
  A tree is a set of nodes that is
  a. an empty set of nodes, or
  b. has one node called the root from which zero or more trees
     (subtrees) descend.
- A tree with N nodes always has ___ edges
- Two nodes in a tree have at most how many paths between
  them?
- Can a non-zero path from node N reach node N again?
- Does depth of nodes in a non-zero path increase or decrease?

Definition and Tree Trivia

- Recursive Definition of a Tree:
  A tree is a set of nodes that is
  a. an empty set of nodes, or
  b. has one node called the root from which zero or more trees
     (subtrees) descend.
- A tree with N nodes always has N-1 edges
- Two nodes in a tree have at most one path between them
- Can a non-zero path from node N reach node N again?
  ✗ No! Trees can never have cycles.
- Does depth of nodes in a non-zero path increase or decrease?
  ✗ Depth always increases in a non-zero path

Implementation of Trees

- Obvious Pointer-Based Implementation: Node with value
  and pointers to children
  ➤ Problem: Do not usually know number of children for a node in
  advance. How many pointers should we allocate space for?
- Better Implementation: 1st Child/Next Sibling Representation
  ➤ Each node has 2 pointers: one to its first child and one to next sibling
  ➤ Can handle arbitrary number of children
  ➤ Exercise: Draw the representation
    for this tree…

Application: Arithmetic Expression Trees

Example Arithmetic Expression:
A + (B * (C / D ))

How would you express this as a tree?

Example Arithmetic Expression:
A + (B * (C / D ))

How would you express this as a tree?
Application: Arithmetic Expression Trees

Example Arithmetic Expression:

\[ A + (B * (C / D)) \]

Tree for the above expression:

- Used in most compilers
- No parenthesis need – use tree structure
- Can speed up calculations e.g. replace \( / \) node with \( C / D \) if \( C \) and \( D \) are known
- Calculate by traversing tree (how?)

Traversing Trees

- Preorder: Root, then Children
  \( + A * B / C D \)
- Postorder: Children, then Root
  \( A B C D / + \)
- Inorder: Left child, Root, Right child
  \( A + B * C / D \)

Example Code for Recursive Preorder

```c
void print_preorder ( TreeNode T )
{
    if ( T == NULL ) return;
    else {
        print_element(T->Element);
        P = T->FirstChild;
        while (P != NULL) {
            print_preorder ( P );
            P = P->NextSibling;
        }
    }
}
```

What is the running time for a tree with \( N \) nodes?

Preorder Traversal with a Stack

```c
void Stack_Preorder (TreeNode T, Stack S)
{
    if (T == NULL) return; else push(T,S);
    while (!isempty(S)) {
        T = pop(S);
        print_element(T->Element);
        if (T->Right != NULL) push(T->Right, S);
        if (T->Left != NULL) push(T->Left, S);
    }
}
```

What is the running time for a tree with \( N \) nodes?
Binary Trees
✦ Every node has at most two children
  ★ Most popular tree in computer science
✦ Given N nodes, what is the minimum depth of a binary tree?
✦ What is the maximum depth of a binary tree?

Given N nodes, what is the minimum depth of a binary tree?
★ At depth d, you can have N = 2^d to 2^{d+1} - 1 nodes (a full tree)
★ So, minimum depth d is: \log N \leq d \leq \log(N+1) - 1 or \Theta(\log N)

What is the maximum depth of a binary tree?
★ Degenerate case: Tree is a linked list!
★ Maximum depth = N-1
★ Goal: Would like to keep depth at around \log N to get better performance than linked list for operations like Find.

Binary Search Trees
✦ Binary search trees are binary trees in which the value in every node is:
  ★ all values in the node’s left subtree
  ★ all values in the node’s right subtree
✦ Application: “Look-up” table
  ★ Example: Given SSN, return student record
  ★ SSN stored in each node as the key value
✦ Operations:
  ★ Find, FindMin, FindMax
  ★ Insert, Delete

Operations on Binary Search Trees
✦ How would you implement these?
  ★ Recursive definition of binary search trees allows recursive routines!
✦ Position FindMin(Tree T)
✦ Position FindMax(Tree T)
✦ Position Find(ElementType X, Tree T)
✦ Tree Insert(ElementType X, Tree T)
✦ Tree Delete(ElementType X, Tree T)
Insert Operation

- Tree Insert(ElementType X, Tree T)
  - Do a “Find” operation for X
  - If X is found, update duplicates counter
  - Else, “Find” stops at a NULL pointer
  - Insert Node with X there!

- Example: Insert 95

Delete Operation

- Delete is a bit trickier…Why?
- Suppose you want to delete 10
- Strategy:
  - Find 10
  - Delete the node containing 10
- Problem: When you delete a node, what do you replace it by?

- Problem: When you delete a node, what do you replace it by?
- Solution:
  1. If it has no children, by NULL
  2. If it has 1 child, by that child
  3. If it has 2 children, by the node with the smallest value in its right subtree
- Examples:
  1. Delete 5
  2. Delete 24 (note: recursive deletion)
  3. Delete 10 (note: recursive deletion)
Example: Delete “10”

Find 10. Replace with smallest value in right subtree

Delete smallest value in right subtree

Example: Delete “10”

Delete “11” in right subtree (recursive delete)

Find “11” 1 child, so replace by child

Delete “17” No child, so replace by NULL

Next Class:
Analysis of Binary Search Tree Operations
Other Species of Trees: AVL, splay, and B-trees
Homework #2 will be assigned

To Do:
Read Chapter 4