CSCI 262
Data Structures
16 – Binary Trees

Trees
A (rooted) tree is defined recursively:

Trees

Tree Terminology

More Tree Terminology

More Tree Terminology

Binary Trees
A binary tree is defined recursively:
Binary Trees

Height of a Binary Tree

What is the min height?
What is the max height?

Minimum Height of a Binary Tree

If we pack the maximum number of nodes into a binary tree of height $k$, then we have*

$$1 + 2 + 4 + \ldots + 2^k = 2^{k+1} - 1$$

nodes, which means...

*This is sometimes called a full tree.

Minimum Height of a Binary Tree

... the minimum height of a binary tree with $n$ nodes is $O(\log_2 n)$.

Implementing the Binary Tree

Just follow the recursive definition to get a simple implementation:

```cpp
template <class T>
class binary_tree_node {
public:
    T data;
    binary_tree_node<T>* left;
    binary_tree_node<T>* right;
};
```

Implementing the Binary Tree

- For now, we’ll just implement a tree as nodes
- Tree functions will be free functions
- Can also encapsulate specific kinds of binary trees as classes/class templates
Binary Tree Traversals

- A traversal of a tree is the act of visiting every node in the tree once.
- There are three traversal orders:
  - Pre-order
  - In-order
  - Post-order

Pre-Order Traversal

Visit the root first, then the left and right sub-trees recursively:

```
1
/   \
2     7
/ \
3   4 8
/   \
5   6
```

The numbers give the order of the visited nodes.

In-Order Traversal

Visit the left sub-tree, the root, and then the right sub-tree:

```
6
/   \
2     8
/ \
1   4 7
/   \
3   5
```

The numbers give the order of the visited nodes.

Post-Order Traversal

Visit the left and right sub-trees first and the root last:

```
8
/   \
5     7
/ \
1   4 6
/   \
2   3
```

The numbers give the order of the visited nodes.

Pre-Order Traversal Implementation

Note naturally recursive description: visit the root first, then the left and right sub-trees.

So we get a naturally recursive implementation:

```
template <class T>
void do_preorder(binary_tree_node<T>* root) {
    if (root != NULL) {
        // do something with root->data
        do_preorder(root->left);
        do_preorder(root->right);
    }
}
```

Other Implementations

Can you write the in-order and post-order traversal code?
Traversal Applications

- Print all nodes (in a particular order):
  ```cpp
template <class T>
void print_preorder(binary_tree_node<T>* root) {
    if (root != NULL) {
        cout << root->data << " ";
        print_preorder(root->left);
        print_preorder(root->right);
    }
}
```

- Count nodes:
  ```cpp
template <class T>
int count(binary_tree_node<T>* root) {
    if (root == NULL) return 0;
    return 1 + count(root->left) + count(root->right);
}
```

Tree Applications

- Decision trees
  - A kind of structure used in AI
  - See project 4 – Animal (20 Questions)
- Sets/Maps
  - Using Binary Search Trees (next lecture)
  - Compression/encoding (Huffman encoding)
  - Organizing high-dimensional spaces (k-d trees)
  - Spelling dictionary (Tries)
  - Many more...

Up Next

- Wednesday, November 7
  - Binary search trees
  - Reading: Chapter 16.3-16.5
- Friday, November 9
  - Lab 10, continued
  - APT 4 due
- Monday, November 12
  - Midterm review
- Wednesday, November 14
  - Midterm 2 (in class)