CSC 212
Data Structures and Abstractions
Fall 2015

Lecture 08: Traversals
Tree Traversals

Visiting each node in the tree exactly once

Types of Traversals
- preorder
- inorder
- postorder
Preorder Traversal

If the tree is not empty

**Visit** the root

Recursively traverse **left** subtree

Recursively traverse **right** subtree
void BSTree::preorder(BSTNode *p) {
    if (p) {
        std::cout << p->data << " ";
        preorder(p->left);
        preorder(p->right);
    }
}
Inorder Traversal

If the tree is not empty

Recursively traverse left subtree

Visit the root

Recursively traverse right subtree
void BSTree::inorder(BSTNode *p) {
    if (p) {
        inorder(p->left);
        std::cout << p->data << " ";
        inorder(p->right);
    }
}
If the tree is not empty

Recursively traverse left subtree
Recursively traverse right subtree
Visit the root
void BSTree::postorder(BSTNode *p) {
    if (p) {
        postorder(p->left);
        postorder(p->right);
        std::cout << p->data << " ";
    }
}
How to destroy a binary tree?
Tree Shape

 Depends on order of insertion
N = 255
max = 16
avg = 9.1
opt = 7.0
Implications

Cost of basic operations — per item
(search, insert, remove)

worst-case

average-case
keys inserted in random order
Implications

Cost of basic operations — per item
(search, insert, remove)

worst-case \quad O(h) = O(n)

average-case
keys inserted in random order
Implications

Cost of basic operations — per item
(search, insert, remove)

worst-case \[ O(h) = O(n) \]
average-case \[ O(h) = O(\log n) \]

keys inserted in random order
Other Operations

Minimum() — find smallest value
Maximum() — find largest value
Floor(k) — find largest value \( \leq \) than \( k \)
Ceiling(k) — find smallest value \( \geq \) than \( k \)
Other Operations

Minimum() — find smallest value \( O(h) \)

Maximum() — find largest value

Floor\( (k) \) — find largest value \( \leq \) than \( k \)

Ceiling\( (k) \) — find smallest value \( \geq \) than \( k \)
Other Operations

Minimum() — find smallest value

Maximum() — find largest value

Floor(k) — find largest value $\leq$ than $k$

Ceiling(k) — find smallest value $\geq$ than $k$
Other Operations

Minimum() — find smallest value \( O(h) \)

Maximum() — find largest value \( O(h) \)

Floor(k) — find largest value \( \leq \) than k \( O(h) \)

Ceiling(k) — find smallest value \( \geq \) than k
Other Operations

Minimum() — find smallest value \( O(h) \)

Maximum() — find largest value \( O(h) \)

Floor(k) — find largest value \( \leq \) than k \( O(h) \)

Ceiling(k) — find smallest value \( \geq \) than k \( O(h) \)
Can we sort using BSTs?

Given $n$ numbers …

what is the cost of a bad case?

what is the cost of a best case?
Can we sort using BSTs?

Given $n$ numbers …

what is the cost of a bad case? $O(n^2)$

what is the cost of a best case?
Can we sort using BSTs?

Given \( n \) numbers …

what is the cost of a bad case? \( \mathcal{O}(n^2) \)

what is the cost of a best case? \( \mathcal{O}(n \log n) \)