CSC 212: Data Structures and Abstractions

16: Binary Search Trees II

Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

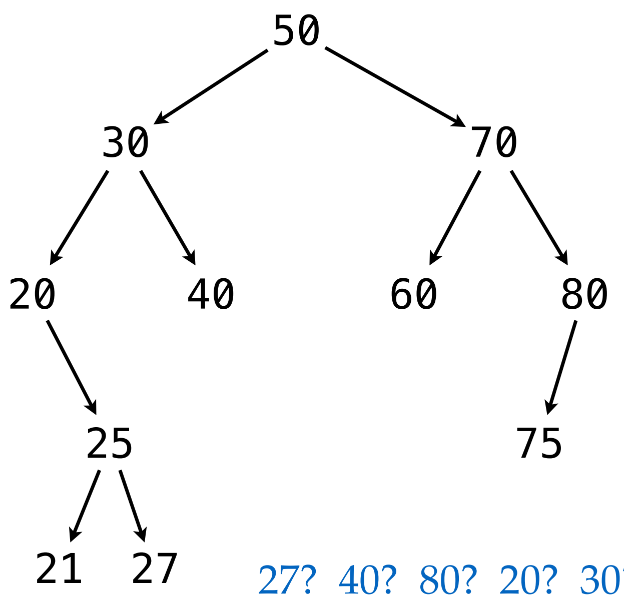
Fall 2017

THINK BIG WE DO℠

Remove

- **Case 1: node is a leaf**
  - trivial, delete node and set parent's pointer to NULL
- **Case 2: node has 1 child**
  - trivial, set parent's pointer to the only child and delete node
- **Case 3: node has 2 children**
  - find **successor**
  - copy successor's data to node
  - delete successor

can also use predecessor



27? 40? 80? 20? 30? 50?

Traversals

- Preorder traversal
- Inorder traversal
- Postorder traversal

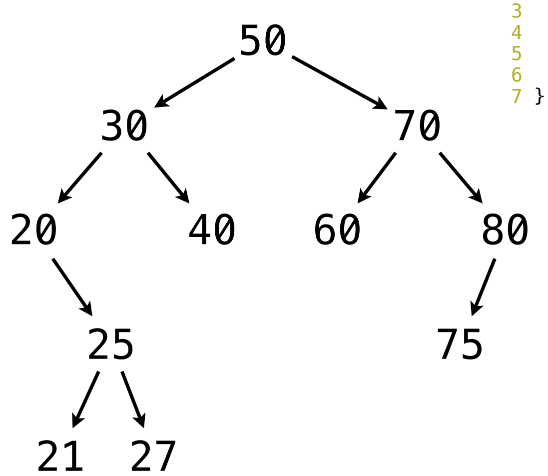
O(n)

Remember!

```
algorithm preorder(p) {
    visit(p)
    for each child c of p {
        preorder(c)
    }
}
```

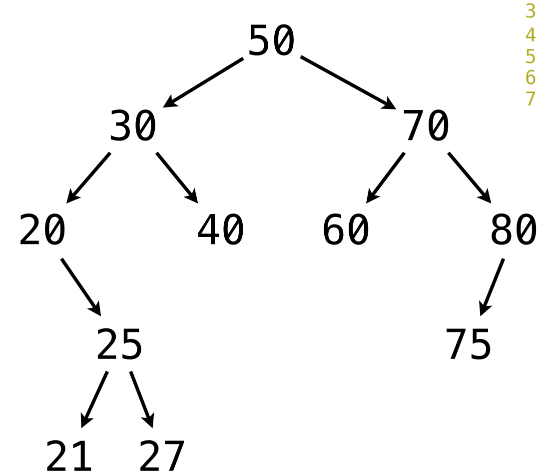
```
algorithm postorder(p) {
    for each child c of p {
        postorder(c)
    }
    visit(p)
}
```

Preorder traversal?



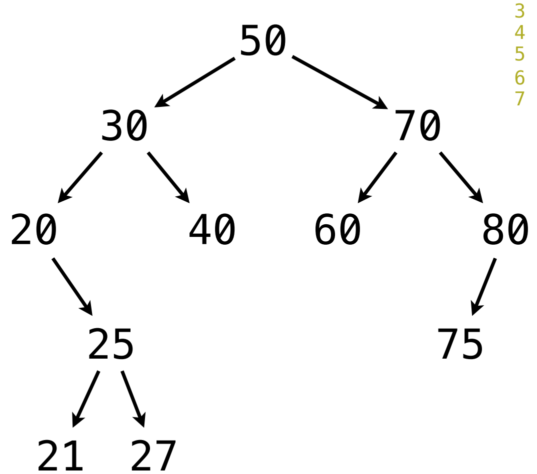

```
algorithm preorder(p) {
    if (p) {
        visit(p)
        preorder(p->left)
        preorder(p->right)
    }
}
```


Inorder traversal?




```
algorithm inorder(p) {
    if (p) {
        inorder(p->left)
        visit(p)
        inorder(p->right)
    }
}
```


Postorder traversal?




```
algorithm postorder(p) {
    if (p) {
        postorder(p->left)
        postorder(p->right)
        visit(p)
    }
}
```


How to destroy a binary tree?

How to print all elements in increasing order?

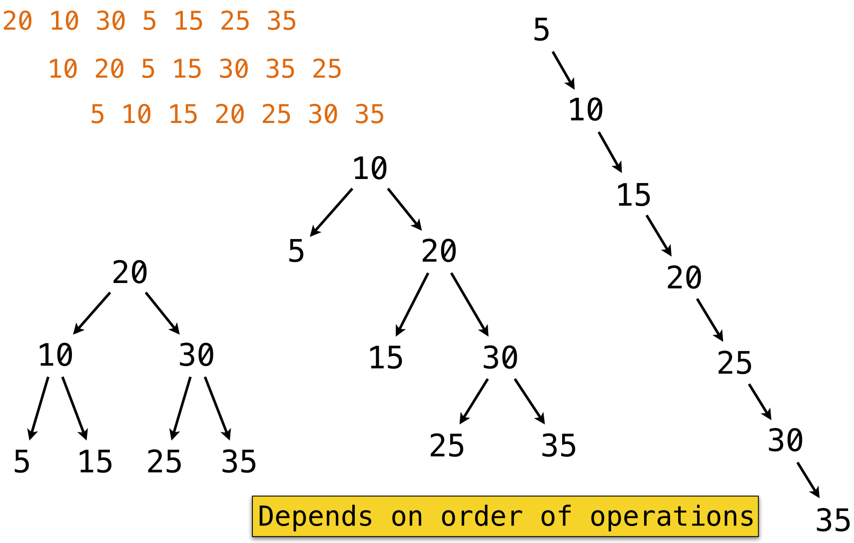
Analysis

Tree Shape?

20 10 30 5 15 25 35

10 20 5 15 30 35 25

5 10 15 20 25 30 35



Depends on order of operations

Implications

Cost of basic Operations

- ✓Search
- ✓Insert
- ✓Remove

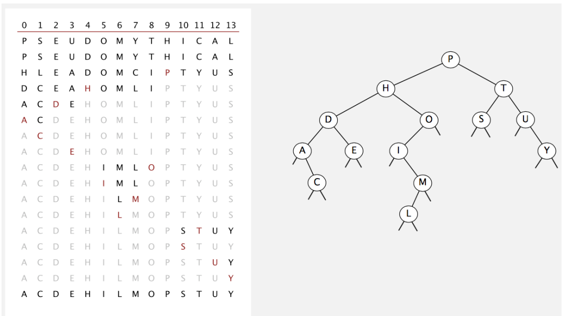
Worst-case?

Best-case?

Average-case?

Average-case analysis

- If **n distinct keys** are inserted into a BST in random order, expected number of compares for basic operations is **~2 ln n ~= 1.39 log n**
  - **proof**: 1-1 correspondence with quick-sort partitioning



h = O(log n)

https://algs4.cs.princeton.edu/lectures/32BinarySearchTrees.pdf

The Height of a Random Binary Search Tree

Expected height: ~4.311 ln n ~= 2.99 log n

BRUCE REED

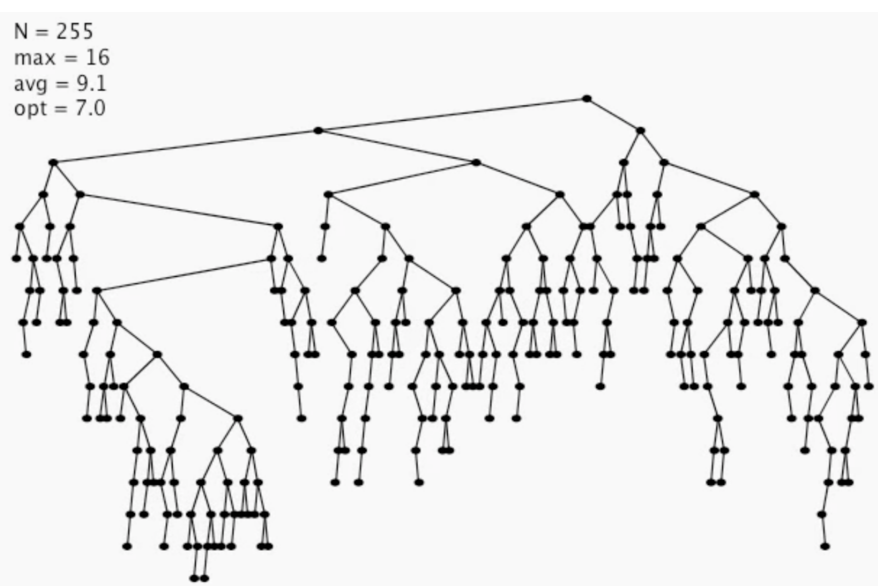
McGill University, Montreal Quebec, Canada and CNRS, Paris, France

Abstract. Let H_n be the height of a random binary search tree on n nodes. We show that there exist constants $\alpha = 4.311\cdots$ and $\beta = 1.953\cdots$ such that $\boxed{\mathbf{E}(H_n) = \alpha \ln n - \beta \ln \ln n + O(1)}$. We also show that $\mathbf{Var}(H_n) = O(1)$.

Categories and Subject Descriptors: E.1 [**Data Structures**]: *trees*; G.2 [**Discrete Mathematics**]; G.3 [**Probability and Statistics**]

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Binary search tree, height, probabilistic analysis, random tree, asymptotics, second moment method



https://algs4.cs.princeton.edu/lectures/32BinarySearchTrees.pdf

Collections/Dictionaries as arrays

	What?	Sequential (unordered)	Binary Search (ordered)	BST
search	search for a key	O(n)	O(log n)	O(h)
insert	insert a key	O(n)	O(n)	O(h)
delete	delete a key	O(n)	O(n)	O(h)
min/max	smallest/largest key	O(n)	O(1)	O(h)
floor/ceiling	predecessor/successor	O(n)	O(log n)	O(h)
rank	number of keys less than key	O(n)	O(log n)	O(h)**

(**) requires the use of 'size' at every node

Can we create a sorting algorithm using BSTs?

Worst-case?

Best-case?

Average-case?