CSE 326: Data Structures
Lecture #7
Dendrology

Bart Niswonger

Summer Quarter 2001

Today's Outline

- Correction & Clarification
- Basic Tree Data Structure
- Dictionary & Search ADTs
- Binary Search Trees

Clarification

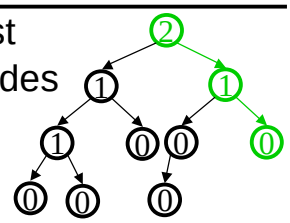
height & *depth* Defined

- Length of a ***path*** equals number of ***edges*** on the path
- *height(n)* : length of the **longest** path from *n* to a leaf
- *depth(n) :* length of path from root to *n*
- *height* of a tree equals *height(root)*

Correction

Right Path in a Leftist Tree is Short

- If the right path has length at least **r**, the tree has at least $2^{r+1}-1$ nodes



- Proof by induction

  Basis: $r = 1$. Tree has at least three nodes: $2^{1+1} - 1 = 3$

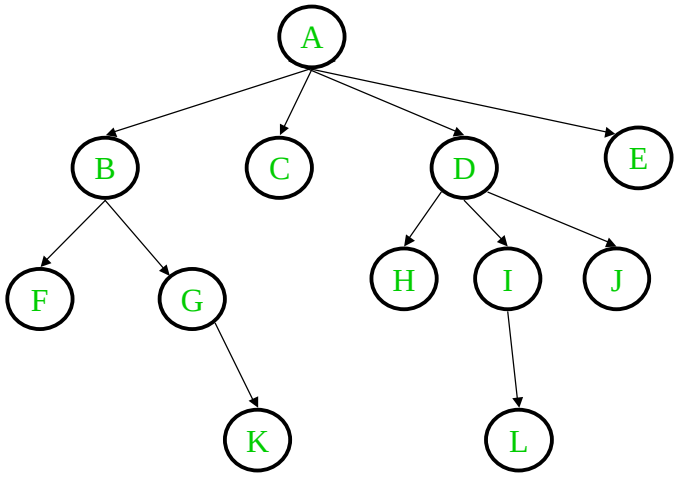
  Inductive step: assume true for $r' < r$. The right subtree has a right path of length at least $r - 1$, so it has at least $2^r - 1$ nodes. The left subtree must also have a right path of length at least $r - 1$ (otherwise $2^r - 1$ [illegible]

  $$(2^r - 1) + (2^r - 1) + 1 = 2^{r+1} - 1$$

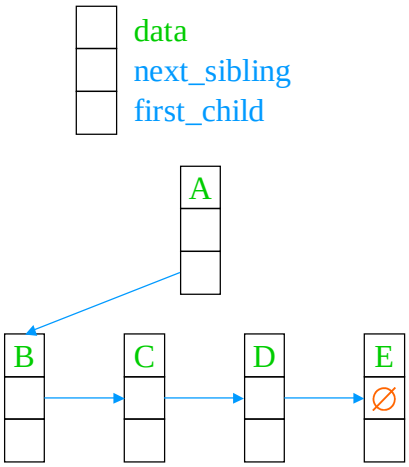
- [illegible]

  $$\log(n + 1) \sim \log n$$

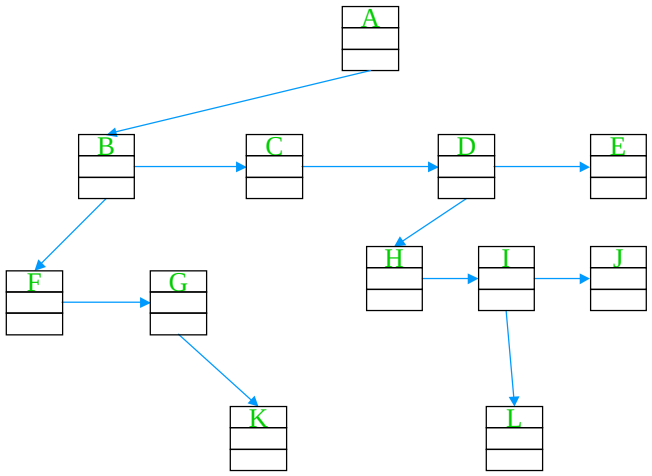
A Generic Tree



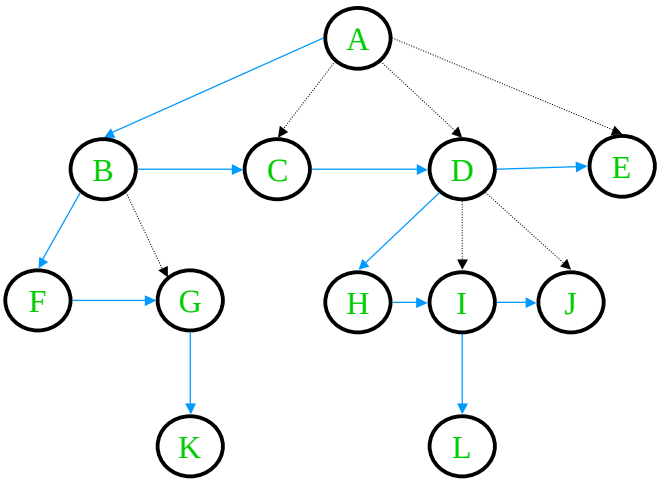
A Generic Tree Data Structure



A Generic Tree in A Generic Tree Data Structure



Combined View of Tree



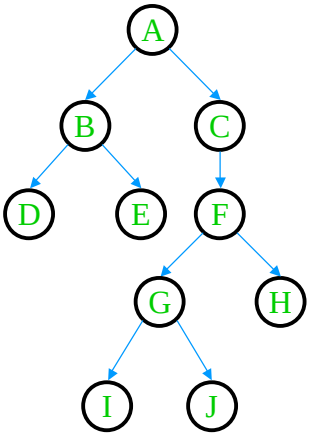
Traversals

- Many algorithms involve walking through a tree, and performing some computation at each node
- Walking through a tree is called a traversal
- Common kinds of traversal
  - Pre-order
  - Post-order
  - Level-order

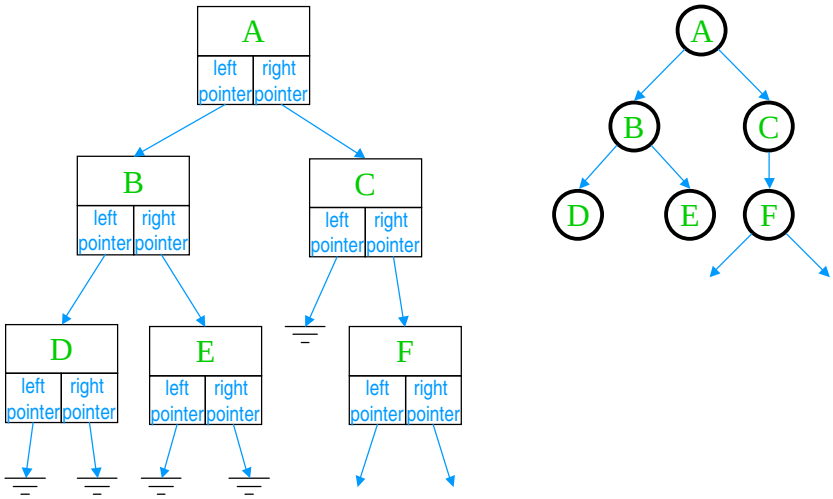
Binary Trees

- Binary tree is
  - a root
  - left subtree *(maybe empty)*
  - right subtree *(maybe empty)*
- Properties
  - max # of leaves:
  - max # of nodes:
  - average height for N nodes:
- Representation:

Data	
left pointer	right pointer

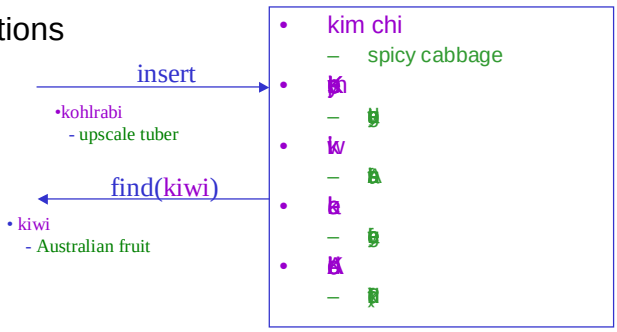


Representation



Dictionary ADT

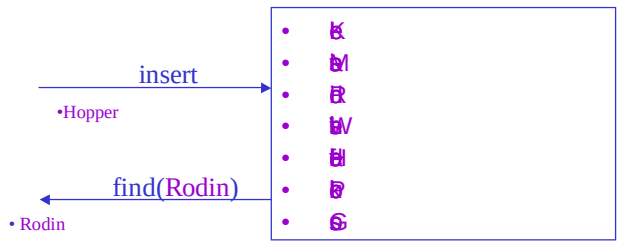
- Dictionary operations
  - create
  - destroy
  - insert
  - find
  - delete



- Stores *values* associated with user-specified *keys*
  - values may be any (homogenous) type
  - keys may be any (homogenous) comparable type

Search ADT

- [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]



- [illegible]
  - [illegible]
  - [illegible]

A Modest Few Uses

- Arrays
- Sets
- Dictionaries
- Router tables
- Page tables
- Symbol tables
- C++ Structures

Naïve Implementations

insert find delete

- Linked list
- Unsorted array
- Sorted array

so close!

Naïve Implementations

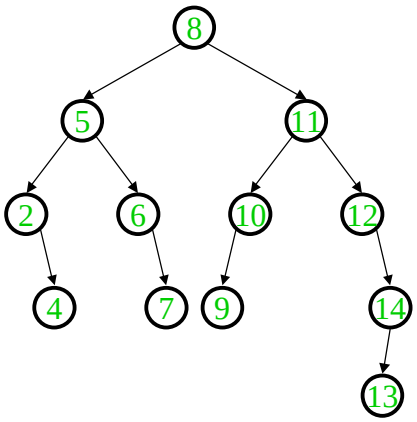
	unsorted array	sorted array	linked list
insert	find + O(n)	O(n)	find + O(1)
find	O(n)	O(log n)	O(n)
delete	find + O(1) (if no shrink)	O(n)	find + O(1)

Goal:

fast find like sorted array,
dynamic inserts/deletes like linked list

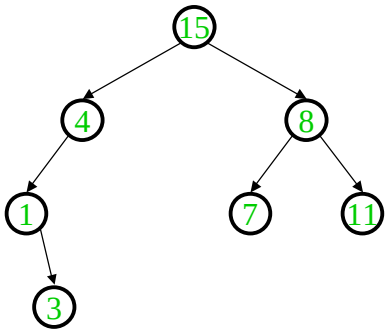
Binary Search Tree
Dictionary Data Structure

- Binary tree property
  - each node has ≤ 2 children
  - result:
    - storage is small
    - operations are simple
    - average depth is small
- Search tree property
  - all keys in left subtree smaller than root's key
  - all keys in right subtree larger than root's key
  - result:
    - easy to find any given key

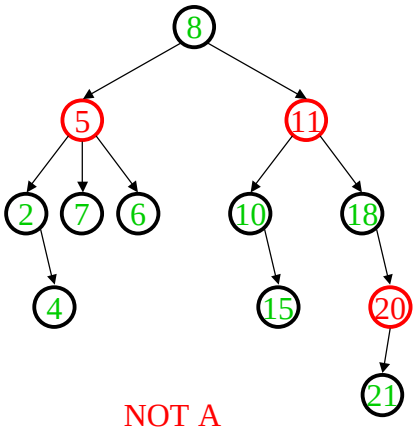


Getting to Know BSTs

Example and Counter-Example



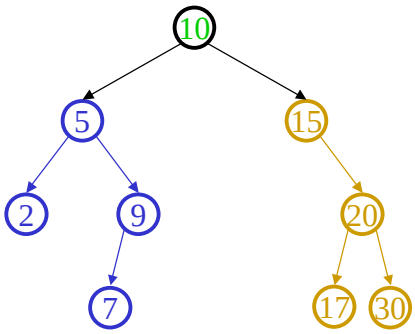
BINARY SEARCH TREE



NOT A
BINARY SEARCH TREE

Getting to Know All About BSTs

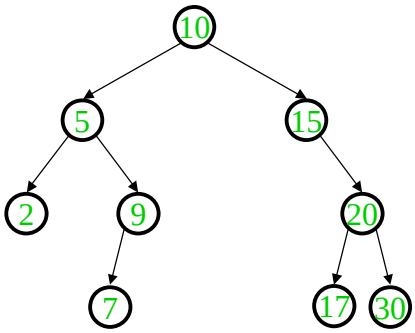
In Order Listing



In order listing:
2→5→7→9→10→15→17→20→30

Getting to Like BSTs

Finding a Node

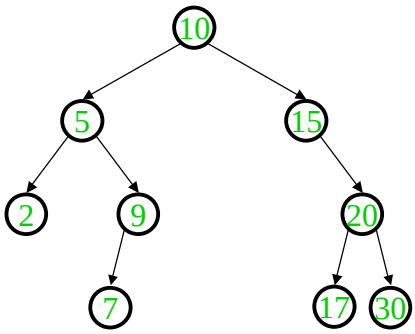


runtime:

```
Node *& find(Comparable key,
             Node *& root) {
  if (root == NULL)
    return root;
  else if (key < root->key)
    return find(key,
                root->left);
  else if (key > root->key)
    return find(key,
                root->right);
  else
    return root;
}
```

Getting to Hope BSTs Like You

Insert




```
void insert(Comparable key,
            Node * root) {
  Node *& target =
      find(key, root);

  assert(target == NULL);

  target = new Node(key);
}
```

runtime:

Digression: Value vs. Reference Parameters

- Value parameters (Object foo)
  - copies parameter
  - no side effects
- Reference parameters (Object & foo)
  - shares parameter
  - can affect actual value
  - use when the value needs to be changed
- Const reference parameters (const Object & foo)
  - shares parameter
  - cannot affect actual value

BuildTree for BSTs

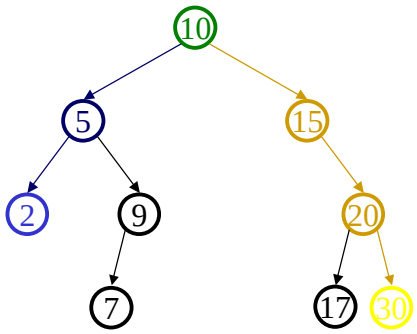
- Suppose the data 1, 2, 3, 4, 5, 6, 7, 8, 9 is inserted into an initially empty BST:
  - in order
  - in reverse order
  - median first, then left median, right median, etc.

Analysis of BuildTree

- Worst case: $O(n^2)$ as we've seen
- Average case assuming all orderings equally likely:

Bonus: FindMin/FindMax

- Find minimum
- Find maximum



To Do

- Start Project II
- Answer the Quiz questions you missed
- Read chapter 4 in the book

Coming Up

- A day off!! (July 4th, Wednesday)
- Second homework due (July 5th)
- Third Quiz (also July 5th)
- A bit more Binary Search Trees
- Self-balancing Binary Search Trees
- **Huge** Search Tree Data Structure