3 - Trees

Joseph Afework
CS 241

Dept. of Computer Science
California Polytechnic State University, Pomona, CA

Agenda

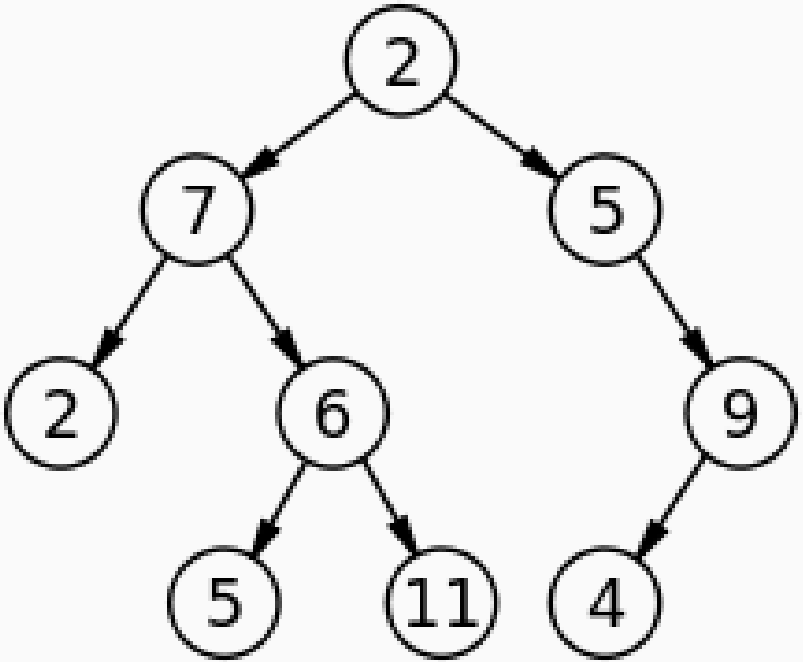
- Intro
- Application
- Terms (Lots of Terms)
- Types of Trees

Reading Assignment

- Read Chapter 23 - Trees
  - Chapter 23 All Sections
  - Read Chapter Summary Section (end of chapter)

- Read Chapter 24 - Binary Trees
    - Chapter 24 All Sections
    - Read Chapter Summary Section (end of chapter)

Tree



Intro

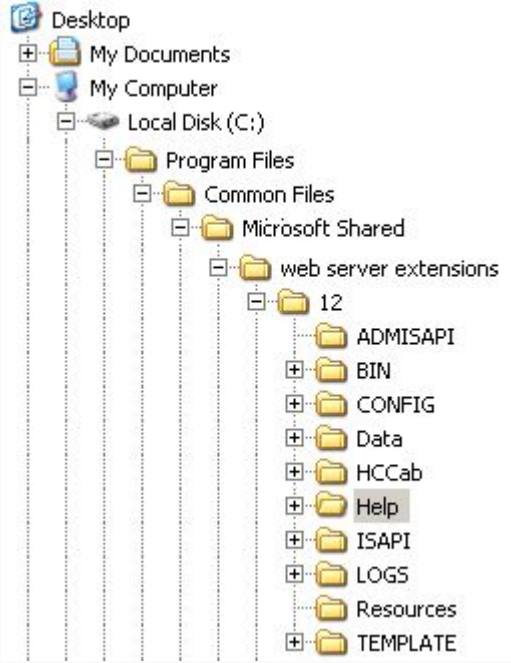
What is a tree?

A tree is a widely-used data structure that emulates a hierarchical tree structure with a set of linked nodes.

Applications

- Representing Hierarchical data
- Encoding Relationships between data
  - Generally Bi-Directional

- Ex.
  - Family Tree
  - Folder Directories



Origins

- **Non-linear organization:**
  - Components do not form a simple sequence of first element, second element, etc

- **Names are from Botany and Genealogy:** node, branch, depth (height), root, leaf, non-leaf (inner node), path, forest; parent, child, sibling, cousin, ancestor, descendant, and subtree.

Terminology

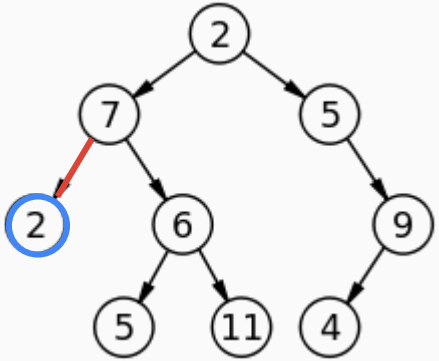
1. **Tree**
    - Properties that describe the tree as a whole

2. **Nodes**
    - Properties that describe the nodes of the tree

Tree Terms

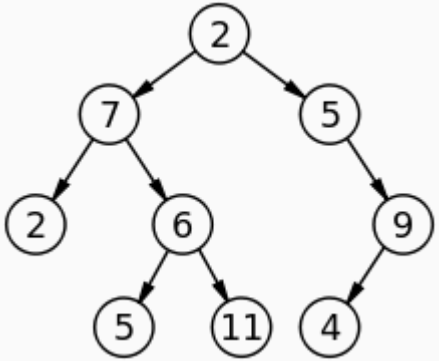
Nodes (Item)

Edges (Relationships)



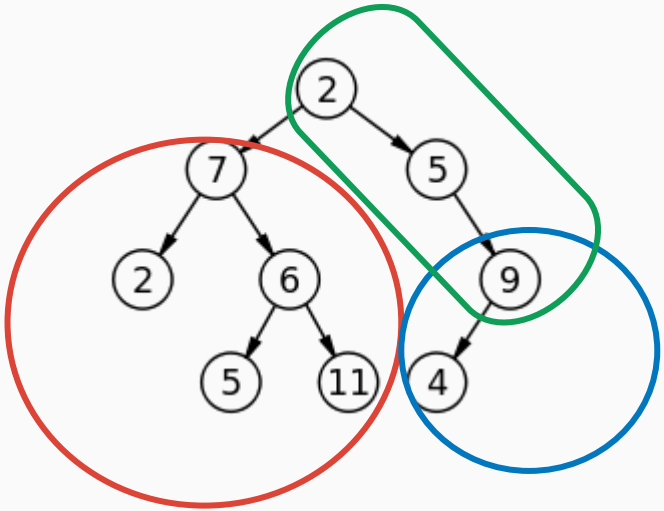
Tree Terms Contd.

- **Root**
  - Special Node - Node without a parent
  - Only 1 in a tree
- **Internal Nodes**
  - Nodes with at least one child
- **External Nodes**
  - (leaf) - node without any children
- **Height**
  - Defined by the maximum node depth



Tree Terms Contd.

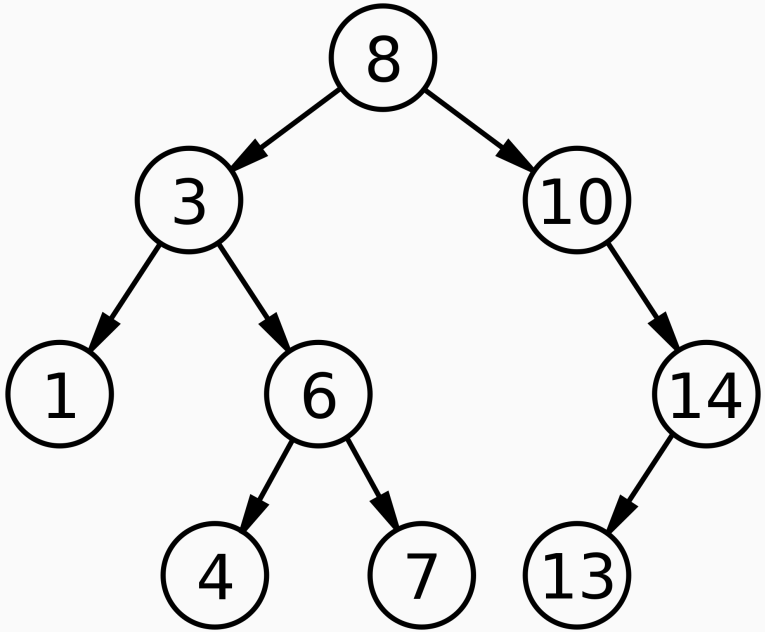
- **Subtree of a node**
  - A tree rooted at the child of a node
- **Subtree of a tree**
  - A subtree of the root node
- **Path**
  - Every node has a path that links the node to the root of the tree



ICE 3.1 Trees

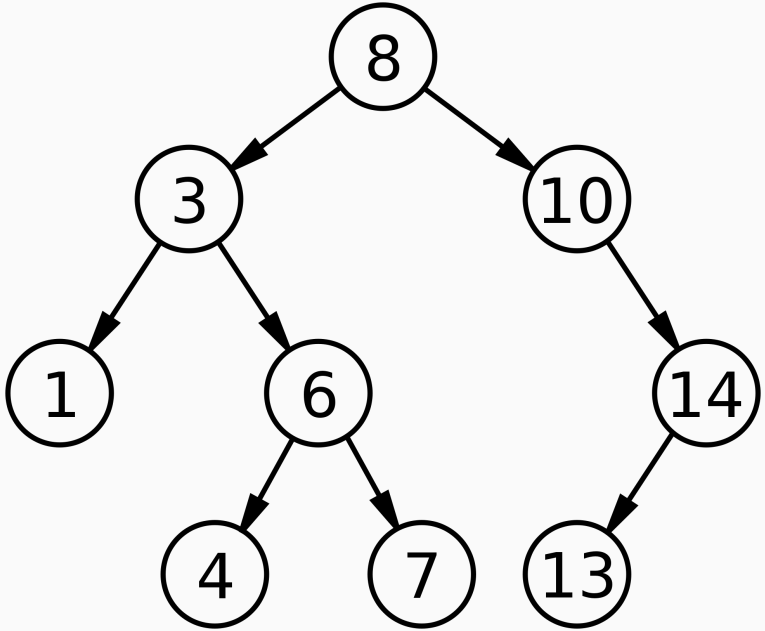
Identify the following:

1. Root Node
2. All Internal Nodes
3. All External Nodes
4. Height of Tree



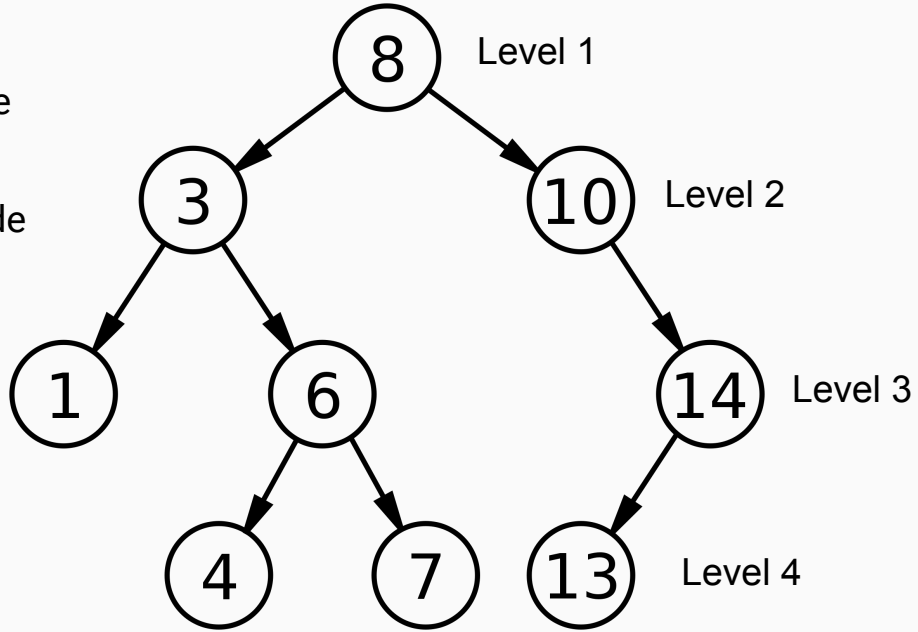
ICE 3.1 Trees - Solution

- Root Node
- All Internal Nodes
- All External Nodes
- Height of Tree



Node Terms

- Ancestors
  - Parents… grandparents of a node
- Descendents
  - Children… grandchildren of a node
- Depth
  - The level at which a node lies



Classifications

1. **By maximum number of branches:**
    - binary, ternary, n-way
2. **By heights of subtree:**
    - balanced, unbalanced trees
3. **By changeability:**
    - static, dynamic trees

Types of Trees

- **Empty (Null)-tree:**
  - a tree without any node
- **Root-tree:**
  - a tree with only one node
- **Binary tree:**
  - a tree in which each node has at most two children (parent, left, and right)

Types of Trees Contd.

- **Two tree:**
  - a binary tree that either is empty or each non-leaf has two children
- **Heap:**
  - a tree where parent node has bigger (smaller) value than children

Binary search tree (BST), 2-3 tree, AVL tree, B-tree, Huffman tree, Red-Black tree, Game tree, Spanning tree, etc.

Why Trees?

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Why Trees?

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Why Trees?

O(Log(n)) Operations

Learning Outcomes

- Understand the tree data structure and its related terminologies.