CS 261-020
Data Structures

Lecture 9
Midterm Report
Binary Trees
2/3/22, Thursday
Odds and Ends

• Assignment 3 posted

• No quiz this week

• Don’t forget to demo your assignment 2!
Lecture Topics:

• Midterm Report
• Binary Trees
Trees

• **Tree**: non-linear data structure, represents data as a hierarchical structure, encoding the hierarchical relationships between different elements

![Tree Diagram]

• **Node**: each individual data element in a tree
  • Contains the data element and points to other nodes

• **Edge (arc)**: an encoded relationship between data elements
  • Represents directed relationships
Tree Examples

- Examples:
  - Computer’s filesystem
  - Object model of a web page
  - Compiler’s abstract syntax tree of a program
Trees

- **Parent**: A node P in a tree is called the parent of another node C if P has an edge that points directly to C.
  - A is parent of B, C, D; B is parent of E and F
- **Child**: A node C in a tree is called the child of another node P if P is C’s parent.
  - B, C, D are children of A; J, K are children of G
- **Sibling**: A node S₁ is the sibling of another node S₂ if S₁ and S₂ share the same parent node P
  - B, C, D are siblings; J, K are siblings
- **Descendant**: The descendants of a node N are all of N’s children, plus its children’s children, and so forth.
  - E, F, and I are descendants of node B, and nodes H and L are descendants of node D
- **Ancestor**: A node A is the ancestor of another node D if D is a descendant of A
  - E, B, and A are ancestors of I, and G, C, and A are ancestors of node K
Trees

- **Root:** Ancestor of all other nodes in the tree. Each tree has exactly one root.
  - node A is the root.
- **Interior (node):** A node has at least one child.
  - A, B, C, D, E, G, and H are interior nodes.
- **Leaf (node):** A node has no children.
  - F, I, J, K, and L are leaves.
- **Subtree:** the portion of a tree that consists of a single node $N$, all of $N$’s descendants, and the edges joining these nodes.
  - the subtree rooted at node B contains the nodes B, E, F, and I and the edges joining those nodes.
Trees

- **Path**: the collection of edges in a tree joining a node to one of its descendants.

- **Path length**: the number of edges in that path.
  - the path from C to K has length 2, since it contains 2 edges.

- **Depth**: The depth of a node $N$ in a tree is the length of the path from the root to $N$.
  - the depth of K is 3.
  - The depth of A (root) is 0.

- **Height**: The maximum depth of any node in the tree.
  - The tree has height 3
Trees

• Constraints to be counted as a tree:
  • Each node in the structure may have only one parent.
  • The edges of the structure many not form any cycles.
    • there cannot be a path from any node to itself.
Binary Trees

• *Binary Tree*: a tree in which each node can have at most two children (left child and right child).

• *Left subtree*: the subtree rooted at that node’s left child

• *Right subtree*: the subtree rooted at that node’s right child
Binary Trees

• **Full Binary Tree**: a binary tree that every interior node has **exactly two** children.
• **Perfect Binary Tree**: a full binary tree where all the **leaves are at the same depth**.
  • If a perfect binary tree has **height** $h$, then
    • It has $2^h$ leaves
    • It has $2^{h+1} - 1$ total nodes
  • If a perfect binary tree has $n$ nodes, then its height is **approximately** $\log(n)$
Binary Trees

• *Complete Binary Tree*: a binary tree that is perfect except for the deepest level, whose nodes are all as far left as possible
Binary Search Trees

• Recall: each node in a tree represents a data element.

• Represent each data element using a key (identifier)
  • The data element may also contain other data, which we can refer to as its value

• Assuming these keys can be ordered in relation to others
  • i.e., integer keys can be ordered numerically, string keys can be ordered alphabetically
A *binary search tree* (BST) is a binary tree that:
- the key of each node $N$ is **greater than** all the keys in $N$’s left subtree and **less than or equal to** all the keys in $N$’s right subtree

*Note: A BST does NOT have to be full, perfect, complete, etc.*
Next Lecture: BST Operations

• BST Operations:
  • Finding an element
  • Inserting a new element
  • Removing an element

• Runtime Complexity of BST operations
• BST traversals