CS 209
Data Structures and Mathematical Foundations

03 / 01 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions/Comments?
- Trees
- Binary Trees
- Binary Search Trees

Trees

- Trees
  - Terminology
  - Binary tree example application
  - binary trees
    - two ideas for representing them in code
    - preOrder, inOrder, postOrder traversals using recursion
    - start binary search trees

Tree terminology definitions

- The next group of data structures that we will learn are trees.
- As we learn about them over the next several class periods, we will discuss what they can be used for, and why.
- Some things seem to be naturally modelled as trees (think biology classification of living things --- Kingdom, Phylum, Class, etc.) and some algorithms depend on the data to be in some kind of tree to work efficiently.

Tree terminology definitions

- A **tree** is a data structure consisting of a finite (possibly empty) set of nodes. If the tree is nonempty it has one **root** node.
  - Each node in a tree can have 0 or more **children**.
  - Each node in a tree has exactly one **parent,** except the root which has 0 parents.
  - Starting at any node in the tree you can trace a path back to the root by following the parent at each step.
- Picture on the board.

Tree terminology definitions

- A **binary tree** is a data structure consisting of a finite (possibly empty) set of nodes. If the tree is nonempty it has one **root** node.
  - Each node in a binary tree can have **0, 1 or 2 children**.
  - Each node in a tree has exactly one **parent,** except the root which has 0 parents.
  - Starting at any node in the tree you can trace a path back to the root by following the parent at each step.
- Picture on the board.

Tree terminology definitions

- The **root** node in a tree is the one without a parent.
- The **parent** of a node *n*, in a tree, is the node that has *n* as one of its **children**.
- **Leaf** nodes are those that have 0 children.
- A **subtree** is part of a tree that has as its root any node in the original tree.
- The **depth of a node** is the number of steps away that node is from the root. The depth of the root is 0. The depth of the root's children are 1, etc.
- The **depth of a tree** is the maximum depth of any of its leaves. This is also the **height** of a tree.

Tree terminology definitions (corrected)

- A **perfect binary tree** is a binary tree where every leaf has the same depth AND all internal nodes have 2 children.
- A **complete binary tree** is a binary tree where the leaves with the maximum depth are all on the left (and any other leaves are only one depth fewer). In other words, every level of the tree except the deepest level, must contain as many nodes as possible. At the deepest level the nodes must be as far left as possible.

Tree terminology definitions

- Two nodes are **siblings** if they have the same parent.

Exercise

- How many nodes are there in a perfect binary tree of depth 8?

Exercise

- How many nodes are there in a perfect binary tree of depth 8?
  - root level has 1 node (which is 2^0)
  - next level has 2 nodes (which is 2^1)
  - next level has 4 nodes (which is 2^2)
  - next level has 8 nodes (which is 2^3)
  - next level has 16 nodes (which is 2^4)
  - next level has 32 nodes (which is 2^5)
  - next level has 64 nodes (which is 2^6)
  - next level has 128 nodes (which is 2^7)
  - next level has 256 nodes (which is 2^8)
- These added up are $2^9 - 1 = 512 - 1 = 511$.

Exercise

- How many nodes are there in a complete binary tree of depth 4?

Exercise

- How many nodes are there in a complete binary tree of depth 4?
  - root level has 1 node (which is 2^0)
  - next level has 2 nodes (which is 2^1)
  - next level has 4 nodes (which is 2^2)
  - next level has 8 nodes (which is 2^3)
  - next level has anywhere from 1 to 16 nodes
- So, any value in range: 16 to 31 nodes.
- When a binary tree has 31 nodes is it guaranteed to be a perfect binary tree?

Exercise

- Is a binary tree of 12 nodes
  - perfect?
  - complete?

Exercise

- Is a binary tree of 12 nodes
  - perfect?
    - No
  - complete?
    - possibly

Binary Search Trees

- A binary **search** tree is a binary tree in which all nodes have the following ordering property.
  - The data in the left subtree of a node is less than the data in the node and the data in the right subtree of a node is greater than or equal to the data in the node.
  - This ordering property is true at every node.

Application of binary trees

- The game 20 questions
  - The game consists only of yes/no questions
- Each node can contain a question (or answer)
- If it contains a question,
  - If the answer to the question is yes then the left child node contains the next question to ask (or the answer to give)
  - If the answer to the question is no then the right child node contains the next question to ask (or the answer to give)

Application of binary trees

- What could you say about the nodes that contain answers, not questions?
- Assuming a perfect binary tree of height 20 to be used for the game, how many possible answers could your tree differentiate between?

Application of binary trees

- What could you say about the nodes that contain answers, not questions?
  - They are leaves.
- Assuming a perfect binary tree of height 20 to be used for the game, how many possible answers could your tree differentiate between?

  $2^{20} = 1048576$ (over 1 million)

  that's the number of leaves

  The number of questions in the tree are $2^{20} - 1 = 1048575$

  The number of nodes in the tree are $2^{21} - 1 = 2097151$

Implementation of binary trees

- Similar to how we implemented a linked list of nodes, we can implement a binary tree of nodes where each node contains a reference to a left child and a right child (each of which could be None.)

```
class BTNode:
        # instance variables
        # data – the data in the BTNode
        # left – the BTNode to the left
        # right – the BTNode to the right

        # constructor that sets left and right to None
        def __init__(self, d):
            self.data = d
            self.left = None
            self.right = None
```

Implementation of binary trees

```
class BinaryTree:
        # instance variable
        # root – a BTNode that is the root of the tree

        # constructor that sets root to None
        def __init__(self):

            self.root = None


        # plenty more methods here to insert nodes, etc...
```

Binary Search Trees

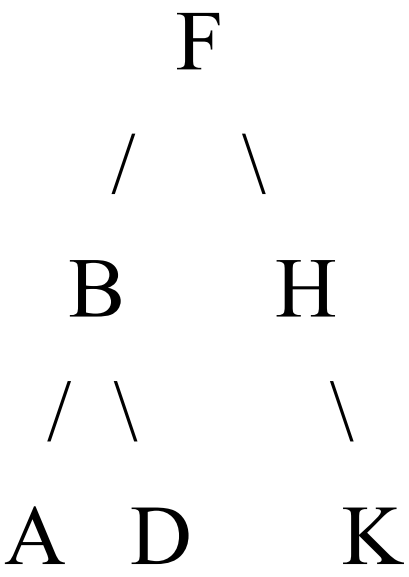
- Binary Search Trees (BSTs)
  - A *tree* that is both a *binary tree* and a *search tree*.
  - we know the definition of a tree and a binary tree so:
  - We just need to define search tree.
    - A *search tree* is a tree where
      - every subtree of a node has data (aka keys) less than any other subtree of the node to its right.
      - the keys in a node are conceptually between subtrees and are greater than any keys in subtrees to its left and less than or equal to any keys in subtrees to its right.

Binary Search Trees

- A *binary search tree* is a tree that is a binary tree and is a search tree. In other words it is a tree that has
  - at most two children for each node and
  - every node's left subtree has keys less than the node's key, and every right subtree has keys greater than or equal to the node's key.

  - Definitions taken from http://www.nist.gov/ (The National Institute of Standards and Technology)

Binary Search Trees



- Assume Alphabet ordering of letters.
- Let's verify whether it is indeed a binary search tree.
  - What properties does it need to have again?
  - Does it satisfy those properties?