CSCI 3333 Homework: Trees

1 Combinatorics

Problem 1. Give a formula for each of the following quantities:
- The number of leaves in a perfect tree of height $h$.
- The number of nodes in a perfect tree of height $h$.
- The number of internal nodes in a perfect tree of height $h$.
- The number of nodes in a perfect tree with $l$ leaves (assume $l$ is valid).

Problem 2. Give a formula for each of the following quantities:
- The maximum number of nodes in a tree of height $h$.
- The minimum number of nodes in a tree of height $h$.
- The minimum number of nodes in a balanced tree of height $h$.
- The maximum number of nodes in an unbalanced tree of height $h$.

2 BST Structure

Problem 3. Draw a BST containing the elements 10, 3, 6, 5. If more than one such BST exists, draw a second with 10 as the root.

Problem 4. Draw a balanced BST containing the set of elements $S = \{5, 8, 3, 2, 1, 9\}$. For which elements $x \in S$ does there exist a balanced BST with $x$ as root containing the elements of $S$?

3 BST Algorithm Examples

Problem 5. Give the sequence of nodes visited when searching for the following elements:
- 2
- 10
- 7
- 14
- 30

Problem 6. See Figure 2. Give a sequence of integers such that inserting the elements of the sequence into an empty BST one-at-a-time (in the given order) results in a BST with the shape seen the left portion of Figure 2. Do the same for the shape in the right portion of Figure 2.
Figure 1: The BST for Problem 5.

Figure 2: The BST shapes for Problem 6.

4 Algorithm Implementation

**Problem 7.** Implement a C++ function that returns the smallest value in a non-empty binary search tree. Assume each **Node** object has an **int** instance variable named **v** storing the value at the node.

**Problem 8.** Implement the same function as in Problem 7 but for general binary trees.

**Problem 9.** Implement a C++ function that returns the longest string in a binary tree that starts with a given **prefix** string. If no such string exists, the function should return the empty string.

5 Algorithm Design

**Problem 10.** Design an $O(h)$-time algorithm that finds the second largest element of a binary search tree of height $h$. 

Problem 11. Design a $O(n)$-time algorithm that determines whether a BST $T$ of size $n$ contains two floats $a$ and $b$ such that $a + b == 0$. 