1 Height-Balanced

Problem 1. Give a BST of height 4 with each of the following properties (if one exists):

- Balanced and height-balanced.
- Balanced but not height-balanced.
- Height-balanced but not balanced.
- Neither balanced nor height balanced.

Solution 1.

- One exists. For instance, the complete tree of height 4.
- None exist, since “balanced” is a stronger requirement than “height-balanced”.
- One exists (see left tree in Figure 1).
- One exists (see right tree in Figure 1).

Problem 2. Call a node almost-height-balanced if the heights of its left and right subtrees differ by at most 2. Call a binary tree almost-height-balanced if all of its nodes are almost-height-balanced.

- Give a recurrence relation $f(h)$ for the minimum number of nodes in an almost-height-balanced tree of height $h$. 
• Use repeated substitution to obtain a recursive inequality of the form \( f(h) > a \cdot f(h-b) \) where \( a, b \) are integers larger than 1.

• Find a closed form for the equality \( f(h) = a \cdot f(h - b) \).

• Give an asymptotic lower bound for \( f(h) \), knowing a closed form for \( f(h) = a \cdot f(h - b) \) and that \( f(h) > a \cdot f(h - b) \).

• Give an asymptotic upper bound for the height of an almost-height-balanced tree with \( n \) nodes.

Solution 2.

• \( f(h) = f(n - 1) + f(n - 3) + 1 \) with \( f(0) = 1, f(1) = 2, f(2) = 3 \).

\[
\begin{align*}
  f(h) &= f(h - 1) + f(h - 3) + 1 \\
  &= [f(h - 2) + f(h - 4) + 1] + f(h - 3) + 1 \\
  &= f(h - 2) + f(h - 3) + f(h - 4) + 2 \\
  &= [f(h - 3) + f(h - 5) + 1] + f(h - 3) + f(h - 4) + 2 \\
  &= 2f(h - 3) + f(h - 4) + f(h - 5) + 3 \\
  &> 2f(h - 3)
\end{align*}
\]

• Starting with \( f(h) = 2f(h - 3) \) from the previous question:

\[
\begin{align*}
  f(h) &= 2f(h - 3) \\
  &= 2(2f((h - 3) - 3)) \\
  &= 2^2 f(h - 3 \cdot 2) \\
  &= 2^3 f(h - 3 \cdot 3) \\
  &= 2^k f(h - 3k)
\end{align*}
\]

So if \( k = h/3 \), then \( f(h) = 2^{h/3} f(h - 3(h/3)) = 2^{h/3} f(0) = 2^{h/3} \). So \( f(h) \geq 2^{h/3} \).

• Starting with \( n = f(h) \geq 2^{h/3} \) from the previous question:

\[
\begin{align*}
  n &\geq 2^{h/3} \\
  \log_2(n) &\geq h/3 \\
  3\log_2(n) &\geq h
\end{align*}
\]

So the height is at most \( 3\log_2(n) = O(\log(n)) \).
2 Algorithm Examples

Problem 3. Draw a smallest tree that cannot be height-balanced with one rotation, but can be height-balanced with two rotations.

Solution 3. See Figure 2.

Problem 4. Draw the AVL tree that results from inserting the following elements in the given order: 1, 2, 3, 4, 5, 6. How many total rotations were done during the insertions?

Solution 4. See Figure 3. Three rotations were done during the insertions.

Problem 5. Draw the AVL tree that results from inserting the following elements in the given order: 6, 1, 2, 3, 4, 5. How many total rotations were done during the insertions?

Solution 5. See Figure 3. Five rotations were done during the insertions.

3 Algorithm Implementation

Problem 6. Implement a C++ function that returns whether a tree is height-balanced. You may assume that a C++ function with prototype int height(Node* root) has already been implemented.

Solution 6.
Figure 4: The solution to Problem 5.

```c
bool height_balanced(Node* root) {
    if (!height_balanced(root->left))
        return false;
    if (!height_balanced(root->right))
        return false;
    int imbalance = height(root->left) - height(root->right);
    return (-1 <= imbalance && imbalance <= 1);
}
```

**Problem 7.** Implement a C++ function that rotates a tree rightward around its root.

**Solution 7.** This implementation returns a pointer to the new root of the tree:

```c
Node* right_rotate(Node* root) {
    Node* lc = root->left;
    root->left = lc->right;
    lc->right = root;
    return lc;
}
```