Tutorial 04 - Dictionaries & Amortized Analysis
CS 240E Winter 2022
University of Waterloo
Monday, January 31, 2022

1. **2-AVL Tree:**
   Let a 2-AVL tree be a binary search tree where for every node, the difference of heights of its left and right subtree is at most 2. Prove that a 2-AVL tree has height at most $3 \log n$ where n is the number of nodes in the tree.

2. **Binary Counter:**
   A *binary n-bit counter* stores the current value of a counter as an array A of length n that contains 0 or 1. It supports the operation $Increment$, which adds 1 to the counter and operates as shown below:

```
void Increment(A, n) {
// A is an n-bit counter whose
// value is less than 2^n - 1
    i <- 1
    while (A[n-i] != 0) {
          A[n-i] <- 0
          i <- i + 1
    }
    A[n-i] <- 1

}
```

   The running time for $Increment(A, n)$ is $\Theta(k)$, where k is the final value of variable i. This is $\Theta(n)$ in the worst case. Argue that the *amortised* cost of $Increment(A, n)$ is $\Theta(1)$.

3. **Balanced BST:**

   Recall that a binary search tree is called *perfectly balanced* if for every node v we have

   $$|v.left.size - v.right.size| \leq 1,$$

   i.e., the size-difference between the left and right is as small as possible. Show that in any perfectly balanced binary search tree T, the leaves are only on the bottom two levels.

   Hint: First consider the case where $n = 2^k - 1$ for some integer k. Then consider the case where $n = 2^k$ for some integer k. Finally for arbitrary n, let k be the integer with $2^k \leq n < 2^{k+1}$. In all three cases, what are the sizes of the subtrees, and hence where are the leaves, relative to k?