COMS W3137 - Theoretical Homework 2 - Written Solutions

Question 1: Amortized Analysis

a) Each push costs 1, so $T_{push}(n) = n$

For copying, the first time we double the array, we need to copy 1 element. The second time, we need to copy 2. The third time, we need to copy 4, etc.

$T_{copy}(n) = 1 + 2 + 4 + ... + \frac{n}{4} + \frac{n}{2} + n = \sum_{i=0}^{\log n} \frac{n}{2^i} < 2n$

So $T(n) = T_{push}(n) + T_{copy}(n) < 3n$.

b) Assume the amortized cost for each push(x) operation is 3. After pushing, because the actual cost of push is 1, we have 2 credits left. We associate these credits with the element on stack that was pushed.

Now assume that $k = L$ and we perform another push. The actual cost required for copying the items is k. Clearly, if all items still have 2 credits we can pay for the copy. After the copy, the items on the stack have 1 credit left. If we have to copy again, the credit associated with these items will become 0, which may look like a problem at first glance.

The important insight is that, when we need to copy, the items in the second half of the array must always have 2 credits left. This is because these items were pushed after the last time we duplicated the array.

There are $k/2$ of these items, to the total ramining credit is at least $k/2 \cdot 2 = k$, which is enough to pay for the copying.

c) This is easiest to show by computing the sum again. Without loss of generality, assume $c < k$.

Every time we need to copy elements there are c more elements than for the previous copy step. So

$$T(copy) > c + 2c + 3c + 4c + \ldots + (n - c) + n$$
$$= \sum_{i=1}^{\frac{n}{c}} i \cdot c = \frac{n(c+n)}{2c} = \Theta(n^2)$$

Question 2: Structural Induction

Proof by structural induction: Any full binary tree containing N nodes has (N-1)/2 internal nodes.

Base case: A full binary tree with 1 node has (1-1)/2=0 internal nodes (because a single node is a leaf).

Inductive step:
Assume that any full binary tree with 1 <= k <= N nodes has (k-1)/2 internal nodes.

Any full binary tree with N+1 > 1 nodes consists of a root node and two non-empty subtrees, containing N_{left} and N_{right} nodes, respectively. Then $N = N_{left} + N_{right}$ (the root node accounts for the +1).

$$\begin{array}{ccc} & \text{root} & \\ / & & \backslash \\ N_{left} & & N_{right} \end{array}$$

Clearly $1 <= N_{left} <= N$ and $1 <= N_{left} <= N$. Therefore, **by the inductive hypothesis,** the left subtree has $(N_{left} - 1) / 2$ nodes and the right subtree has $(N_{right} - 1) / 2$ nodes.

The root node adds an internal node, so the total number of internal nodes in a full binary tree with N+1 nodes is

$$
\begin{aligned}
& (N_{left} - 1) / 2 + (N_{right} - 1) / 2 + 1 \\
= \; & (N_{left} - 1 + N_{right} - 1) / 2 + 1 \\
= \; & (N_{left} + N_{right} - 2) / 2 + 1 \\
= \; & (N_{left} + N_{right})/ 2 - 1 + 1 \\
= \; & N / 2 \\
= \; & ([N+1] -1) / 2.
\end{aligned}
$$

∎

Question 3: Tree Traversal Sequences

Proof by counter-example: Consider the following two trees

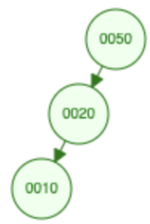
```
      A               A
     /                 \
    B                   B
   /                   /
  C                   C
```

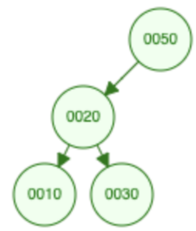
The pre-order traversal for both trees is A B C, the post-order traversal for both trees is C B A. These pre and post order traversals together do not correspond to a unique tree.

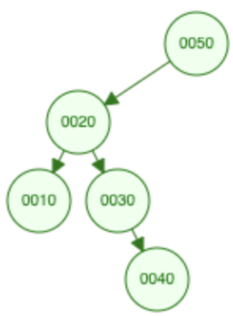
NB: The in-order traversal is C B A for the left tree, and A C B for the right tree. The in-order traversal together with either post or pre-order does allow you to uniquely reconstruct the tree.

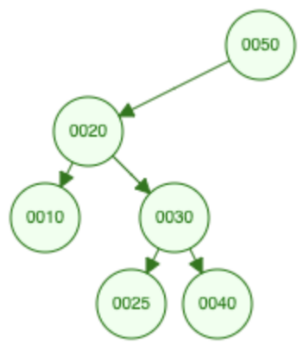
Question 4: BSTs and AVL Trees

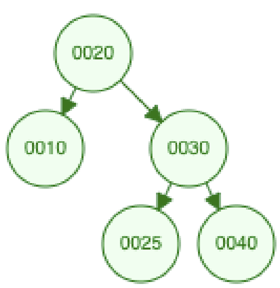
To check your solution, please use the BST and AVL visualizer here:

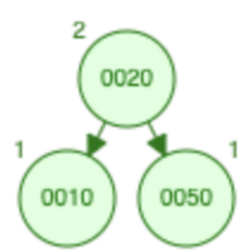


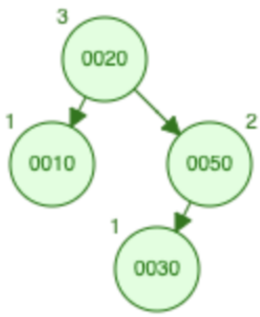


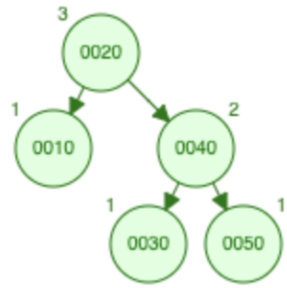


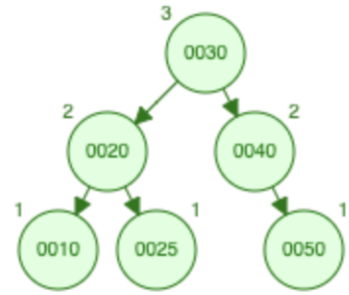












https://www.cs.usfca.edu/~galles/visualization/AVL.html

Question 5: Minimal AVL Trees

```
Height 3

          o
        /   \
       o     o
      / \   /
     o  o  o
    /
   o
```

```
Height 2

          o
         / \
        o   o
       /
      o
```

```
Height 4

                o
             /     \
           /         \
          o           o
        /   \        / \
       o     o      o  o
      / \   /      /
     o  o  o      o
    /
   o
```

To satisfy the AVL balance condition, a subtree of height k needs to be balanced with a subtree of height k-1 (or k).
So in order to construct a tree with minimal number of nodes of height k+1, use a new root, attach a minimal subtree of height k and as the left subtree, and balance it with a minimal subtree of height k-1 as the right subtree.