COM S 311 EXTRA HOMEWORK

NATHAN TUCKER (NJTUCKER@IASTATE.EDU)

This assignment represents my own work in accordance with University regulations. Collaboration done with Haadi Majeed (hmajeed@iastate.edu) and Matthew Hoskins (mgh@iastate.edu)

- Nathan Tucker

Problem

Consider a nearly complete binary tree that is represented by an array A of elements, where each node of the tree corresponds to an element of the array, and the root of the tree corresponds to $A[1]$. Let n be the number of nodes in the tree, which is also the length of the array A. For index $i = 1, 2, 3, ..., n$, the value of node i is $A[i]$, and the indices of its parent, left child and right child (if they exist) are $\lfloor i/2 \rfloor$, $2i$ and $2i + 1$. The height of a node in the tree is the number of edges on the longest simple downward path from the node to a leaf, and the height of the tree is the height of the root.

Given an array A of n numbers which represents a nearly complete binary tree of height h, we want to produce an array B of length h such that for $k = 1, 2, 3, ..., h$, $B[k]$ is the value at the last node of height k (with the largest index). Note that h is a function of n. For this assignment, the values in the nodes may not satisfy a heap property.

Example (30 points). Given the following array A of numbers, write down all the values in the array B in increasing order of array indices.

```
Array A: 10 30 20 50 70 55 65 25 45 85 15 90 75 60 35 80 95 40


Index  :  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18


Array B: 45 50 30 10


Index  :  1  2  3  4
```

Algorithm (35 points). Design an $O(\log n)$ algorithm for producing the output array B on an input array A of length n. Analyze your algorithm to obtain its running time.

```
FindBArray
    Input: An array reprentation of a Binary Search Tree, $A$
    This would technically be a complete tree, but add check if A size is 1
    if $A.size == 1$ then
        return
    $height = 0$
    $curNode$ = A[A.$size$]
    while $curNode \geq 1$ do
        $height = height + 1$
        $curNode = \lfloor \frac{curNode}{2} \rfloor$
    Let B be a new array of size $height$
    $curNode$ = A[A.$size$]
    for $k$ from 0 to $height$ do
        Add the parent of the current node
        B.add(A[$\lfloor \frac{curNode}{2} \rfloor$])
        $curNode = \lfloor \frac{curNode}{2} \rfloor$
    return B
```

For this algorithm, we can exploit a property of Parent Array representations of Binary Search Trees to get the height and the last index of the height very easily. First, to find the height, we can start at the last node in the array, as we know it will be a leaf at the greatest depth possible for this nearly-complete tree. Then we can keep finding the parent until we reach the root node, then we have the total height. Because we are calculating the height by going in increments of $\lfloor \frac{i}{2} \rfloor$ each iteration we know we have a runtime of $\log n$.

Now that we know the height, we can declare a new array B with the length of the height we just found. Once we've done that, it's as easy as following the exact same path to the root node of the BST to get the nodes at the greatest index of each height. This follows the exact same path to find the height, so for the same reason, we can say we have a runtime of $\log n$ for this search and addition to B.

Now with both of those runtimes, we can sum them together to have a total runtime of $2 \log n$ or a big O of $O(\log n)$.