COM S 311 HOMEWORK 4

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: 10 20 65 60 40


Index  :  1  2  3  4  5
```

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.

```
1  FindBArray
       Input: An array reprentation of a Binary Search Tree, $A$
2      Let $B$ be a new array of length $h$, where $h$ is the height of the BST represented by $A$
3      Let $i$ be 1, let $k$ be 1
4      while $i \leq A.size$ do
5          B[$k$] = A[$i$]
6          $i = 2i + 1$
7          $k = k + 1$
8      We've maybe missed the last node in the BST because we've gone "out of
        bounds" trying to find only right children, need to add if not full and
        complete BST
9      if $\lfloor \frac{i}{2} \rfloor = A.size$ then
10         B[$k + 1$] = A[$A.size$]
```

For this algorithm, it really just boils down to finding one of two things: the right child of the current node or, if one does not exist, the last node in the BST array. The only instance in which this fails is in the case of a full and complete BST, because the last node in the array *will* be the last node we're looking for.

We start at the first index of the array A and will search for the right children only, so rather than a linear search of the array, we are looking for right children only. To accomplish this, we increment i by $2i+1$ rather than just by $i+1$ in the case of a linear search. Because of this method of tree traversal, the runtime of the algorithm is $log(n)+1$ because of the $log(n)$ traversal and the final constant time check of the size, and an $O(1)$ array access. All of this leads to a runtime of $log(n)+1$, or a big O of $O(\log n)$