Weekplan: Binary Search Trees
The 02105+02326 DTU Algorithms Team

Reading


Exercises

1 Simulation and Properties

1.1 [w] Which of the following trees are binary search trees?

1.2 [w] Where are the elements with respectively the smallest and largest key located in a binary search tree?

1.3 [w] CLRS 12.1-1.

1.4 [w] Specify the pre-order, in-order og post-order sequence of keys for the tree in (b)

1.5 CLRS 12.1-2.

1.6 CLRS 12.1-3. Write pseudo code for the algorithm.

1.7 CLRS 12.2-1.

1.8 [BSc] CLRS 12.2-5. *Hint: prove by contradiction.*

2 Leaves and Heights

Let $T$ be a binary tree with $n$ vertices and root $v$.

2.1 Give a recursive algorithm that given $v$ computes the number of leafs in $T$. Write pseudo code for your solution.

2.2 Give a recursive algorithm that given $v$ computes the height of $T$. Write pseudo code for your solution.

2.3 [*] Implement your solution to compute the height.

3 More Recursion on Trees (Exam 2011)

This exercise is about rooted binary trees. Each node $x$ has fields $x.parent$, $x.left$, and $x.right$ denoting the parent, left child, and right child of $x$. For the root $root$, $root.parent = null$. Furthermore, we also store a field $x.size$ containing an integer. Consider the following algorithm.

```
ZERO(x)  
if x ≠ null then  
    ZERO(x.left)  
    ZERO(x.right)  
end if
```
3.1 Analyze the running time of the procedure \( \text{ZERO}(x) \) as a function of \( n \), where \( x \) is the root node in a tree with \( n \) nodes.

3.2 Let \( T(x) \) denote the subtree of the tree rooted at \( x \) and let \( |T(x)| \) denote the number of nodes in \( T(x) \). Give an algorithm, \( \text{INITSIZE}(x) \), that given the root node \( x \) of a tree sets \( y \text{size} \) to be \( |T(y)| \) for all nodes \( y \) in the tree. Write your algorithm in pseudocode and analyse the running time of your algorithm as a function of \( n \), where \( n \) is the number of nodes in the tree.

3.3 Given a node \( x \) with a child \( y \) of \( x \), we say that the edge \((x, y)\) is red if \( |T(x)| \) is at least twice as large as \( |T(y)| \). Give a recursive algorithm, \( \text{REDEDGE}(x) \), that given the root node, computes the number of red edges in the tree. Write your algorithm in pseudocode and analyse the running time of your algorithm as a function of \( n \), where \( n \) is the number of nodes in the tree.

3.4 Analyse and give an upper bound in \( O \)-notation on the maximum number of red edges on any path from the root of the tree to a leaf.

4 Traversal of Binary Search Trees

4.1 Give an algorithm that given a binary search tree \( T \) with a key in each vertex, determines if \( T \) satisfies the binary search tree property.

4.2 Give an algorithm that given a binary search tree \( T \) constructs a reversed binary search tree \( T^R \). \( T^R \) should be a binary search tree with the same keys as \( T \). For each vertex \( v \) in \( T^R \) the vertices in the left subtree must be \( \geq v \) and the keys in the right subtree must be \( \leq v \).

4.3 [x] Give an algorithm that given two binary search trees \( T_1 \) and \( T_2 \) constructs a single binary search tree with all the elements from both \( T_1 \) and \( T_2 \).

5 Perfectly Balanced Binary Search Trees Let \( A \) be a sorted array of \( n = 2^{h+1} - 1 \) distinct numbers. Give a sequence of insertions of the numbers in \( A \) into a binary search tree \( T \) such that \( T \) becomes a complete binary search tree of height \( h \).

6 Pre-Order Traversal [t] Implement a recursive algorithm for pre-order traversal of a binary tree.

7 Even More Recursion on Trees (Exam 2010) This exercise is about rooted binary trees. Each node \( x \) has fields \( x\text{.parent} \), \( x\text{.left} \), and \( x\text{.right} \) denoting the parent, left child, and right child of \( x \). For the root \( root \), \( root\text{.parent} = \text{null} \). Furthermore, we also store a field \( x\text{.label} \) containing a single character. Consider the following algorithm and binary tree.

\[
\text{PRINTTREE}(x) \\
\text{if } x \neq \text{null} \text{ then } \\
\quad \text{print } x\text{.color} \\
\quad \text{if } x\text{.left} \neq \text{null} \text{ then } \\
\quad\quad \text{PRINTTREE}(x\text{.left}) \\
\quad \text{end if} \\
\quad \text{if } x\text{.right} \neq \text{null} \text{ then } \\
\quad\quad \text{PRINTTREE}(x\text{.right}) \\
\quad \text{end if} \\
\text{end if}
\]

\[\text{C} \quad \text{R} \quad \text{O} \quad \text{L}\]
7.1 If $x$ is the root of the above tree, then $\text{PRINT_TREE}(x)$ outputs "CROOL". Explain how to modify $\text{PRINT_TREE}$ to print "COLOR" instead.

7.2 Give a recursive algorithm, $\text{INTERNAL}(x)$, that given the root node $x$ of a tree computes the number of internal nodes in the tree. Write your algorithm in pseudocode and analyse the running time of your algorithm as a function of $n$, where $n$ is the number of nodes in the tree.

7.3 We say that a tree has an R-path if there is a root-to-leaf path consisting of only nodes labeled R. Give a recursive algorithm, $\text{R-PATH}(x)$, that given the root node $x$ of a tree determines if the tree has an R-path. Write your algorithm in pseudocode and analyse the running time of your algorithm as a function of $n$, where $n$ is the number of nodes in the tree.