Cody Adams

CSC375-01

Winter 2009

5.5, 5.7, 5.8, 5.21, 5.22, 5.24

Extra credit: 5.1, 5.2, 5.3 (3 points each, up to +9)

Homework 7

5.1) Given a non-full binary tree, there exists an internal node with only one non-empty child, call this node N. When you remove N from the tree and replace it with its child, the resulting tree has higher ratio of non-empty nodes since one non-empty node and one empty node have been removed.

5.2) Base Case: Given a single leaf node ( no internal nodes, one leaf)

Induction Hypothesis: Assume THM holds for all trees having n-1 nodes

Induction Step: Given a tree T with n nodes; remove a single node from the tree and

label it tree T2. Accordingly, T2 has one more leaf node than it has internal nodes. Restoring the node to T gives two cases:

(1) T2 is T’s only child-- the number of internal nodes and leaves have not changed. □

(2) T2 is a child of an internal node in T. This internal node is the root of T2, so an equal amount of leaves and internal nodes were restored. □

5.5) bool search(BinNode\* subroot, int K)

{

if ( subroot == NULL )

return false;

if ( subroot->value() == K )

return true;

if ( search( subroot->right( ) ) )

return true;

return search( subroot->left( ) );

}

5.7) int TreeHeight( BinNode<int> \* node )

{

if (node == NULL)

return 0;

if ( height ( node->left( ) ) > height( node->right( ) ) )

return ( 1 + height ( node->left( ) ) );

else

return ( 1 + height ( node->right( ) ) );

}

5.8) int NumberOfNodes( BinNode<int>\* node )

{

if ( node == NULL)

return 0;

if ( node->isLeaf( ) )

return 1;

return ( 1 + count( node->left( ) ) + count( node->right( ) ) );

}

5.21) Tree and codes below, average code length is 3.23445

|  |  |
| --- | --- |
| A | 1100100 |
| B | 1100101 |
| C | 110011 |
| D | 11000 |
| E | 1000 |
| F | 1001 |
| G | 1101 |
| H | 010 |
| I | 011 |
| J | 101 |
| K | 111 |
| L | 00 |

