CSCE 221 Cover Page

Homework Assignment #3

First Name Peng Last Name Li UIN 822003660

User Name abc E-mail address billlipeng@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)		Stack Overflow		
Printed material	CSCE 221 Slides			
Other Sources				

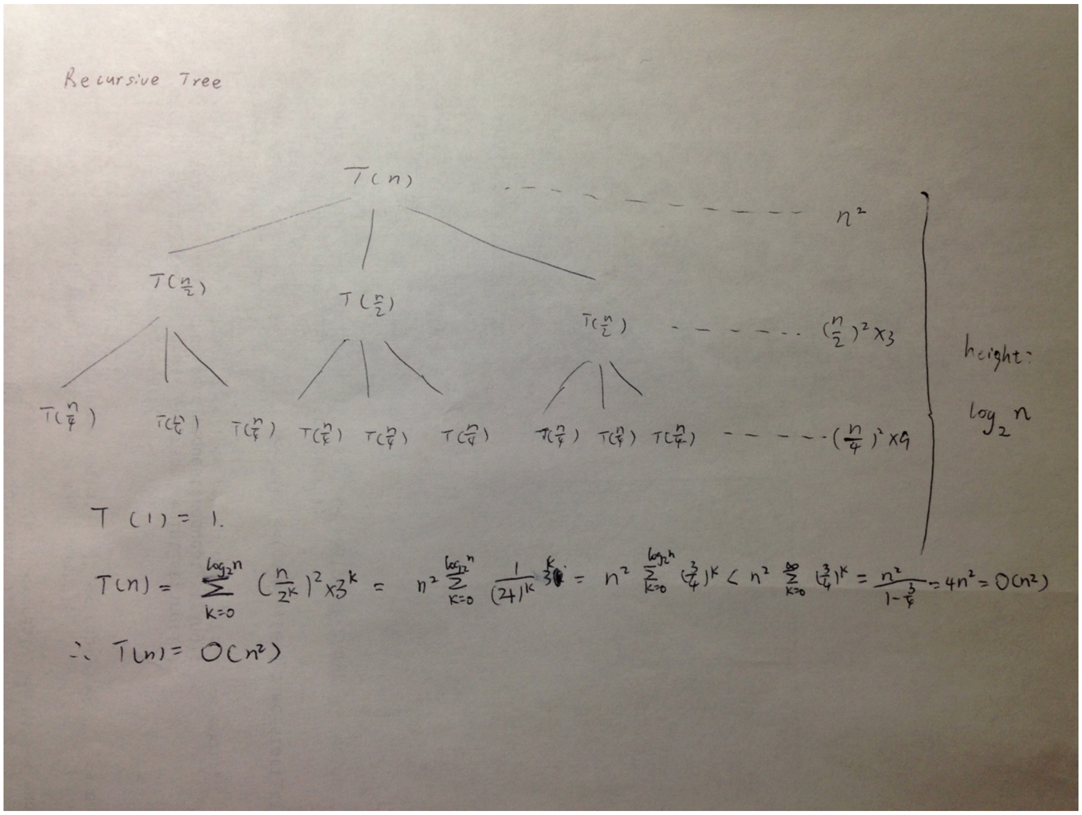
I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Date Oct 29, 2014

Homework 3
due October 29 at 11:59 pm.

1. (15 points) Use a recursive tree and the Master theorem to classify an algorithm described by the following recurrence relation: T (n) = 3T (n/2) + n^2 with the initial condition T (1) = 1.

a. Recursive Tree.



b. Master Theorem

T (n) = 3T (n/2) + n^2 ; Master theorem: T(n) = aT(n/b) + f (n)

Therefore, a = 3, b = 2, f(n) = n^2.

$\therefore n^{\log_b a} = n^{\log_2 3} < n^2$ && $af\left(\frac{n}{b}\right) = 3\left(\frac{n}{2}\right)^2 = 0.75n^2 \leq cf(n) = cn^2$ holds for some constant c = 0.8 < 1 and all sufficiently large n.

So $T(n) = \Theta(f(n)) = \Theta(n^2) = O(n)$.

2. (15 points) Provide a recursive algorithm for evaluating an algebraic expression represented as a binary tree.

```
double calculate(Node* n){
    if(n->left=NULL && n->right=NULL)//tree leaf stores
operand
        return n->token.value();
    double x=calculate(n->left);
    double y=calculate(n->right);
    if(n->token == operator)
        return x n->token.operator() y;
}
```

3. (40 points) R-7.16 p. 311
Answer the following questions for **both an extended binary tree and a proper binary tree** in (a) and (b), use the Proposition 7.10 to justify your answers.

(a) What is the minimum number of external nodes for both binary trees with height h? Prove your answer using induction.

Extended binary tree: minimum h+1 external nodes.
Proof:
For n = 1, which means the height of an extended binary tree is 0, then $n = 1 \geq h+1=0+1 = 1$, proved.
Suppose when the height of extended binary tree is h-1, $n_E \geq h$ is correct.
When the height is h, suppose in height = h-1 case root has a left sub-tree and right sub-tree. If the height of left sub-tree is h-2, then in order to get the minimum external nodes, the right sub-tree should have no internal node but only one external node. So when the height of the tree increased by 1 an internal node should be added to the minimum left leaf, thus n_E should add 1. On the left side, it is h + 1. Thus, $n_E \geq h + 1$.
From height = h-1 we conclude height = h case is correct, thus we can conclude for all $n \geq 1$, the extended binary tree has minimum h+1 external nodes.

Proper binary tree: minimum h+1 external nodes.
Proof:
For n = 1, which means the height of a proper binary tree is 0, then $n = 1 \geq h+1=0+1 = 1$, proved.
Suppose when the height of proper binary tree is h-1, $n_E \geq h$ is correct.
For the height of h, suppose in height = h-1 case root has a left sub-tree and right sub-tree. If the height of left sub-tree is h-2, then in order to get the minimum external nodes, the right sub-tree should have no internal node but only one external node. So when the height of the tree increased by 1 an internal node should be added to the minimum left leaf, thus n_E should add 1. On the left side, it is h + 1. Thus, $n_E \geq h + 1$.
From height = h-1 we conclude height = h case is correct, thus we can conclude for all $n \geq 1$, the proper binary tree has minimum h+1 external nodes.

(b) What is the maximum number of external nodes for both binary trees with height h? Prove your answer using induction.

Extended binary tree: maximum 2^h external nodes.
For n = 1, which means the height of a extended binary tree is 0, then $2^h = 2^0 = 1 \geq 1 = n$, proved.
Suppose when height of extended binary tree is h-1, $n_E \leq 2^{h-1}$ is correct.
To prove the height = h is correct, first let's look at height = h-1 case. An extended binary tree with height of h-1 can only get 2^{h-1} external nodes if and only if it is a full tree. Thus, all the nodes at the height of h-1 are external nodes. When the height increased by 1, we can conclude each external node in height h-1 case will grow two external nodes. Thus, $n_E = 2^{h-1} * 2 = 2^h$. Thus, $n_E \leq 2^h$.
From height = h-1 we conclude height = h case is correct, thus we can conclude for all $n \geq 1$, the extended binary tree has maximum 2^h external nodes.

Proper binary tree: maximum 2^h external nodes.
For n = 1, which means the height of a proper binary tree is 0, then $2^h = 2^0 = 1 \geq 1 = n$, proved.
Suppose when height of proper binary tree is h-1, $n_E \leq 2^{h-1}$ is correct.
To prove the height = h is correct, first let's look at height = h-1 case. An proper binary tree with height of h-1 can only get 2^{h-1} external nodes if and only if it is a full tree. Thus, all the

nodes at the height of h-1 are external nodes. When the height increased by 1, we can conclude each external node in height h-1 case will grow two external nodes. Thus, $n_E = 2^{h-1} * 2 = 2^h$. Thus, $n_E \leq 2^h$.
From height = h-1 we conclude height = h case is correct, thus we can conclude for all $n \geq 1$, the proper binary tree has maximum 2^h external nodes.

(c) Let T be a proper binary tree with height h and n nodes. Prove that

$$\log(n + 1) - 1 \leq h \leq (n - 1)/2$$

From proposition 7.10.1, we know
$2h+1 \leq n \leq 2^{h+1} - 1$
Thus,
$n \leq 2^{h+1} - 1$ and $2h+1 \leq n$
$n + 1 \leq 2^{h+1}$ and $2h \leq (n - 1)$
$\log(n+1) \leq h+1$ and $h \leq (n - 1)/2$
$\log(n+1) - 1 \leq h$ and $h \leq (n - 1)/2$
Hence,
$\log(n + 1) - 1 \leq h \leq (n - 1)/2$

Proposition 7.10 (pg 287): Let T be a nonempty binary tree, and let n, n_E, n_I and h denote the number of nodes, number of external nodes, number of internal nodes, and height of T, respectively. Then T has the following properties:
1. $h+1 \leq n \leq 2^{h+1} - 1$
2. $1 \leq n_E \leq 2^h$
3. $h \leq n_I \leq 2^h - 1$
4. $\log(n+1) - 1 \leq h \leq n - 1$
Also, if T is proper, then it has the following properties:
1. $2h+1 \leq n \leq 2^{h+1} - 1$
2. $h + 1 \leq n_E \leq 2^h$
3. $h \leq n_I \leq 2^h - 1$
4. $\log(n+1) - 1 \leq h \leq (n - 1)/2$

4. (15 points) R-7.22 p. 312
Describe, in pseudo-code, an algorithm for computing the number of descendants (see the definition in the textbook on p. 270) of each node of a binary tree. The algorithm should be based on the Euler tour traversal.

```
int EulerTour(Node *node){
    int left = 0;
    int right = 0;
    if (node == T.root) { // to avoid calculate the root
        if (node->left != NULL)
            left = EulerTour(node->left);
        if (node->right != NULL)
            right = EulerTour(node->right);
        return left + right;
    }
    else{
        if (node->left != NULL)
            left = EulerTour(node->left);
        if (node->right != NULL)
            right = EulerTour(node->right);
        return left+right+1;
    }
  }
```

5. (15 points) C-7.33 p. 317
Describe, in pseudo-code, a non-recursive method for performing an in-order traversal of a binary tree in linear time. (Hint: Use a stack.)

```
void In_order_traversal(Tree t) {
    Stack s = new Stack();
    Node* node = t.root();
    // go to the far left first
    while (node != NULL){
        s.push(node);
        node = node -> left;
    }
    // pop process and jump to the right
    while (s.size() > 0)
    {
        node = (Node) s.pop();
        process(node);
        if (node -> right != NULL){
            node = node -> right;
            while (node != NULL){
                s.push(node);
                node = node -> left;
            }
        }
    }
  }
```