|  |  |  |
| --- | --- | --- |
| **ECS 60** | **Written Homework #3 Key: Trees (26 points)** | **Winter 2016** |

Due Friday, October 23rd, 4:00 pm in 2131 Kemper

1. (5 points, 1 point each) For the following M = 3 and L = 3 BTree as the starting point for each part, draw the BTree that would result after the specified insertion. Sean’s Rule for Insertion: Give left, give right, else split.

41 81

11 21 51 61 91

1 11 21 41 51 61 81 91

3 13 23 43 53 63 83 93

15 25 45 55 85

* 1. 62

41 81

11 21 51 61 91

1 11 21 41 51 61 81 91

3 13 23 43 53 62 83 93

15 25 45 55 63 85

* 1. 12

41 83

12 21 51 61 91

1 12 21 41 51 61 83 91

3 13 23 43 53 63 84 93

11 15 25 45 55 81 85

* 1. 24

41 --

21 -- 81 --

11 -- 23 -- 51 61 91 --

1 11 21 23 41 51 61 81 91

3 13 22 24 43 53 63 83 93

15 25 45 55 85

* 1. 52

41 81

11 21 51 55 91

1 11 21 41 51 55 81 91

3 13 23 43 52 61 83 93

15 25 45 53 63 85

1. (5 points, 1 point each) For the following M = 3 and L = 3 BTree as the starting point for each part, draw the BTree that would result after the specified series of deletion. Sean’s Rule for Deletion: Borrow left, merge left, borrow right, merge right.

41 --

11 21 51 --

1 11 21 41 51

3 13 23 43 53

25

* 1. 41

25 --

11 23 51 --

1 11 21 25 51

3 13 23 43 53

* 1. 25

41 --

11 21 51 --

1 11 21 41 51

3 13 23 43 53

* 1. 53

21 --

11 -- 41 --

1 11 21 41

3 13 23 43

25 51

* 1. 11, 51

21 41

1 21 41

3 23 43

13 25 53

1. (1 point) Draw a binary search tree with only the elements {1, 2, 3, 4, 5, 6} in it, for which the postorder and inorder traversals generate the same sequence.

**6**

**5**

**4**

**3**

**2**

**1**

1. (4 points, 1 point each) Using the following binary search tree as the starting point for each part, draw the tree that results after the given operation.

50

30 62

20 43 58 73

10 54 60 65

52 56

* 1. Assuming it is an AVL tree, delete 43.

58

50 62

20 54 60 73

10 30 52 56 65

* 1. Assuming it is an AVL tree, insert 55

50

30 62

20 43 56 73

10 54 58 65

52 55 60

* 1. Assuming it is a splay tree, delete 54.

56

50 58

30 52 56 62

20 43 60 73

10 65

* 1. Assuming it is a splay tree, insert 5.

5

30

10 50

20 43 62

58 73

54 60 65

52 56

1. (2 points) Construct a Huffman trie based on the following distribution of letters. When combining two trees, always place the smaller on the left.

A

BB

CCCC

DDDDD

EEEEEE

FFFFFFFF

GGGGGGGGG

HHHHHHHHHH

JJJJJJJJJJJJJ

KKKKKKKKKKKKKK

LLLLLLLLLLLLLLLL

MMMMMMMMMMMMMMMMM

**t11**

**t9 t10**

**t5 t6 t7 t8**

**G H t3 J K t4 L M**

**D E t2 F**

**t1 C**

**A B**

1. (2 points) Show that for any leaf *v* in a binary search tree, if *u* is the parent of *v*, then either key[*v*] is the largest key in the tree smaller than key[*u*], or key[*u*] is the smallest key in the tree larger than key[*v*]. (Heileman, p.216)

**The proof is done by contradiction. There are two cases to explore: *v* is the left child of *u*, or *v* is the right child of *u*.**

**Case 1: *v* is the left child of *u*.**

**Since *v* is the left child of *u*, then key[*v*] ≤ key[*u*]. If key[*v*] was not the largest key in the tree smaller than key[*u*]** **then there must be a node *w* such that key[*v*] < key[*w*] ≤ key[*u*]. If such a node *w* existed, then it must be the right child of *v*. But that contradicts the assertion that *v* is a leaf. So there must be no such *w*, and therefore key[*v*] is the largest key in the tree smaller than key[*u*].**

**Case 2: *v* is the right child of *u*.**

**Since *v* is the right child of *u*, then key[*v*] ≥ key[*u*]. If key[*v*] was not the smallest key in the tree larger than key[*u*]** **then there must be a node *w* such that key[*v*] > key[*w*] ≥ key[*u*]. If such a node *w* existed, then it must be the left child of *v*. But that contradicts the assertion that *v* is a leaf. So there must be no such *w*, and therefore key[*v*] is the smallest key in the tree larger than key[*u*].**

1. (2 points) Prove that the maximum number of nodes in a binary tree of height *h* is 2*h* + 1 – 1.

**Base case: The theorem is trivially true for *h* 0; 20 + 1 – 1 = 1.**

**Inductive hypothesis: Maximum number of nodes of height k is 2*k* +1 –1.**

**Inductive conclusion: Maximum number of nodes of height k + 1 is 2*(k* +1) + 1 –1.**

**A tree of height *k* +1 can have two subtrees of height at most *k* .**

**Each of these subtrees can have at most 2*k* +1 -1 nodes each by the inductive hypothesis.**

**The whole tree (including root) will have a maximum number of nodes of**

**1 + 2 \* (2*k* +1 –1)**

**= 2*k* +2 –1 which proves the theorem for height *k* +1 and hence for all heights.**

1. (1 point) In terms of *M* and *L*, what is the maximum height of a BTree with *n* elements?

***Height =***

Sources of questions:

Gregory L. Heileman, *Data Structures, Algorithms, and Object Oriented Programming*, New York, NY, McGraw-Hill, 1996.