Century College

CSCI 2082.01

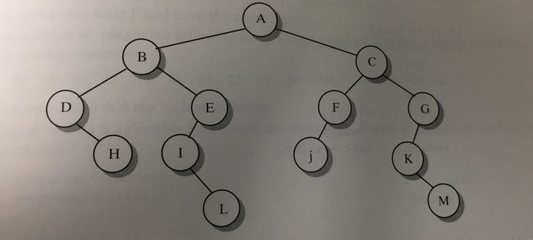
Fall 2016

## PA5

## Due on 11/29 @ 11:30PM

(25 Points)

## Question 1 (5 Points)



1. What is the parent of node L? I
2. What is the depth of this tree? 4 (Root is at depth of 0)
3. List the nodes in subtree B. B D E H I L (Sub-root B plus all of its descendant nodes)
4. List all of the leaf nodes. H L J M (Leaf nodes have zero children nodes)
5. Give the order in which nodes are visited in a postorder traversal of the tree.

H D L I E B J F M K G C A

Answer:

## Question 2 (2 Points)

1. A binary tree with all possible nodes has a depth of 8. How many nodes does it have? 511
2. A complete binary tree has 5125 nodes. What is its depth?

12

**Answer:**

Max number of total nodes = ( 2n+1 - 1)

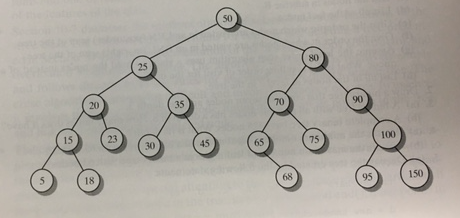
(28+1 – 1) = 511

(211+1 – 1) = 4095

(212+1 – 1) = 8191

Since its greater than a depth of 11 and less than the max number of a depth of 12 it has to have at least of depth of 12 to be able to reach 5125 nodes.

## Question 3 (5 Points)



1. If the value 33 is inserted into the tree, which node is its parent? 30
2. If the value 64 is inserted into the tree, which node is its parent? 65

**Answer:**

## Question 4 (5 Points)

Implement the body of the following method using a binary search of the array. You do not need to check the precondition.

public static boolean has33(int[ ] data, int start, int end)

// Precondition: The elements data[start]...data[end] are sorted from //smallest to largest. This array segment might be empty (indicated by end //being less than start).

// Postcondition: A true return value indicates that the number 33 appears in

// data[start]...data[end]. A false return value indicates that 33 doesn’t

// appear.

**Answer:**

**public** **class** **Search** {

**public** **static** **boolean** **has33**(**int**[] data, **int** start, **int** end) {

**int** **target** = 33;

**if** (start >= end) {

**return** **false**;

}

**int** **middle** = ((start + end) / 2);

**if** ((data[middle]) == target) {

**return** **true**;

} **else** **if** (data[middle] < target) {

**return** (*has33*(data, middle + 1, end));

} **else** {

**return** (*has33*(data, start, middle));

}

}

**public** **static** **void** **main**(**String**[] args) {

**int**[] **array1** = { 0, 4, 7, 18, 35, 51 };

**int**[] **array2** = { 0, 3, 7, 31, 33, 45 };

**System**.***out***.println("Array 1" + " contains the value 33 " + *has33*(array1, 0, array1.length));

**System**.***out***.println("Array 2" + " contains the value 33 " + *has33*(array2, 0, array2.length));

}

}

## Question 5 (5 Points)

Draw a hash table with chaining and a size of 9. Use the hash function "k%9" to insert the keys 5, 29, 20, 0, and 18 into your table.

**Answer:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 0 | 29 | 20 | 18 |  |  |  |  |  |

## Question 6 (3 Points)

Here is an array of ten integers:

5 3 8 9 1 7 0 2 6 4

Draw this array after the FIRST iteration of the large loop in an insertion sort (sorting from smallest to largest). This iteration has shifted at least one item in the array!

**Answer:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | **5** | **8** | **9** | **1** | **7** | **0** | **2** | **6** | **4** |

## What to turn in:

* This document with your answers in Green color.