# Data Structures and Algorithms

# INFO 6205

# Homework 7

# Due: October 31, 2020

1. Provide descriptions for these:

What is the Balanced Tree, Complete Tree and Non-Complete Tree?

How does JVM manages References and Data

Why Java is PassByValue, how does it work?

Why would you consider BTree?

2. Consider this diagram with added code: Explain and Redraw the digram after code changes:

String s3=s2;

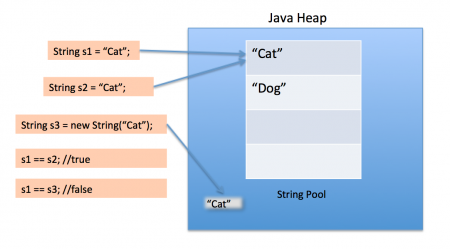
String s4=s1;

String s5= new String(“Cat”);

String s6=new String(“Dog”);

String s7=s5;

String s8=s6;



3. Consider the following string: “ABDCEDDFFCABEEECCEFDDAAF”.

a) Use key-indexed counting sort algorithm to sort the string. Show each step

and results.

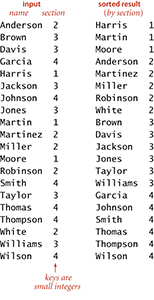
b) What is the running time complexity of the algorithm as compared to Insertion sort?

c) Write the java code to sort the string using steps described in (a).

4. Write Java program that generates random text string with a length of 600 bytes for 300,000 iterations. For each iteration, reverse the string using: a) String operations, and b) StringBuilder operation(s). Then c) What is the running time complexity of (a) and (b) after all iterations?

d) Present your results in (c) as a graph showing the running times.

5. Consider the following data to maintain grades for a class with students assigned to sections numbered 1, 2, 3, and so forth. Sort the data with key-indexed-counting to list the class by section.



6. Consider following data: {3,7,9,23,45,1,5,14,55,24,13,11,8,19,4,31,35,56}

a) Construct Binary Tree

b) Construct 2-3 Tree

c) Construct 2-3-4 Tree

d) What is Time complexity of each case, Why would you use one versus the other?

e) Insert 17, 36, 32 in (b)

f) Delete 13 in (a) and (b)

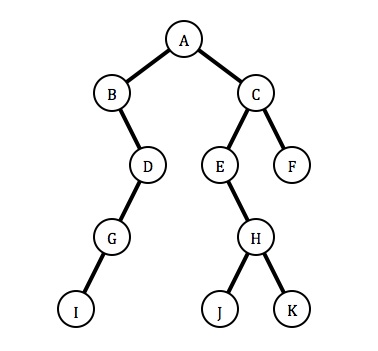
g) What is the Height of (a), (b), (c)?

h) Write Java Search and Insert code for (a) and (b)

i) Write Java code for DeleteMin() and DeleteMax Algorithms for (a),

provide example

7. Consider the following Binary tree, write Java code to find **minimum** element in binary search tree. You may write either a recursive or iterative implementation.



8. Insert the following items into an empty binary search tree in order:

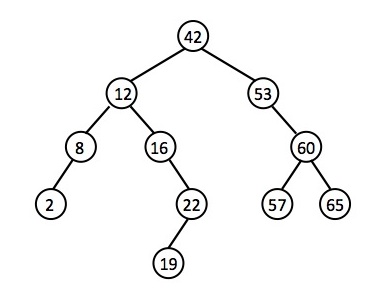
{30, 40, 23, 27, 48, 26, 12, 31}

a) Insert the following items into an empty binary search tree in order:

b) What is the maximum height of a binary search tree? why?

c) What is the time complexity of the Tree? why?

9. Consider the following binary tree:



Use BST functions deleteKey, MaxValue: Show step-by-step code logic

to delete the following nodes and redraw the binary tree for each deletion:

Delete 57

Delete 8

Delete 19

10. Consider data: {3,7,9,23,45,1,5,14,5,24,13,11,8,19,4,31,35,56,17,29,6,22}

a) Construct a B-Tree order of t=5

b) Delete elements 45, 11, 31, Construct the new BTree, and Describe

the delete algorithm step(s) for each element.

c) Write Java code for (a) and (b)

d) Perform Inorder, Postorder, Preorder Traversals on B-Tree

e) Class Record is described below. Write Java code to build the B-Tree

you constructed in (a)

public class Record {

private int key

private Node leftNode;

private Node rightNode;

public Record(int key, Node leftNode, Node rightNode) {

this.key = key;

this.leftNode = leftNode;

this.rightNode = rightNode;

}

public Record(int key){

this.key = key;

}

public int getKey() {

return key;

}

public Node getLeftNode() {

return leftNode;

}

public Node getRightNode() {

return rightNode;

}

public void setKey(int key) {

this.key = key;

}

public void setLeftNode(Node leftNode) {

this.leftNode = leftNode;

}

public void setRightNode(Node rightNode) {

this.rightNode = rightNode;

}

}

11. Consider Data: {9 30 -28 -25 -1 -3 6 12 15 20 25 26 33 35 39}

a) Build a btree with minimum degree of 4

b) What is the order of this btree?

c) Why is this Btree with minimum degree of 4?

d) Insert 27 into btree

e) Delete 20, -1, 35

12. Consider a disk with a sector size of 512 bytes, 1,000 tracks per surface, 100 sectors per track, five double-sided platters and a block/page size of 1024 bytes. Suppose that the average seek time is 5 msec, the average rotational delay is 5 msec, and the transfer rate is 100 MB per second. Suppose that a file containing 1,000,000 records of 100 bytes each is to be stored on such a disk and that no record is allowed to span two blocks.

a) How many records fit onto a block?

b) How many blocks are required to store the entire file?

13. Consider the following sequence of letters:

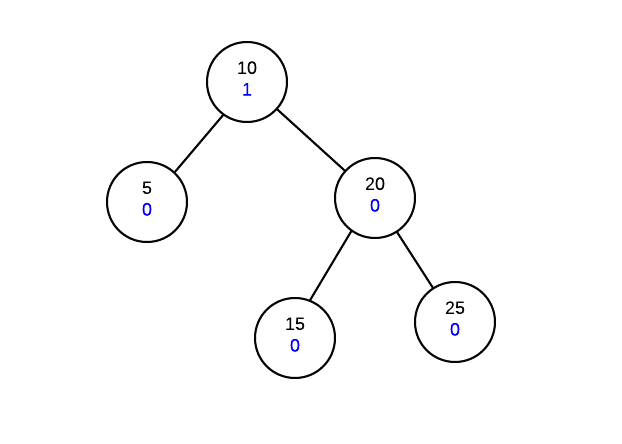
‘A’G’F’B’K’D’H’M'J'E'S'I'R'X'C'L'N'T'U'P'

a) Build BTree with order of t=5

b) What is minimum degree for this BTree?

c) Discuss height, time and space complexity

14. Consider the following AVL Tree:



AVL trees work by ensuring that the tree is height balanced after an operation. Each node

must not only maintain its data and children information, but also a height balance value.

The height balance of a node is calculated as follows:

Height-balance-of-node = height of right-subtree - height of left-subtree

The above formula means that if the right-subtree is taller, the height balance of the node will be positive. If the left-subtree is taller, the balance of the node will be negative.

Show AVL Tree after performing these insert operations:

Insert 30

Insert 27

Insert 35

The height of AVL Tree is not balanced after insertions. You would need to do several

rotations, single-rotation and double-rotation to make tree balanced.

A) Show all rotations to make tree balanced

B) Write AVL Tree Algorithm

C) Write Java code for algorithm and Run

D) Consider the data structure “Record’ in problem10, what change do you need

to make to the data structure for AVL Tree?

15. Does the following code for Thread synchronization works correctly?

Why or why not? If not, how do you fix it?

class Table{

synchronized void printTable(int n){//synchronized method

for(int i=1;i<=5;i++){

System.out.println(n\*i);

try{

Thread.sleep(400);

}catch(Exception e){System.out.println(e);}

}

}

}

class MyThread1 extends Thread{

Table t;

MyThread1(Table t){

this.t=t;

}

public void run(){

t.printTable(5);

}

}

class MyThread2 extends Thread{

Table t;

MyThread2(Table t){

this.t=t;

}

public void run(){

t.printTable(100);

} }

public class TestSynchronization2{

public static void main(String args[]){

Table obj1 = new Table();//only one object

MyThread1 t1=new MyThread1(obj1);

MyThread2 t2=new MyThread2(obj1);

t1.start(); t2.start(); }}