# Data Structures and Algorithms

# INFO 6205, Sat

# Homework 6

# Due: October 20, 2019

Put all your java, compiled class files and documentation files into a zip file named Homework6.zip and submit it via the drop box on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

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

2. 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) Construct Binary Heap Tree

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

f) Insert 17, 22, 32 in (b)

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

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

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

j) Write Java code for DeleteMin() and DeleteMax Algorithms for (a), provide example

3. Consider the following string: “ABDCEDDFCABBEECCEFDDAAF”.

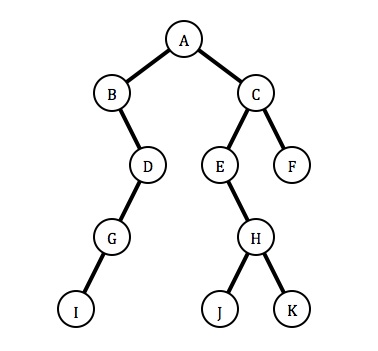
a) Use key-indexed counting sort algorithm to sort the string. Show each step

and the results.

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

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

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



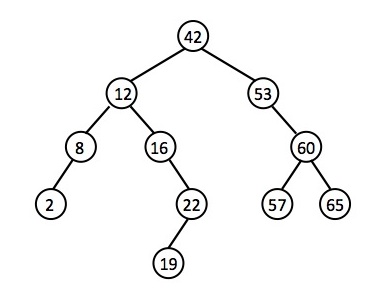
5. Insert the following items into an empty binary search tree in order:

{30, 40, 23, 58, 48, 26, 11, 13}

6. What is the maximum height of a binary search tree in problem-5? Why?

What is the time complexity of the Tree you built in problem-5? Why?

7. Consider the following binary tree:



Use BST functions deleteKey, deleteRec, and minValue discussed in class: Show step-by-step code logic to delete the following nodes and redraw the binary tree for each deletion:

Delete 65

Delete 60

Delete 42

8. Consider arr[] = {15, 80, 20, 90, 40, 60, 70}. Walk through InsertionSort, SelectionSort, TimSort, and QuickSort algorithms, show step by step how the sort works.

9. Consider attached image Boston.jpg. Write a program to sort the image Pixels by “brightness”. Use four sorting algorithms: InsertionSort, SelectionSort, TimSort, and QuickSort. You need to sort the Pixel array size of the image in Descending order and compare and show the runtime time complexity of each Sorting algorithm.

Notes:

You may NOT use any Java library function for sorting. You should use ONLY the Sorting

Java code I provided in class. The Pixel sorting should start from (0,0) to (high,high)

for Brightness. For each Pixel, you need to convert RGB color to appropriate intensity.

Use intensity formula: I = 0.2989R + 0.5870G + 0.1140B. If the current pixel Intensity

is larger than the next pixel intensity, you need to swap, going in descending order.