# Data Structures and Algorithms

# INFO 6205

# Homework 4

# Due: June 7 2020

Put all your java, compiled class files and documentation files into a zip file named Homework4.zip and submit it via dropbox to blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

References

<https://algs4.cs.princeton.edu/lectures/>

<https://introcs.cs.princeton.edu/java/23recursion/>

1. The Recursive operations for Factorial and Fibonacci sequence was discussed in class.

A) For factorial 5!a) Show recursive stack operations, provide details step-by-step, b**)** Walk through your stack operations and provide the result. c**)** Write Java code with input factorial 5! d**)** Compile and run your program, what is the running time of your algorithm?

B) For Fibonacci sequence with n=5, a) Show recursive stack operations, provide details step-by-step, b**)** Walk through your stack operations, provide the result. c) Provide Iterative algorithm for Fibonacci function, d**)** Write Java code for both recursive and iterative algorithms, e**)** Compile and Run your program.

C) For Towers of Hanoi problem with n=7 discs, how does the algorithm work? What data structures would you use? provide step by step operations. Write Java code, compile and run program.

2. An *n*-bit Gray code is a list of the 2*n* different *n*-bit binary numbers such that each entry in the list differs in precisely one bit from its predecessor. The *n* bit binary reflected Gray code is defined recursively. How does algorithm works for n=4, describe step-by-step. Write Java code, compile and run program.

3. Write a recursive method countBinary that has one integer parameter n and returns the number of binary strings of length n that do not have two consecutive 0's. For example, for n = 4, the number of binary strings of length 4 that do not contain two consecutive 1's is 2: 1111, 1110, 1101, 1011, 1010, 0111, 0110, 0101

4. Consider the following MergeSort algorithm with Input Data:

{11, 27, 43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}

MergeSort(arr[], l, r)

If r > l

1.Find the middle point to divide the array into

two halves: middle m = (l+r)/2

2.Call mergeSort for first half:

Call mergeSort(arr, l, m)

3. Call mergeSort for second half:

Call mergeSort(arr, m+1, r)

4.Merge the two halves sorted in step 2 and 3:

Call merge(arr, l, m, r)

A) Sort data Graphically, show step-by-step, recursion on Stack

and what is Termination point?

B) Write Java code and Compile and Run with provided data.

5. Consider the following code for HeapSort:

<https://www.studytonight.com/data-structures/heap-sort>

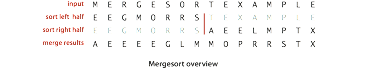
A) Compile and run this program with data = {11, 27, 43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}

B) Is the output result is same as your program in problem-4?

C) Change the program to sort in reverse direction, either Ascending or Descending

6. Consider attached image Boston.jpg. Write a program to sort the image Pixels by “brightness”. Write program for four sorting algorithms: HeapSort and MergeSort. You need to sort the Pixel array size of the image in descending order and show the runtime time complexity of each Sorting algorithm and compare.

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 ascending order.

Note: You may need to use these classes:

Java.awt.image.BufferedImage: image class.

eg: image = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_ARGB);

java.util.\*: collection of List data types.

javax.imageio.ImageIO: for reading/writing images to file

7. Consider mergeSort algorithm for the following input string. Show the stack operations push and pop step by step for call mergeSort(arr, l, m) and call mergeSort(arr, m+1, r). Note: I don’t need the entire program, just show step by step stack push and pop operations, the recursive Tree structure