# Data Structures and Algorithms

# INFO 6205

# Homework6

# Due: March 23, 2022

Put all your java, compiled class files and documentation files into a zip file Homework6.zip

and submit it via Canvas before the END of due date. Put your name on all .java files.

1. Consider the following MergeSort recursive Algorithm with 4-steps and termination-point

r > 1 and MergeSort and Merge operations:

A) Provide a diagram that shows the entire Stack operations. You start with step1 and then

step2 to handle the first-half of data and then second-half, at what point step3 starts?

At what point step4 starts? Note: This is recursive call at each layer

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)

B) Consider the following diagram. How many merges are required, how and why?



C) Consider the following input Data: {43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}

a) Merge-sort the data Graphically, show step-by-step recursion on Stack, and

what is Termination point?

b) Write Java code, Compile and Run with provided data

c) What is the run-time and space complexity of merge-sort?

2. Consider HeapSort algorithm with input array data = {43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}

I graphically showed step-by-step heap-sort algorithm in Ascending order with

two operations: MaxHeapify and Replacing first-element with last-element

a) Graphically show step-by-step algorithm to sort data in Descending-order with two

operations: MinHeapify and Replace one element

b) Write Java code for the algorithm with input data to sort in Descending order

c) Change the program in (b) to sort in Ascending order

d) Explain output results for programs (b) and (c)

e) What is the run-time and space complexity of heap-sort?

3. Consider the following quick-sort Partition algorithm. Apply partition algorithm to data with pivot on first-element: {43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}; Show step-by-step

Algorithm:

i=0; j=0;

if ( a[j] < pivot)

i++;

swap (a[i], a[j]);

if ( a[j] > pivot)

do-nothing

j++;

4. Consider a[] = {43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}, Show QuickSort algorithm with Pivot on last element: a) show step by step how the sorting works, b) Write Java code, compile and run,

c) Time and space complexity, d) compare the results of QuickSort with results of MergeSort (problem-1-part-c) and HeapSort (problem-2)

5. Consider a[] = {43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}. Show InsertionSort algorithm: a) show step by step how the sorting works, b) Time and space complexity, c) compare the results of InsertionSort with results of MergeSort problem-1-part-c and HeapSort problem-2

6. Consider TimSort algorithm with input array data = {43, 38, 3, 9, 82, 10, 21, 8, 34, 19, 6}

a) Show step-by-step Algorithm to sort data in Ascending order with MinRun=32

b) Tim-sort uses the best of insertion-sort and merge-sort algorithms. Explain how?

c) What are time and space complexity of Tim-sort?

7. Consider mergeSort algorithm for the following input string. Show step-by-step stack operations push and pop 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

8. Table shows the performance of different Sorting algorithms for best, average, and worst cases. In each case, a) which algorithm performs worst and which algorithm performs best in terms of time-complexity and space-complexity? b) Explain best, average, worst complexity of BubbleSort? Explain bubble-sort algorithm?, is it possible algorithm iterations repeat over and over again on input array with no change to sorting elements? How do you explain that? `

Note: you need to read the article entirely