CS2223: Algorithms

Assignment 2

Teams: To be done individually

Release date: 03/25/2017

Due date: 04/01/2017 (11:59 PM)

Submission: Electronic submission only

****Note: No late submission of Assignment 2 will be accepted because the solution will be posted immediately after the due date to be available before midterm I.**

General Instructions

- ***Executable vs. Pseudocode:*** Each question will explicitly state whether the deliverable is pseudocode or an executable program that the TA will run to give you a grade.

- ***Programming Language:*** If a question asking for an executable program, then choose a language from Java, C, C++, or Python. ***You must make it clear in your report:***
  - How to compile your program
  - How to execute it and with which arguments

- ***Submissions:*** The submission of Assignment I must be done electronically through blackboard system. All programs plus your report (.doc or .pdf) should be zipped into a single file and that is the file to submit.

Question 1 (Implementing a Sorting Technique) [14 Points]

We learned a couple of sorting algorithms over the last two weeks. Your task in this problem is to implement one of the most efficient algorithms, namely *quick sort*.

//Hint: When selecting a pivot, use the hint given in the slides to avoid the worst case scenario

Deliverables of Question 1

(1) A single executable program implementing quick sort.

(2) At the beginning, the program should create an array of 100 values randomly selected from the range of [1..1000]. That will be the array to be sorted.

(3) The output of the algorithm is a file containing several lines. Each line will list the current state of the 100 values (comma separated). After each iteration of the algorithm, one line should be added to the output file. Certainly, the last line should have all values completely sorted.

Question 2 (Recurrence Relations) [20 Points]

For each of the following recurrences, derive its Big O complexity. For 3) & 5) below use the Tree-Based method (Section 4.4 in the Textbook), while for 1), 2) & 4) use the Master Theorem (Section 4.5).

1) $T(n) = T(n/2) + O(1)$
   ** Also mention an algorithm we took in class or HW1 that closely follow this recurrence.

2) $T(n) = 2T(n/2) + n$
   ** Also mention an algorithm we took in class or HW1 that closely follow this recurrence.

3) $T(n) = 4T(n/2) + n$

4) $T(n) = 7T(n/2) + n^2$

5) $T(n) = T(n/3) + \text{Log}_3\, n$

Question 3 (Recurrence Relations) [6 Points]

For each of the following sentences, state whether it is True or False. And given an explanation for your answer.

1) $T(n) = 2T(n/2) + n$ implies that $T(n) = \Theta$ (worst case of mergesort)

2) $T(n) = 4T(n/2) + n$ implies that $T(n) = \Omega\ (n^2 \log n)$

3) Worst case of quick sort (Say F()) is upper bound (i.e., Big O) for $T(n) = T(n-1) + \text{Log } n$
   That is: $T(n) = O(F())$

Question 4 (Analyze Algorithm) [10 Points]

Assume we have the following sorting algorithm:

To sort an array of size N (A[1…N]), the algorithm will do the following:

a- Recursively, Sort the first N-1 elements A[1…N-1]

b- Use binary search to find the correct place of A[N] to add it to the sorted list. After finding the correct place, it will need to shift the values to make place for A[N].

1) [5 Points] Write the detailed recurrence equation for this algorithm (do not omit any terms).

2) [5 Points] Simplify the recurrence equation by throwing away terms that are dominated by others. And then use **either of** the Master Theorem or the Tree-Based method to compute the complexity of this algorithm.

Question 5 (Heap Structure) [10 Points]

(a) [8 Points] Show Step by step construction of the MaxHeap tree for the following sequence of inserted values:

10, 30, 5, 11, 7, 22, 25, 31, 40, 16

Show the final heap tree after each insertion (you do not need to show the swaps).

(b) [2 Points] Build the Heap array corresponding to the tree that you built in the previous step.