CS2223: Algorithms
D-Term, 2015

Assignment 3

Teams: To be done individually

Due date: 04/16/2015 (1:50 PM)

Submission: Electronic submission only

General Instructions

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

- ***Programming Language:*** If a question asks you to write a program, then use Python language.

- ***Submissions:*** The submission of Homework 3 must be done electronically through the blackboard site for CS2223 available from myWPI. Login to myWPI, go to CS2223 under "My Courses", then go to "Assignments", and submit your homework under "HW3". All programs plus your report (.doc or .pdf) should be zipped into a single file and that is the file to submit.

Question 1 (Recurrence) [20 Points = 5 Points x 4]

For each of the following recurrences, use either the Recursion-Tree method (Section 4.4 in the Textbook), or the Master Theorem (Section 4.5) to solve it and produce the Big-O complexity of the recurrence.

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

2) $T(n) = 2T(n/2) + O(n)$
   ** Also mention a sorting algorithm we taught in class that closely follow this recurrence.

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

4) $T(n) = 7T(n/2) + O(n^3)$

Question 2 (Recurrence & Sorting 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 last N-1 elements: A[2…N]
   b - Use binary search to find the correct place of A[1] to add it to the sorted list. After finding the correct place, shift the values of A[2…N] to make place for A[1].

1) [4 points] Write the recurrence equation for the running time of this algorithm.

2) [6 points] Simplify the recurrence equation by throwing away terms that are dominated by others. And then use the Recurrence Tree method to compute the complexity of this algorithm.

Question 3 (Merge Sort Algorithm) [10 Points]

Here is the pseudocode of MergeSort:

```
MergSort(A, left, right):
        if left < right:
                mid = (left+right)/2
                MergSort(A, left, mid)
                MergSort(A, mid+1, right)
                Merge(A, left, mid, right)
```

Consider the following change to the code, which only Merge the two smaller lists if A[mid] > A[mid+1].

```
MergSort(A, left, right):
        if left < right:
                mid = (left+right)/2
                MergSort(A, left, mid)
                MergSort(A, mid+1, right)
                if A[mid] > A[mid+1]:
                        Merge(A, left, mid, right)
```

Can this code still correctly sort List A? Briefly explain your answer. (using less than 5 sentences)

Question 4 (QuickSort) [10 Points]

Trace the operation of QuickSort's Partition procedure (the procedure which divides a list into two smaller list) on the following list.
Your answer should be a sequence of lists: write one list after each swap happens, including the final list after the final swap happens.
We assume the first element in the list is selected as the pivot.

LIST: 13, 2, 38, 12, 82, 7, 4, 87

Question 5 (HeapSort) [10 Points]

Given the following list, after building a Max-Heap on this list, what the heap will be like? (Show the heap as an array/list)

LIST: 13, 2, 38, 12, 82, 7, 4, 87

Question 6 [20 Points] (Binary Tree)

1) Assume a binary tree where each node has two fields, a unique ID, and a Value (positive or negative). We need to find the node X where the sum of all values in the subtree rooted at X (including X) is the maximum.

(a) [10 points] Please sketch a pseudocode for a recursive algorithm that reports the ID of node X.

(b) [5 points] Write the recurrence equation for the running time of this algorithm and state its time complexity.

2) [5 Points]
Given the following Pre-Order and In-Order traversals of a binary tree, construct the tree (draw it in your report). If you think it cannot be done, then state why.

Pre-Order: 1, 6, 5, 11, 15, 7, 8, 2, 3, 19
In-Order: 11, 5, 6, 15, 1, 2, 8, 7, 3, 19