CS2223: Algorithms
D-Term, 2013

Assignment 2

Teams: To be done individually

Release date: 03/23/2013

Due date: 03/30/2013 (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 asks you to write an executable program, then choose a language of your choice, but make it clear in your report:
  - How to compile your program
  - How to execute it and with what arguments

- ***Submissions:*** The submission of Assignment 2 must be done electronically through blackboard system. All programs plus your report (.doc, .docx, or .pdf) should be zipped into a single file and that is the file to submit.

Question 1 (Comparison of Sorting Techniques) [20 Points]

We learned a couple of sorting algorithms over the last two weeks. Your task in this problem is to implement two of the most efficient algorithms, namely the merge sort and quick sort. Also create a simple graphical interface that render the array while being sorted (That is, after several iterations of the algorithm, re-draw the content).

Hint: The interface should be very simple, e.g., map each value to a vertical line with scale corresponding to the value.

Deliverables of Question 1

(1) A single executable program written any a language of your choice. The program should contain the two sorting algorithms mentioned above. The program takes one argument with values:
-- 1 (calls the merge sort),
-- 2 (calls the quick sort),

(2) At the beginning, the program should create an array of 200 values randomly selected from the range of [1..500]. That will be the array to be sorted.

(3) The output of the algorithm is the graphical view of the array content rendered after several iterations (You can fix this in your code). Just make sure the interface will capture the progress of the algorithm.

Question 2 (Recurrence Relations) [20 Points]

For each of the following recurrences, use either the Tree-Based 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 took in class or HW1 that closely follow this recurrence.

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

3) $T(n) = 4T(n/2) + O(n)$
   ** Hint: $1 + 2 + 4 + 8 + \ldots + n = 2^{(n+1)} - 1$

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

Question 3 (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) Write the detailed recurrence equation for this algorithm (do not omit any terms).

2) Simplify the recurrence equation by throwing away terms that are dominated by others. And then use **Both** the Master Theorem and the Tree-Based method to compute the complexity of this algorithm.