CS 4310 - Algorithms - Fall 2017

HomeWork2

Given: September 28, 2017

Due: October 12, 2017

General Comments: *Read questions carefully and answer ALL parts of the question. Show all your work, otherwise no partial credit. No credit without proper justifications. State all your assumptions. No Algorithm is complete without its time and space complexity. When presenting an algorithm, first indicate if it is similar to a well-known algorithm, second describe intuitively how the algorithm works (which may be supported by examples), third give its pseudo code and finally analyze its time and space complexity.* Always describe general idea of the algorithm before giving its pseudo-code. Do not reinvent the wheel, i.e., if a well-known algorithm can be modified to solve a problem efficiently, use that solution and clearly indicate the changes required. Do not unnecessarily complicate a solution, i.e., if a simple but efficient solution exists then we should use it. Finally, if you just write pseudo-code of a well-known algorithm without indicating how it applies or modified to the problem at hand, no credit will be given.

1. (10pts; Basic D&C) Problem R-5.5 on page 282 of Goodrich book: use divide and conquer algorithm from section 5.2.2 to compute 10110011.10111010 in binary. Show your work.

2. (10pts; Basic D&C) Assuming that the pivot is the 1st element of a list, simulate the execution of QuickSort on the following input:

   23, 18, 27, 6, 30, 35, 14, 1, 0, 100, 47, 43, 122, 5, 8, 250, 19, 34

   Will the runtimes differ based on the data-structure used to store the data? Consider the cases of an array or a single linked list as a data structure to store the data. Note that the middle element of a list {a1, a2, ..., am} of length m would be the element that is at position [m/2] of the list (and not necessarily the mean, median or average) where [m/2] represents the integer division

3. (10pts; Basic D&C) Problem R-5.6 on page 282 of Goodrich book: Use Strassen's matrix multiplication algorithm to multiply the matrices:

   $X = \begin{bmatrix} 3 & 2 \\ 4 & 8 \end{bmatrix}$ $Y = \begin{bmatrix} 1 & 5 \\ 9 & 6 \end{bmatrix}$

4. (20pts; hw/sw co-design)
   Recall the Karatsuba's multiplication algorithm to multiply two n-digit numbers. Design an 8-bit multiplier using Karatsuba's algorithm with basic building blocks of 1-bit multipler and m-bit adders and shifters, m>0. Note that in class we did not consider the cases when n is not a power of two, you may have to modify the algorithm to take care of these cases. Also, we dicussed algorithm using decimal digits, obviously it can be easily extended to any radix, so for this problem radix is 2.

5. (20pts; Recurrence relations and basic D&C) Consider the following extension to BinarySearch a sorted list A for an item x: *Let ap, aq and ar be the partitioning elements of A so that A is divided into four roughly equal-sized lists (i.e., a difference of 1, 2 or 3 is allowed between the sizes of these sublists but no more). Based on ap, aq and ar, decide the sublist where x may exist. Repeat this process until the sublist is small enough that the answer can be directly obtained.*

   - Write pseudo-code for the above QuadSearch method, specifying all the parameters correctly.
   - Derive the recurrence relation for the running time of the QuadSearch algorithm.
   - Solve the recurrence relation, find a close-bound for the running time of QuadSearch and then express this close-bound using asymptotic notation. Justify your answer.
   - Derive an expression and its asymptotic bound for the space complexity of QuadSearch.

6. (30pts; Basic D&C and empirical observations) In this problem, you will implement a modified version of merge sort. In this version array is divided into three almost equal size arrays (differences of 1 or 2 and not more). The merge part is very similar to merging two sorted arrays, but this time you have to compare three elements each time instead of two.

   - Write pseudo-code for the above 3-way merge sort method
   - Implement it
   - Derive the recurrence relation for the running time of the algorithm, Solve the recurrence relation, find a close-bound for the running time and compare it with the time complexity of original merge sort. Justify your answer.

   Testing: Run several tests (at least 20 runs) with arrays of different lengths. Measure the running times of your implementation, and compare that with the theoretically derived complexity.

General Instructions on submitting your homeworks.

- For programming assignments, submit a SINGLE zipped file of your source codes, scripts (to run your program if any) and a brief report along with a copy of a couple of sample executions of your solution to the class's Elearning. No need to say, but you should be using good conventions and programming practices in developing your programs [just in case you forgot, refresh them from some of the coding conventions etc links provided on the TopicsCovered page.]

- Use <hw#cs4310_yourlastname_mmddyy.{zip,ppt,doc,tex}> as the naming convention for your zipped, ppt, MS-Word, or LaTex files when submitting on Elearning. Replace '#' with the appropriate homework number.
- There will be significant point penalties for not following the naming convention above, good coding practices, submitting a different format of archive file or if your program does not run. Make sure it is a .zip and NOT another format (no .rar, .tar, .tar.gz, etc)

Any student may be asked to show and discuss his or her solution in class, so be ready with your presentation.