CS 330 – Spring 2016, Assignment 4

Problems due by 7PM on Thursday, March 17
Late assignments will only be accepted up until 5PM, Friday, March 18

Question 1. *Suppose you are choosing between the following three algorithms, all of which have O(1) base cases for size 1:*

1. *Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.*
2. *Algorithm B solves problems of size n by recursively solving one subproblem of size $n/2$, one subproblem of size $2n/3$, and one subproblem of size $3n/4$ and then combining the solutions in linear time.*
3. *Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.*

What are the running times of each of these algorithms (in asymptotic notation) and which would you choose? You may use the Master Method. Hint: Approach Algorithm B via the substitution method (pp. 211-217) to avoid tough summations. Also, solving that one as O(n^d) for the smallest valid integer d is all I'm looking for.

Question 2. *Recall the problem of* finding a majority element *of n bank cards we solved in Lab 5, using O($n \log n$) yes-no equivalence tests. Now we'll try to solve it faster. Consider the following procedure when n is even: Arbitrarily pair up the cards. Test each pair and if the two elements are different, discard both of them; if they are the same, keep just one of them. Show that this procedure preserves the majority element: if the original batch of cards had a majority element, then the reduced batch of cards has the same majority element. Use this procedure to build a divide-and-conquer algorithm for all n. Analyze its running time by writing a recurrence relation and evaluating it.*

Question 3. *A* Toeplitz matrix *is an $n \times n$ matrix $A = (a_{ij})$ such that $a_{ij} = a_{i-1,j-1}$ for $i = 2, 3, \ldots n$ and $j = 2, 3, \ldots n$. [Important: the values a_{1j} and a_{i1} can be arbitrary.]*

(a) Is the sum of two Toeplitz matrices necessarily Toeplitz? What about the product of two Toeplitz matrices?

(b) Describe how to represent a Toeplitz matrix so that two $n \times n$ Toeplitz matrices can be added in $O(n)$ time.

(c) Give an $O(n \log n)$ time algorithm for multiplying an $n \times n$ Toeplitz matrix by a vector of length n, using your representation from part (b).

(d) Give an efficient algorithm for multiplying two $n \times n$ Toeplitz matrices. Analyze its running time.

Question 4. *Let's take a last look at the maximum sum subinterval problem.*

(a) Define an appropriate recurrence OPT(j) that correctly describes the maximum sum subinterval between 1 and j as a function of OPT(i) for some $i < j$, and concisely describe why it is correct. (A deficiency of our previous algorithms for this problem were that they needed special handling when the input was all negative. Your recurrence should not have this flaw, i.e., it should still work when the input is all negative).

(b) Write pseudocode for a memoized recursive algorithm that leverages the recurrence in (a) to implement a solution to the maximum sum subinterval problem. Briefly argue that it runs in linear time.

(c) Now provide pseudocode for an iterative algorithm that leverages the recurrence in (a) to implement a solution to the maximum sum subinterval problem. Briefly argue that it runs in linear time.

Question 5. *Chapter 6, Problem 2 on p. 312.*