Computer Science 130B
Spring 2017
Written Assignment #1

Due:**4pm** Friday, April 14th

Problem 1 In divide-and-conquer, the way a problem is divided into pieces often affects the effectiveness and applicability of the proposed solution. Consider the Towers of Hanoi problem again. Define the original problem as

$$Hanoi(n, A, B, C),$$

where the first parameter denotes the number of disks, the second parameter the starting peg, the third parameter the ending peg, and the last parameter the temporary storage peg.

a. Consider the following divide-and-conquer strategy:

$$Hanoi(n, A, B, C) \Rightarrow Hanoi(1, A, C, B) + Hanoi(n-1, A, B, C) + Hanoi(1, C, B, A),$$

which divides the original problem into *two* instances of size 1 and *one* instance of size $n-1$. If you think the strategy will work, give the detailed motion sequence for Hanoi(4,A,B,C) and estimate the complexity of the algorithm. If you think the strategy will not work, provide a counterexample.

b. If $n = 2^k$, consider the following divide-and-conquer strategy:

$$Hanoi(n, A, B, C) \Rightarrow Hanoi(n/2, A, C, B) + Hanoi(n/2, A, B, C) + Hanoi(n/2, C, B, A).$$

Answer **1.a** for this particular strategy.

Problem 2 Prove by induction that the devide-and-conquer strategy

$$Hanoi(n, A, B, C) \Rightarrow Hanoi(n-1, A, C, B) + Hanoi(1, A, B, C) + Hanoi(n-1, C, B, A),$$

solves the Hanoi Tower problem without violating the constraints that no bigger disks are placed on top of smaller disks on the same peg and only one disk is moved at a time in the solution sequence.

Problem 3 You are given an array A containing n integers with no duplication. The integers are sorted, but not necessarily sequential (or consecutive). You want to find out if there exists an index i for which $A[i] = i$. For example, if A = [-17; -3; 1; 4; 6; 20], then A[4] = 4 (assuming 1-based array indexing instead of 0-based like in C). Give a divide-and-conquer algorithm that runs in time $O(logn)$.

Problem 4 Consider a set of numbers. The median is the middle number that is larger than or equal to half of the numbers and smaller or equal to the other half in the set.

a. Design a brute-force algorithm to find the median of an *unsorted* array of an odd length. Analyze the complexity of your algorithm.

b. Design a divide-and-conquer algorithm to find the median of an *unsorted* array of an odd length. Analyze the complexity of your algorithm. Your divide-and-conquer algorithm should be more efficient than your brute-force algorithm in order notation.

Problem 5 You are given n arrays of numbers. All the arrays are sorted and are roughly of the same length l. You are to merge these n arrays into one single sorted array. This is an essential operation in combining many local databases into one single, global database.

a. Obviously, you can merge the first two input arrays into one combined array, then merge this combined array with the third input array into an even bigger combined array, then merge this bigger combined array with the fourth array, and so on so forth. What is the complexity (number of record moves) of such a strategy?

b. Now design a divide-and-conquer strategy to merge these n arrays. What is the complexity of your divide-and-conquer algorithm? You can assume for simplicity of analysis that n is a power of 2.