EECS660 **HW2** **Due: 9-18-14**

Instruction: Read and follow the Homework Submission Policy carefully. You must show all your work clearly for credit. Partial credit will only be given to meaningful answers. If algorithm is required in your solution, you must first explain your algorithm in plain English and then present your algorithm in pseudocode. You must always justify the correctness of your algorithm. You will be graded according to your approach to the problems, mathematical rigor, and quality of your solutions. A correct but inefficient algorithm will receive very little credit!

1. (10) Assuming that a divide-and-conquer algorithm A is used in solving a given problem Π and let T(n) be the complexity in solving Π with any input of size n. If A divides Π into 64 subproblems each of which has an input of size $\frac{n}{6}$, and if the dividing cost and the combining cost are given by the functions $660n^2 + 128n - 433$ and $n^3 - 18n\lg n + 210$, respectively, set up the recurrence equation for T(n). Compute T(n) using the Master Theorem.

2. (10) Using the Method of Repeated Substitutions, solve the following recurrence for T(n) in closed-form:

$$T(1) = 0,$$
$$(n-1)T(n) = nT(n-1) + n(n-1)(n+1), n > 1.$$

3. (20) Using the Method of Repeated Substitutions, compute $T(n)$ in closed-form and then in big-O. Verify your solution in big-O using the Master Theorem.

$$T(8) = 2,$$
$$T(n) = 4T(\frac{n}{2}) + 18n^2 - 5, n = 2^k > 8.$$

4. (20) Given an array A[1..n] and a key x, $n = 3^k \geq 1$. If $\Pr(x \in A) = \frac{3}{8}$ and, if x is in A, the probability that x is in the first third of the array is three times more likely for x to be in the second third of the array, and the probability that x is in the second third of the array is twice more likely for x to be in the last third of the array. Furthermore, if x is in any third part of the array, x is equally likely to be found in any one of the positions in that part of the array. Based on the number of comparisons, compute $T_a(n)$ if sequential search is performed on A for x. You must set up the equation for $T_a(n)$ and then evaluate the sums. Do not simplify your expression.

5. (20) Given an array A[1..n] with n distinct integers, $n = 4^m \geq 4$. By dividing A into four equal parts, design an efficient DAC merge sort algorithm by extending the standard merge sort algorithm for sorting A. Assuming that it will take 2n-3 comparisons to merge the four sub arrays together, based on the number of comparisons between elements in A, set up the recurrence equation for $T_w(n)$ and then compute it in closed-form.

6. (20) Given a positive integer $k \geq 2$, an array A[1..n] with n distinct integers, $n = k^m \geq 1$, and an integer key x. Design a k-ary search algorithm by extending the standard binary search algorithm for searching the array A for x . Based on the number of comparisons between x and the elements in A, set up and then solve the corresponding recurrence equation for the k-ary search algorithm for $T_w(n)$. Can you determine the value of k for the best performance k-ary search algorithm.

Remark: Turn in Problems 2, 3, and 4 for grading.