CS 430 – FALL 2017
INTRODUCTION TO ALGORITHMS
HOMEWORK #2
DUE Fri Sept 15, 11:25am

1. (3 points) What if you are sorting a collection of data that can have multiple entries of the some of the values. When calling Quicksort's Partition(A, p, r), where do elements equal to the pivot end up and why?
How could we modify Quicksort and Partition (write pseudocode) so that if we happen to partition on a pivot that has many duplicate values, we can improve the runtime of Quicksort?

2. (3 points) Exercise 7.2-5

3. (3 points) Given a maxHeap storing n keys give an efficient, O(k), algorithm for outputting all keys (in any order) in the maxHeap that are greater than or equal to give query value x (x is not necessarily in the maxHeap). "k" is the number of maxHeap keys output.

4. (3 points) Professor Fermat has the policy of giving A's to the top n^(1/2) students of his class, where n is the number of students. The algorithm that he uses to determine the top n^(1/2) students first sorts the list of students by their numerical, real-valued grade, and then picks the top n^(1/2) students from the sorted list. This algorithm has time-complexity O(n logn) because of the sort. Can you suggest a more efficient algorithm that has time-complexity O(n)? Describe your algorithm informally in English and justify its time-complexity.

5. (6 points) Professors Howard, Fine, and Howard have proposed the following "elegant" sorting algorithm:

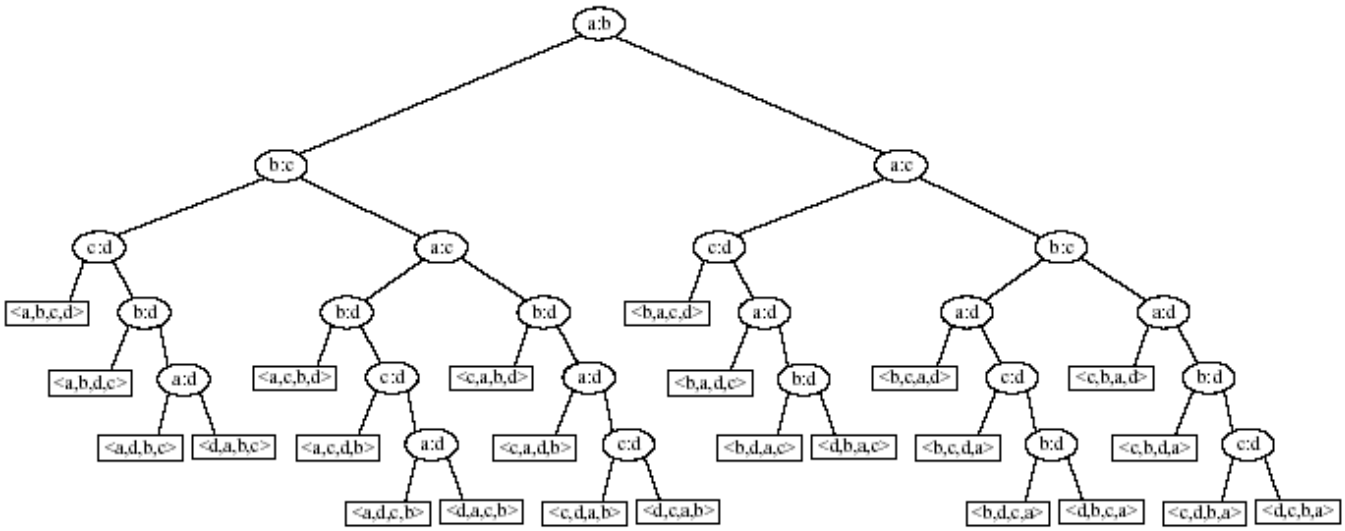
```
STOOGE-SORT(A, i, j)
  if A[i] > A[j]
     then exchange A[i] with A[j]
  if i + 1 ≥ j
     then return
  k = floor((j - i + 1)/3)          // Round down.
  STOOGE-SORT(A, i, j - k)          // First two-thirds.
  STOOGE-SORT(A, i + k, j)          // Last two-thirds.
  STOOGE-SORT(A, i, j - k)          // First two-thirds again.
```

5a) Argue that then STOOGE-SORT(*A*, 1, *n*) correctly sorts the input array *A*[1 .. *n*].

5b) Give the recurrence for the worst-case running time of STOOGE-SORT and solve the recurrence for a tight asymptotic (Θ-notation) bound on the worst-case running time.

5c) Compare the worst-case running time of STOOGE-SORT with that of insertion sort, merge sort, heapsort, and quicksort. Do the professors deserve tenure?

6. (1 point) The following comparison tree sorts the four values *a*, *b*, *c*, and *d*.



(a) What is the worst-case number of comparisons performed by the comparison tree?
(b) What is the best-case number of comparisons performed by the comparison tree?
(c) What is the average-case number of comparisons performed by the comparison tree, assuming that any permutation of the five inputs is equally likely?

7. (4 points) The Hewlett-Packard HP9810A programmable desk calculator cost $2960 in 1971 and could perform about 100 instructions per second. Below is a table showing the sizes of the largest problems that it could solve in **one minute**. For less than this price, you can now buy a computer that is 10^6 times faster (10^6 is about 2^{20} times faster). State the worst-case **time complexity** of each algorithm in the form "O( )" and then estimate the **size of the largest problem** that the **modern** computer could solve in the same amount of time (one minute).

Algorithm	largest problem size in 1971
(a) Multiplying two n×n matrices by the naïve algorithm.	n = 6
(b) Heap sort on an array of n integers.	n = 32
(c) Selection sort on an array of n integers.	n = 32
(d) Linear search on an array of n integers.	n = 200

You are advised to write down the mathematical equation that needs to be solved before you do any arithmetic, and set out your rough calculations as clearly as possible. It is enough to give **order-of-magnitude** answers, so long as they correctly indicate the **relative** values.

8. (2 points) We have a problem that can be solved by a iterative (nonrecursive) algorithm that operates in N^2 time. We also have a recursive algorithm for this problem that takes $N \lg N$ operations to divide the input into two equal pieces, two recursive calls to conquer, and $\lg N$ operations to combine the two solutions together. Show whether the iterative or recursive version is more efficient.