[**Closed book, calculator, and notes**] Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. Note that the exam is double sided.

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \quad \Rightarrow \quad f(n) = \Theta(h(n)) \tag{1}$$
$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \quad \Rightarrow \quad f(n) = O(h(n)) \tag{2}$$
$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \quad \Rightarrow \quad f(n) = \Omega(h(n)) \tag{3}$$

$$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n)) \tag{4}$$

$$\lg n = \log_2 n \tag{5}$$
$$\ln n = \log_e n \tag{6}$$
$$a = b^{\log_b a} \tag{7}$$
$$\log_c(ab) = \log_c a + \log_c b \tag{8}$$
$$\log_b a^n = n \log_b a \tag{9}$$
$$\log_b a = \frac{\log_c a}{\log_c b} \tag{10}$$

$$\sum_{k=1}^{n} k = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} = \Theta(n^2) \tag{11}$$

$$\sum_{k=0}^{n} x^k = 1 + x + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1} = \Theta(x^n), \qquad x \neq 1 \tag{12}$$

$$\sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln n + .577 = \Theta(\log n) \tag{13}$$

Given positive functions $f(n)$ and $g(n)$ such that

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$$

for some constant c.

1. If $0 < c < \infty$, then $f(n) = \Theta(g(n))$
2. If $0 \leq c < \infty$, then $f(n) = O(g(n))$
3. If $0 < c \leq \infty$, then $f(n) = \Omega(g(n))$

If $f(n)$ and $g(n)$ both approach zero or both approach ∞ in the limit, then

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{f'(n)}{g'(n)}$$

where $f'(n)$ and $g'(n)$ denote derivatives of f and g with respect to n.

1. Consider a modification to merge sort in which n/k sublists of length k are sorted using quicksort and then merged using the standard merging mechanism.

(a) (10 points) Show that the n/k sublists, each of length k, can be sorted by quicksort in $\Theta(nk)$ worst-case time.

(b) (10 points) Show that the sublists can be merged in $\Theta(n \log(n/k))$ worst-case time.

2. Consider an array, A, of n elements where all elements in A are equal.

(a) (5 points) What is tight asymptotic time complexity for running insertion sort on A? Justify your answer.

(b) (10 points) What is tight asymptotic time complexity for running heapsort on A? Justify your answer.

(c) (10 points) Write the recurrence equation for running quicksort on A and give the $\Theta(\cdot)$ notation for the running time. Justify your answer.

3. Find the asymptotic growth rate of the following recurrence:

$$T(n) = \begin{cases} 1 & n = 1, \\ T(\lceil \frac{n}{3} \rceil) + T(\lfloor \frac{n}{3} \rfloor) + \frac{n}{3} & \text{otherwise.} \end{cases} \tag{14}$$

Show all of your work and state any assumptions you make.

(a) (5 points) Using the Master method

(b) (15 points) Using iteration

4. (15 points) About a year ago, Google, Inc. announced that they had sorted 10,000,000,000,000 100-byte records in just over six hours. They used 4,000 computers and 48,000 hard drives to store and sort the data.[1] Given the constraints listed above, describe how you would make use of known sorting algorithms to sort the 10 trillion records? Answer in English as if this were a job interview question.

Grading — Definite hire: 15, Second interview: 12, Consider next year: 10, Maybe: 7, Never: 3

[1] http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html

5. (20 points) An element x of an array A is the *majority* element of A if at least 50% of the array elements have the value x (Note: it is not neccessary for an array to have a majority element.)
Suppose you can perform comparisons on the elements of the array. The comparison test can give three possible outcomes: "=", "$<$", and "$>$". Describe in Java, pseudocode, or unambiguous English an algorithm to determine whether or not a majority element exists. Your algorithm should require $O(n)$ comparisons. Hint: recall that the Select algorithm (terminate-early-quicksort) for finding the i^{th} order statistic is $\Theta(n)$ for the average case.

Receive up to 20 points for a worst-case $O(n)$ solution, up to 18 points for an average-case $O(n)$ solution, and up to 10 points for a best case $O(n)$ solution.

bonus (10 points… no partial credit) Suppose that you have a basket of fruit containing apples, oranges, bannanas, etc.… Comparing apples to oranges is a no-no, so we are left with a comparison that only has two outcomes: "=" and "≠".

First show that if $A[i] \neq A[j]$ are both removed from the array, x is the majority element of the resulting array if and only if x was the majority element of the original array.

Then implement a Java method that accepts a basket of fruit (an array of Fruit references) and returns a reference to an object from the majority fruit type (the majority element, if it exists) or `null`. (Assume that the Fruit.equals() method returns `true` if and only if the fruit being compared are the same type (e.g., both apples)). Your algorithm must require $O(\text{A.length})$ comparisons and should not assume a fixed number of varieties of fruit.

Additional work area for any problem. Clearly identify to which problem the work on this page is related.