CS161 - Homework 2
Due: Thursday July 9, 5pm

1. (24 points, 3 each) (CLRS 4-4, subproblems a-g,j (pg. 86). You only need to provide upper bounds (i.e. O) for recurrences you don't solve using the master method.

2. (5 points) CLRS 4.3-2 (pg. 75). Explain your answer.

3. (25 points) There are some situations where we are asked to sort data that is *almost* sorted. A k-sorted array contains no element that is more than k positions from it's position in the properly sorted array. For the questions below, A is a k-sorted array with $k << n$:

   (a) (10 points) What are the runtimes of Insertion-Sort, Merge-Sort and Quicksort on A? Explain your answers.

   (b) (5 points) What is the runtime of Bubble-Sort on A (see pseudocode below)? Explain your answer.

```
BUBBLE-SORT(A)
1  sorted ← false
2  while sorted = false
3          sorted ← true
4          for i ← 1 to length[A] −1
5                  if A[i] > A[i + 1]
6                          swap A[i] and A[i + 1]
7                          sorted ← false
```

   (c) (10 points) Write pseudocode and provide the runtime for an algorithm that performs better than the above algorithms for sorting a k-sorted array.

   (d) (5 points) Prove the correctness of your algorithm.

4. (15 points) Even $O(n \log n)$ may not be fast enough for certain cases where the data set is very large and/or when response time is very important. One way to solve this problem is to use multiple machines to sort the data.

   (a) (5 points) Of the three sorting methods we've examined so far (Insertion-Sort, Merge-Sort and Quicksort), which algorithm is the most amenable to this situation? What are the challenges for the other two?

   (b) (5 points) Describe how you would implement your choice on m computers. You can assume $m = 2^i$ for integer i.

   (c) (5 points) What is the runtime of this method?

5. (8 points) A stable sorting algorithm is an algorithm where elements with equal value appear in the output (sorted) array in the same order as they do in the input array. Which of Insertion-Sort, Merge-Sort, Bubble-Sort and Quicksort are stable? Briefly explain your answers.

Extra Credit

6. (5 points) Prove that you cannot do any better than your answer provided for a k-almost sorted array. (Hint: see pgs. 165-168)

Just for fun

7. Given n points in a 2-dimensional plane, find the pair that is closest together. You can do better than $O(n^2)$ time.

8. Although Quicksort is one of the most commonly used sorting algorithms, it is not the fastest in all situations. Program (or find versions of) a few sorting algorithms (e.g. Insertion-Sort, Merge-Sort, Bubble-Sort and Quicksort). Compare the run-times of the algorithms as you vary:

   - The size of the array
   - The "sortedness" of the array (One easy way to vary this is to take a sorted array and randomly swap some number of elements. You can also fix the distance of the swap to test for k-sorted data.)
   - The size of the elements (i.e. chars vs. integers vs. longs)

Make sure to run multiple experiments for a given setting to get a feeling for the variance in the data. Also, be careful about things like garbage collection if you're having trouble getting consistent results. If you do this, please do let me know and we can share your results with the class.