FALL 2005: **COT 5407** Intro. to Algorithms

[Homework 2; Due Sep 29 at start of class]

Reminder: As in the previous homework add a signed statement: I have adhered to the collaboration policy for this class and that whenever no explicit citations or sources of help are indicated, what I have presented is my own work.

Problems

9. (**Exercise**) Solve these exercises (These will not be graded; it is enough to provide just final answers to show you have worked it out): Exercise 7.1-1, p148; Exercise 7.2-2, p153; ercise 7.2-4, p153; Exercise 6.2-1, p132; Exercise 6.3-1, p135; Exercise 6.4-1, p136; Exercise 6.5-1, p140;

10. (**Regular**) List out the sorting algorithms that have a **worst-case** time complexity of $O(n \log n)$. No explanation is required.

11. (**Regular**) Write down the **best-case** time complexities of all the sorting algorithms discussed in class and list them in increasing order.

12. (**Regular**) Study Randomized-Partition and Randomized-Quicksort from page 154 of your text. Now modify it to implement two new features:

    (a) Implement the **median-of-3** method for choosing the pivot (described on page 162);

    (b) Implement R. Sedgewick's idea (1978) to avoid recursive calls when the size of the array is at most k. Set the value of k to be 8, which is the cutoff value used in the 1997 Microsoft C library implementation of Quicksort.

13 (**Exercise**) Read Section 6.5 and algorithm Max-Heap-Insert on page 140 and then solve Exercise 6.5-2, p140.

14. (**Regular**) A sorting algorithm is said to be *stable* if the relative order of any equal items in the input list is not changed in the output list. In order to show that a sorting algorithm is stable, one would need a mathematical proof. However, to prove that it is not stable, all we need is a simple "counterexample". It should be obvious to you that Insertion Sort, Bubble Sort, and Merge Sort are stable sorting algorithms (first convince yourself of this). For each of the following sorting algorithms, state which ones are **not** stable by using simple examples: Selection Sort, Quick Sort, and Heap Sort. Devise the smallest example you can build, if you claim the algorithm is not stable. Write down a brief argument if you think it is stable. You should consider a sorting algorithm to be stable if the algorithm given to you in the book or the one given to you in class (or a minor variant thereof) is stable.

15. (**Extra Credit**) (Exercise 8-5, p180)