COT 5405 Analysis of Algorithms, Fall 2008.
Homework 1

Due Thursday, September 25 2008, 2:30pm.

Notes
• The TA will be available in CSE E309, at the time the homework is due. You may also hand the homework to Prof. Ranka in class. No late submissions.
• To force you to write succinctly, we have enforced page limits. These are noted in front of each question.
• Answer each question on a fresh page.
• If the problem necessitates writing an algorithm, you must first informally describe the algorithm, in brief, in a paragraph. You can choose to follow this up with pseudocode that formally describes the algorithm. We will peruse your pseudocode only if your English description is not clear.
• Write your name on the top right hand corner of your homework. Be sure to write your last name as the last word in your name.
• If you are designing an algorithm, you must write a formal proof of correctness.
• Please write legibly.

1. [1 page] Suppose you were given a blackbox, which when given a list as input, returns a pivot in linear time in the following way: The list is partitioned into sublists of length 5. For each of these sublists, the median is found. Further, the median of these medians is found and returned as the pivot. Now, using this (the output of the blackbox) as the pivot, quicksort is done. Is it now possible to obtain a better worst-case performance guarantee for quicksort than the classical quadratic one? If so, perform the analysis and prove the new bounds. What would the average-case bounds be?

2. [1 2 page] You are interested in analyzing hard-to-obtain data from two separate databases or sets. Each set contains \( n \) numbers - so there are total of \( 2n \) numbers. You can assume that all the numbers are distinct. You are required to find the median of these \( 2n \) numbers. However, the only way to access these numbers is through queries to the two databases or sets. Each query has to be a selection query - i.e. for a given value of \( k \), it returns the \( k^{th} \) smallest number in the set. Develop an algorithm to find the median that requires at most \( O(\log n) \) queries.

3. [1 2 page] In this problem we are interested in solving a modified “closest pair” problem. This time rather than having \( n \) indistinguishable points, there will be \( n \) red and \( n \) blue points. The task is to find the closest red-blue pair (i.e. the minimum distance between a red point and a blue point). Can this problem be solved within the same \( O(n \log n) \) bound by modifying the divide-and-conquer algorithm given in class? If ‘yes’, give the modified algorithm with its running time, if ‘no’ explain why.
4. [\frac{1}{2} \text{ page}] The merge sort algorithm divides the array into two parts and after recursively solving the problem for the parts merges them to get the sorted array. But we could also divide the array into \( k \) parts and merge them. Show if and how the asymptotic running time would change. What results would you expect in practice, and would it be a good idea to implement the algorithm that way?

5. [\frac{1}{2} \text{ page}] Solve the following recurrence relations.

(a) \( T(n) = 3T(n/2) + n \)

(b) \( T(n) = 8T(n/2) + n^3 \)

(c) \( T(n) = 2T(n/4) + \sqrt{n} \)

(d) \( T(n) = T(\sqrt{n}) + 1 \)

6. [1 \text{ page}] You are the TA for a class with an enrollment of \( n \) students. You have their final scores (unsorted), and you must assign them one of the \( k \) available grades (A, B, C etc.). The constraints are (assuming \( n \) is a multiple of \( k \)):

- Exactly \( \frac{n}{k} \) students get each grade (for example, if \( n = 30 \), and \( k = 3 \), i.e. the available grades are \{A, B, C\}, then exactly 10 students get A, 10 get B, and 10 get C)
- A student with a lower score doesn’t get a higher grade than a student with a higher score (however, they may get the same grade)

Assuming that each student received a different score, derive an efficient algorithm and give its complexity in terms of \( n \) and \( k \). Any algorithm that first sorts the scores will receive zero credit.

7. [\frac{1}{2} \text{ page}] Suppose an oracle knows a natural number \( n \) that you wish to know. The oracle answers only Yes/No, to the following three types of queries:

(a) Is the number greater than \( x \)?

(b) Is the number lesser than \( x \)?

(c) Is the number equal to \( x \)?

(\text{where } x \text{ can be an arbitrary natural number, that can changed across queries}).

Describe a method for posing queries to the oracle, which is asymptotically efficient in the number of queries posed. Perform the analysis and write a proof of correctness. Note that the number of queries posed will be a function of \( n \).

8. [\frac{1}{2} \text{ page}] Consider a list of size \( n \), which has at most \( \log n \) distinct numbers, the rest being repetitions of those numbers. Design an algorithm to sort this list in \( \Theta(n \log \log n) \) time. (Hint: First partition the list into \( \frac{n}{\log n} \) lists of size \( \log n \) and use a mergesort-like strategy).