Homework 2

Divide-and-Conquer Algorithms, Sorting Algorithms, Greedy Algorithms

Deadline: start of 13th lecture (i.e., 3:20pm, March 8, 2013).
Available points: 110. Perfect score: 100.

You will receive 10% extra credit points if you submit your answers as a typeset PDF (preferably using LaTeX, in which case you can also submit electronically your source code). There will be a 5% bonus for typewritten but not typesetted answers. Resources on how to use LaTeX are available on the course's website. Do not submit Word documents, raw text, etc. Make sure to generate and submit a PDF if you want to get the extra credit points. In this case you can submit your solutions electronically through `sakai.rutgers.edu`.

If you choose to submit handwritten answers and we are not able to read them, you will not be awarded any points for the part of the solution that is unreadable. Handwritten answer-sheets can be submitted to the instructor in class or to one of the TAs during office hours.

Try to be precise. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Each pair of students must write its solutions independently from other teams, i.e., without using common notes or worksheets with other students. Each pair of students need to submit only one copy of their solutions. You must indicate at the top of your homework who you worked with. You must also indicate any external sources you have used in the preparation of your solution. Make sure you do not violate any of the academic standards of the course, the department or of the university (available through the course's website)

Problem 1: Give asymptotic bounds for the following recurrences. Assume $T(n)$ is constant for $n=1$. Make your bounds as tight as possible, and justify your answers. (20 points)

A. $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$
B. $T(n) = 7T(\frac{n}{2}) + n^2$
C. $T(n) = T(n-1) + n$
D. $T(n) = T(\sqrt{n}) + 1$

Problem 2: Assume n screws of different sizes, and n nuts. You can test a nut and a screw, from which you can decide whether the screw is too large, too small, or a match. Nevertheless, you cannot compare two screws together or two nuts. Your goal is to match each screw to its nut (assume that screws and nuts match one to one).

A. Show that any algorithm for this problem must take $\Omega(nlogn)$ comparisons in the worst case. (10 points)

B. Design a randomized algorithm for this problem that runs in expected time $O(nlogn)$. (10 points)

Problem 3: Assume that you have n sets of k sorted numbers. Your objective is to merge them into a single sorted set of nk numbers.

A. One possible way is to use the `merge` procedure of `mergesort` to combine the first two sets, then combine in the third, the fourth, and so on. What is the running time of this solution, in terms of n and k? (10 points)

B. Give a more efficient solution to this problem, using divide-and-conquer. Prove its correctness and running time. (15 points)

Problem 4: Assume that there are p points in a three-dimensional sphere $\mathcal{B}(R)$ centered at the origin with radius R. The points are uniformly distributed. This means the probability of finding a point in any subset of the sphere is proportional to the volume of that subset. Design an expected linear-time algorithm to sort the p points in $\mathcal{B}(R)$ by their distances $d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$ from the origin. (20 points).

Problem 5: You have to solve n homework problems for a class. You have H time slots available and you cannot divide an hour between problems. Instead you must spend each hour entirely on a single problem. The i^{th} hour you spend on problem p will improve your grade by $G_p(i)$, where $G_p(1) \geq G_p(2) \geq \ldots \geq G_p(H) \geq 0$ for each p. In other words, if you spend h hours on problem p, your grade will be

$$\sum_{i=1}^{i=h} G_p(i)$$

and time spent on each problem has diminishing returns, the next hour being worth less than the previous one. Note that for $p \neq p'$: $G_P(i) \neq G_{p'}(i)$. You want to divide your H hours between the problems to maximize your total grade on all the assignments.

Give an efficient algorithm for this problem, show that it is correct and argue about its running time. Consider what data structures may help you in your implementation. (25 points)