
Homework 5 – Due Wednesday, October 10, 2007

Please refer to the general information handout for the full homework policy and options.

Reminders

- Your solutions are due *before* the lecture. Late homework will not be accepted.
- Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.
- *For all problems where you are asked to design an algorithm, do not forget to prove correctness and analyze your algorithms time and space complexity.*

Exercises These should not be handed in, but the material they cover may appear on exams:

1. Using the PARTITION subroutine and the linear time algorithm for finding the median of list of numbers, design a divide-and-conquer sorting algorithm that runs in $O(n \log n)$ time.
2. Show how to multiply the complex numbers $a + bi$ and $c + di$ using only three real multiplications. The algorithm should take a, b, c , and d as input and produce the real component $ac - bd$ and the imaginary component $ad + bc$ separately.
3. What is the largest k such that if you can multiply 3×3 matrices using k multiplications, then you can multiply $n \times n$ matrices in time $o(n^{\log 7})$, i.e., asymptotically faster than in time $\Theta(n^{\log 7})$? What would the running time of this algorithm be?
4. There is a way of multiplying 68×68 matrices using 132,464 multiplications, a way of multiplying 70×70 matrices using 143,640 multiplications, and a way of multiplying 72×72 matrices using 155,424 multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?

General hint for designing divide-and-conquer algorithms: When the target running time is given, think which recurrence would yield this running time. Try to divide your input accordingly and see if you can combine the results of the recursive calls in the required time.

Problems to be handed in

1. (**Recurrences**) Solve the following recurrences and express your answer using Θ -notation. Use Master Theorem when it applies, and the recursion-tree method otherwise. (You may use the fact that the n th **harmonic number**, defined as $H_n = \sum_{k=1}^n \frac{1}{k}$, is equal to $\ln n + O(1)$.)
 - (a) $T(n) = 27T(n/3) + 2n^3 \log n$.
 - (b) $T(n) = T(49n/50) + \Theta(n)$.
 - (c) $T(n) = 2T(n) + \frac{n}{\log n}$.
 - (d) $T(n) = 4T(n/2) + n^{1/3} \log^2 n$.
2. (**Inversions**) Chapter 5, problem 2. Briefly argue correctness: assuming that the results of recursive calls are correct, your algorithm should perform the “combine” step correctly. (Do not forget to take care of the base case.) Analyze the running time and the space complexity of *your* algorithm.
3. (**Median**)
 - (a) The *majority element* of an array $A[1..n]$, containing n integers, is the element that appears more than $n/2$ times in the array. Design an efficient algorithm that, given an integer array, outputs its majority element if it exists. Your algorithm should use the Order Statistics algorithm we covered in class as a subroutine. Briefly describe your algorithm (at most 3 sentences), and state its running time. You do not have to write down the proof of correctness.
 - (b) Chapter 5, problem 1. The same instructions as for the **Inversions** problem.
4. (**Bank cards**) Chapter 5, problem 3. The same instructions as for the **Inversions** problem.
- 5* (**Optional; no collaboration is allowed on this problem**) Chapter 5, problem 7.