
Homework 1 – Due Friday, January 26, 2007

Reminders 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.

When you describe an algorithm, remember that clarity is the main goal. Pseudocode is preferred, but an English-language description is also acceptable if it is unambiguous.

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

- Levitin (in-class handout) exercises 1.2.3, 1.2.7, 1.2.9.
- CLRS (textbook) problem 1-1, exercises 2.1-3 and 2.2-2.

Problems to be handed in

1. CLRS problem 2-2 (*Correctness of bubblesort*). Add the following, final part:
 - e. For each statement below, either prove it (by giving an example) or explain why it cannot be true.
 - i. There exists a permutation of integers 1 to n on which bubblesort takes time $\Theta(n^2)$, but insertion sort takes time only $\Theta(n)$.
 - ii. There exists a permutation of integers 1 to n on which insertion sort takes time $\Theta(n^2)$, but bubblesort takes time only $\Theta(n)$.
2. (*Anagrams*). Consider the following problem: Given two words, determine whether they are anagrams, that is if one word can be obtained by permuting the letters of the other. (For example, “stop” and “tops” are anagrams; “mutt” and “tumm” are not.) The input is given as a pair of arrays of English letters, and its length, n , is the sum of the lengths of the words. Use big- Θ notation to express running times in the problems below.
 - (a) Using sorting as a subroutine, give an algorithm for the anagrams problem and analyze its worst-case running time. You may assume that the sorting subroutine takes time $\Theta(n \log n)$ on inputs of length n .
 - (b) Give a faster algorithm for the anagrams problem (hint: think about counting). What is the worst-case running time of your algorithm in terms of n ?
 - (c) Suppose now that the “words” are arrays of integers between 1 and n^2 (that is, the alphabet is now $\{1, \dots, n^2\}$ instead of $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots, \mathbf{z}\}$). As before, two arrays are anagrams if one is a permutation of the other. Modify the algorithms from parts (2a) and (2b) to handle these inputs. Do their worst-case running times change and, if so, how?