CSCI 570 Homework #1 Solution

Prof. Ming-Deh Huang
TAs: Iftikhar Burhanuddin, Ranjit Raveendran
08/22/06

1) 1.2-3

What is the smallest value of n such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?

2) 2.3-7

Describe a $\Theta(n \log n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

3) 2-1

Although merge sort runs in $\Theta(n \log n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small n. Thus, it makes sense to use insertion sort within merger sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

1. Show that the n/k sublists, each of length k, can be sorted by insertion sort in $\Theta(nk)$ worst-case time.

2. Show that the sublists can be merged in $\Theta(n \log(n/k))$ worst-case time.

3. Given that the modified algorithm runs in $\Theta(nk + n \log(n/k))$ worst-case time, what is the largest asymptotic (Θ-notation) value of k as a function of n for which the modified algorithm has the same asymptotic running time as standard merge sort.

4. How should k be chosen in practice?

4) 2-4

Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion* of A.

1. List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.
2. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?
3. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.
4. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \log n)$ worst-case time. (Hint: Modify merge sort.)

5) 3-1

Let

$$p(n) = \sum_{i=0}^{d} a_i n^i,$$

where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notations to prove the following properties.

1. If $k \geq d$, then $p(n) = O(n^k)$.
2. If $k \leq d$, then $p(n) = \Omega(n^k)$.
3. If $k = d$, then $p(n) = \Theta(n^k)$.
4. If $k > d$, then $p(n) = o(n^k)$.
5. If $k < d$, then $p(n) = \omega(n^k)$.

6) 3-3

1. Rank the following function by order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{30}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{29} = \Omega(g_{30})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln\ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg\lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2\lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

2. Give an example of a single nonnegative function $f(n)$ such that for all functions $g_i(n)$ in part (a), $f(n)$ is neither $O(g_i(n))$ nor $\Omega(g_i(n))$.

7) 4-1 c, h; 4-4 c, h

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 16T(n/4) + n^2$.
2. $T(n) = T(\sqrt{n}) + 1$.
3. $T(n) = 4T(n/2) + n^2\sqrt{n}$
4. $T(n) = T(n-1) + \lg n$

8) 4-7

An $m \times n$ array A of real numbers is a *Monge array* if for all i, j, k and l such that $1 \leq i < k \leq m$ and $1 \leq j < l \leq n$, we have

$$A[i,j] + A[k,l] \leq A[i,l] + A[k,j].$$

In other words, whenever we pick two rows and two columns of a Monge array and consider the four elements at the intersections of the rows and the columns, the sum of the upper-left and lower-right elements is less or equal to the sum of lower-left and upper-right elements. For example, the

following array is Monge:

10	17	13	28	23
17	22	16	29	23
24	28	22	34	24
11	13	6	17	7
45	44	32	37	23
36	33	19	21	6
75	66	51	53	34

1. Prove that an array is Monge if and only if for all $i = 1, 2, \ldots, m-1$ and $j = 1, 2, \ldots, n-1$, we have

$$A[i, j] + A[i+1, j+1] \leq A[i, j+1] + A[i+1, j].$$

   (Hint: For the "only if" part, use induction separately on rows and columns.)

2. The following is not Monge. Change one element in order to make it Monge. (Hint: Use part (a).)

37	23	22	32
21	6	7	10
53	34	30	31
32	13	9	6
43	21	15	8

3. Let $f(i)$ be the index of the column containing the leftmost minimum element of row i. Prove that $f(1) \leq f(2) \leq \ldots \leq f(m)$ for any $m \times n$ Monge array.

4. Here is a description of a divide-and-conquer algorithm that computes the leftmost minimum element in each row of an $m \times n$ Monge array A:

   Construct a submatrix A' of A consisting of the even-numbered rows of A. Recursively determine the leftmost minimum for each row of A'. Then compute the leftmost minimum in the odd-numbered rows of A.

   Explain how to compute the leftmost minimum in the odd-numbered rows of A (given that the leftmost minimum of the even-numbered rows is known) in $O(m + n)$ time.

5. Write the recurrence describing the running time of the algorithm described in part (d). Show that its solution is $O(m + n \log m)$.