CSE 101 - ALGORITHMS - SUMMER 2000
Sample Midterm

Problem 1:

Outline an efficient algorithm for solving each of the following problems. Give the order of the worst-case complexity for your algorithms. You can use the well-known algorithms as subroutines.

1. (4 points) Let S be an unsorted array of n distinct integers. Give an algorithm that finds the pair of elements x, y in the array S such that $x \neq y$ and $|x - y|$ is minimized.
2. (6 points) Give an efficient algorithm for sorting a list of n keys that may each be either 0 or 1. What is the order of the worst-case running time of your algorithm? Write the complete algorithm in pseudo-code. Make sure your algorithm has the stable sorting property.

Proof: (Hint)

1. First sort S. Takes $O(n \cdot \log n)$.

   Then scan the sorted array, getting the minimum of $|S[i] - S[i+1]|$ for all i. Let m be that minimum.

   Takes $O(n)$ time.

   Output m.

   Obviously, the total time is $O(n \cdot \log n)$.
2. Use radix sort (with count sort) for one digit numbers (so $k = 1$). Then the running time is $O(n \cdot k)$, that is, $O(n)$. Alternatively, you can create two arrays, one for zeros and another one for ones, scan the input updating the two arrays, and then append them. The pseudo-code is easy.

Problem 2:

(10 points) Given an array of n real numbers, consider the problem of finding the maximum sum in any contiguous subvector of the input. Give a $\Theta(n)$-time dynamic programming algorithm for this problem.

Proof: (Hint)

Let L_i be the maximum sum in any contiguous subvector starting with position i. Notice that $L_i \geq 0$ as we can take the subvector of size 0. Then

$L_{n+1} = 0$, and

$$L_i = \textbf{if } A[i] > 0 \textbf{ then } \max\{A[i], A[i] + L_{i+1}\}$$
$$\textbf{else } \max\{0, A[i] + L_{i+1}\}.$$

Now it is easy to write a linear algorithm to calculate L. Then take the largest L_i also in linear time.