Algorithms Section 2
Spring 2003
Homework 3a
Due Thursday, February 6

1. **[15 Points] Professor Lai's Dot Product Algorithm.** Let $a = [a_1, a_2, \ldots, a_n]$ and $b = [b_1, b_2, \ldots, b_n]$ be two arrays of length n where each a_i and b_i is a floating point number. The *dot product* of these arrays is defined to be $a_1b_1 + a_2b_2 + \ldots + a_nb_n$. Professor Lai knows that he can compute the dot product in $O(n)$ time by just taking the product a_ib_i for each $1 \leq i \leq n$ and then adding these terms up. However, he is interested in analyzing the following recursive implementation which took him only a few minutes to write.

```
float dot-product([a_1, a_2, ... , a_n], [b_1, b_2, ... , b_n])
{
  if (n == 1) return a_1 * b_1
  else
  {
     x = dot-product([a_1, ... , a_(n/2)], [b_1, ... , b_(n/2)]);
     y = dot-product([a_(n/2 + 1), ... , a_n], [b_(n/2 + 1), ... , b_n]);
     return x+y;
  }
}
```

(The notation `a_i` or `b_i` in the program corresponds to a_i or b_i.)

   (a) Set up a recurrence relation for this recursive program, solve it (showing all of your work), and give the running time in big-Oh notation as a function of n.

   (b) Asymptotically, how does the running time of this program compare to the standard $O(n)$ implementation?

2. **[15 Points] Sorting Variable-Length Items** Solve Exercise 8.3 of Cormen *et al.* second edition (page 179).

3. **[Optional 10 Points] More Recurrence Examples** Solve recurrence f and h in Exercise 4.1 of Cormen *et al.* second edition (page 85).