22C:31 Homework 4

Due in class on Thursday, April 8th

This homework will be graded out of 60 points and it is worth 6% of your grade. The Teaching Assistant will grade some 6 out of the 8 problems, with each problem worth 10 points.

---

1. Design an algorithm that takes two positive integers a and n, and computes a^n in $O(\log n)$ time. Think "divide and conquer."

2. This problem is on Strassen's multiplication.

   (a) While describing Strassen's Algorithm for matrix multiplication, I first suggested a simple divide-and-conquer algorithm whose running time $T(n^2)$ is given by the following recurrence:
   $$T(n^2) = 8T\left(\frac{n^2}{4}\right) + n^2.$$
   Solve this recurrence and show that the running time of this algorithm is asymptotically no better than that of the obvious algorithm that uses three nested loops

   (b) Strassen's mysterious observation led to a divide-and-conquer algorithm that had the following recurrence:
   $$T(n^2) = 7T\left(\frac{n^2}{4}\right) + n^2.$$
   Solve this recurrence.

3. In the algorithm for the closest point pair problem, we took a lot of care to ensure that the divide step and the combine step take $O(n)$ time each. Suppose that we were somewhat careless and instead, just sorted the points by x-coordinate and by y-coordinate on an "as needed" basis. Write a recurrence relation for the running time of this algorithm. Solve it to obtain a function $f(n)$ such that the running time of the algorithm is $\Theta(f(n))$.

4. The *maximum subsequence sum* problem takes as input a size-n array A of integers (the elements can be negative also) and finds values i and j, $1 \leq i \leq j \leq n$ such that
   $$\sum_{k=i}^{j} A[k]$$
   is maximized. Use *divide-and-conquer* to devise an $O(n \log n)$ time algorithm for this problem.

5. Problem 1 from Chapter 5.

6. Problem 2 from Chapter 5.

7. Problem 5 from Chapter 5.

8. Problem 6 from Chapter 5.
