ASSIGNMENT 3

Due at the start of class on Thursday, March 31.

1. Drawing a skyline.

Suppose you are given a description of rectangular, possibly overlapping buildings and you would like to draw the corresponding skyline with no overlapping lines. More concretely, a skyline is a list of tuples, each consisting of alternating $x$ coordinates and the heights connecting them. For ease of reading, we place bars over the entries representing heights. For example, the skyline $(1, \bar{4}, 3), (2, \bar{5}, 5), (4, \bar{3}, 5), (7, \bar{1}, 16), (9, \bar{3}, 11), (14, \bar{2}, 15)$ can be drawn as follows:

![Skyline diagram]

The desired output in this case is described by the drawing

![Skyline diagram]

with the corresponding skyline $(1, \bar{4}, 2, \bar{5}, 5), (7, \bar{1}, 9, \bar{3}, 11, \bar{1}, 14, \bar{2}, 15, \bar{1}, 16)$. Design a divide-and-conquer algorithm that, when given a skyline for $n$ separate buildings (i.e., a list of $n$ 3-tuples, not necessarily occurring in any particular order), produces the corresponding skyline of the drawing with overlapping lines removed. Prove that your algorithm is correct and analyze its running time. For full credit, your algorithm should run in time $O(n \log n)$.

2. Fast Fourier transform details.

(a) The discrete Fourier transform modulo $d$ is the $d \times d$ matrix with entries $F_{j,k} = \frac{1}{\sqrt{d}} \omega_d^{jk}$ for $j, k \in \{0, 1, \ldots, d - 1\}$, where $\omega_d = e^{2\pi i/d}$. Show that the discrete Fourier transform is unitary, i.e., its conjugate transpose is its inverse.

(b) In class, we discussed a polynomial multiplication algorithm based on the fast Fourier transform. This algorithm takes as input the coefficients of two polynomials of degree $n - 1$ and outputs the coefficients of their product, which is a polynomial of degree $2n - 2$. Write pseudocode for this algorithm, and show that your implementation runs in time $O(n \log n)$. 
3. **Maximum ordered ratio in linear time.**

Suppose you are given as input a sequence of positive numbers $a_1, a_2, \ldots, a_n$ with $n \geq 2$. Your goal is to find the largest ratio between two of these numbers where the numerator occurs after the denominator in the sequence. In other words, you would like to compute

\[
\max \left\{ \frac{a_i}{a_j} : i, j \in \{1, 2, \ldots, n\} \text{ with } i > j \right\}.
\]

Use dynamic programming to design a linear-time algorithm for this problem. Prove that your algorithm is correct and that its running time is $O(n)$.

4. **Commuting to optimize profit.**

Suppose you run a company with two offices, one in Washington and the other in San Francisco. Each week, you must choose where to work, and your choice will affect your profit: in week $i$, you will make $DC_i$ dollars if you work in Washington or $SF_i$ dollars if you work in San Francisco. While you clearly would like to work in the location with the higher profit, each flight from Washington to San Francisco (or vice versa) costs $1000. Given the lists $DC_1, \ldots, DC_n$ and $SF_1, \ldots, SF_n$, your goal is to find a schedule that maximizes your total profit, taking transportation costs into account. Since you live in Washington, your schedule must start and end there.

(a) An obvious greedy strategy is to always work in the office with the higher profit. Give a (minimal) example showing that this strategy is not always optimal.

(b) Design a polynomial-time algorithm that finds the maximum total profit. Justify your algorithm’s correctness and establish its running time. To get full credit for your solution, you should make your running time as small as possible.