Problem Set no. 2

Given: November 26, 2020
Due: December 16, 2020

Exercise 2.1 Let $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{R}^n$ and let $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{C}^n$ be such that $\mathbf{y} = DFT(\mathbf{x})$. Prove that $y_{n-j} = y_j^*$, for $j = 1, .., n-1$. (Here z^* is the *conjugate* of $z \in \mathbb{C}$, i.e., if $z = a + ib$, where $a, b \in \mathbb{R}$, then $z^* = a - ib$.)

Exercise 2.2 Let $\mathbf{x} = (f(0), f(\frac{1}{32}), \ldots, f(\frac{31}{32})) \in \mathbb{R}^{32}$, where $f(x) = \sin(11(2\pi x)) + 5\sin(7(2\pi x) + \frac{\pi}{4}) + 5\cos(7(2\pi x))$. Compute $DFT(\mathbf{x})$. (Hint: no complicated calculations are necessary. Use the relations $\cos x = \frac{1}{2}(e^{ix} + e^{-ix})$, $\sin x = \frac{1}{2i}(e^{ix} - e^{-ix})$, and the fact that Fourier basis is an orthonormal basis.)

Exercise 2.3 The *chirp transform* of a vector $(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{C}^n$, with respect to an arbitrary complex number $z \in \mathbb{C}$, is defined as follows: $y_k = \sum_{j=0}^{n-1} x_j z^{jk}$, for $k = 0, 1, \ldots, n-1$.

(a) Show that the DFT is a special case of the chirp transform. (For which z?)

(b) Use the relation $y_k = z^{k^2/2} \sum_{j=0}^{n-1} (x_j z^{j^2/2})(z^{-(k-j)^2/2})$ to express the chirp transform as a convolution. (Use caution. In the sum given, j ranges from 0 to $n-1$, for every value of k, while this is *n*ot the case for the non-cyclic convolution.)

(c) Show that the convolution of two vectors of length n can be computed in $O(n \log n)$ time for every value of n, not necessarily a power of 2.

(d) Use the previous results to show that the DFT of a vector of length n can be computed in $O(n \log n)$ time for any value of n.

Exercise 2.4 The *cross-correlation* of $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\mathbf{y} = (y_0, y_1, \ldots, y_{m-1})$ is defined to be $\mathbf{z} = (z_{-(m-1)}, \ldots, \ldots, z_{n-1})$ such that $z_k = \sum_j x_{j+k} y_j$. (The sum here is over j such that $0 \le j + k < n$ and $0 \le j < m$.) Show that the *cross-correlation* of two vectors of lengths n and m respectively, where $m \le n$, can be computed in $O(n \log m)$ time.

Exercise 2.5 Perform the multiplication $(4, 3, 2, 1)_{16} \times (8, 7, 6, 5)_{16}$ using one iteration of the algorithm of Schönhage and Strassen. (Note that $n = 16$ and $k = \log_2 n = 4$. The algorithm performs only one iteration of FFT computation. The two FFTs and FFT^{-1} are of size 4. 'Luckily' $\omega_4 = i$, so no irrational numbers and no approximations are required. The multiplications needed for the computation of the FFTs are done using the 'school method'. You do not need to give the details of these multiplications.)

Exercise 2.6 Let T be a text of length n and let P be a pattern of length m over a finite and small alphabet Σ. Let D be a $|\Sigma| \times |\Sigma|$ matrix such that $D(a, b)$ specifies the *similarity* or *dissimilarity* of $a, b \in \Sigma$. For every $k = 0, 1, .., n-m-1$ define $d_k = \sum_{j=0}^{m-1} D(T[k+j], P[j])$ to be the total pattern-text dissimilarity when the 0-th pattern character is aligned with the k-th text character.

(a) Show that d_k, for $k = 0, 1, .., n-m-1$, can be computed in $O(|\Sigma|\, n \log m)$ time.

(b) Suppose now that $\Sigma \subset \mathbb{Z}$, i.e., that each character is actually an integer, and that $D(a, b) = ab$, for every $a, b \in \Sigma$. How fast can the d_k's be computed?

(c) Suppose that we again have $\Sigma \subset \mathbb{Z}$ but this time $D(a, b) = (a - b)^2$, for every $a, b \in \Sigma$. How fast can the d_k's be computed?

Exercise 2.7 (a) Given a text T of length n and a pattern P of length m over the alphabet $\{0, 1, \ldots, m-1\}$ such that each character appears in P no more than c times, describe an algorithm that computes an array M of length n, where $M[k]$ is the number of matches when P is aligned with $T[k : k + m - 1]$. The running time of the algorithm should be $O(nc)$. (Hint: For each character in $T[i]$ in T, increment entries in M to which this character contributes a match.) (b) Combine the algorithm in (a) with an FFT-based algorithm to obtain an algorithm that computes the array M in $O(n\sqrt{m \log m})$ for *any* pattern P of length m. (Hint: Treat separately characters that appear many times in P and characters that appear only a few times in P.)