ESO207A: Data Structures and Algorithms

Practice Set *1b: Computing Convolutions* –

Recall the definition of convolution. Given two vectors $a = (a_0, a_1, \ldots, a_{n-1})$ and $b = (b_0, b_1, \ldots, b_{n-1})$, the convolution $c = a \otimes b$ is defined as a vector of dimension $2n-1$, where,

$$c_k = \sum_{i=0}^{k} a_i b_{k-i}, \quad k = 0, 1, \ldots, 2n-2$$

where, for purposes of calculating c_k, it is assumed that $a_n, a_{n+1}, \ldots, a_{2n-2} = 0$ and $b_n, b_{n+1}, \ldots, b_{2n-2} = 0$. Thus,

$$a \otimes b = (a_0 b_0, a_0 b_1 + a_1 b_0, a_0 b_2 + a_1 b_1 + a_2 b_2, \ldots, a_{n-2} b_{n-1} + a_{n-1} b_{n-2}, a_{n-1} b_{n-1}) \ .$$

Alternatively, $c_k = \sum_{\substack{(i,j): i+j=k \\ i<n, j<n}} a_i b_j, k = 0, 1, \ldots 2n-2$. Convolution is easily generalized to vectors of different lengths. For $a = (a_0, \ldots, a_{m-1})$ and $b = (b_1, b_2, \ldots, b_{n-1})$, $c = a \otimes b$ is a vector of dimension $m + n - 1$ defined as

$$c_k = \sum_{\substack{(i,j): i+j=k \\ 0 \le i \le m \\ 0 \le j \le n}} a_i b_j \ .$$

1. Consider the classical Gaussian smoothing of a vector. Let $a = (a_0, a_1, \ldots, a_{n-1})$ be an n-dimensional vector and assume that references to $a_{-k}, a_{-(k-1)}, \ldots, a_{-1}$ and $a_n, \ldots, a_{n+k-1}$ all return 0. You would like to compute the vector $a' = (a'_0, \ldots, a'_{n-1})$ defined as follows.

   $$a'_i = \frac{1}{C} \sum_{j=i-k}^{i+k} a_j e^{-(i-j)^2},$$

   where, k is some fixed width parameter and C is a constant chosen to normalize the sum (for e.g., $C = \sum_{j=-k}^{k} e^{-j^2/2}$). Give an $O(n \log n)$ time algorithm to compute a' assuming that individual arithmetic operations can be computed in $O(1)$ time.

2. Given two n-dimensional vectors $a = (a_0, a_1, \ldots, a_{n-1})$ and $b = (b_0, b_1, \ldots, b_{n-1})$, a wrap-around convolution is defined as an n-dimensional vector $c = (c_0, c_1, \ldots, c_{n-1})$ whose coordinates are defined as follows.

   $$c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_k b_0 + a_{k+1} b_{n-1} + a_{k+2} b_{n-2} + \ldots + a_{n-1} b_{k+1}$$

   Show how to evaluate this transform in $O(n \log n)$ time by viewing it as a convolution. (*Hint:* Express $\sum_{j=0}^{k} a_j b_{k-j}$ and $\sum_{j=k+1}^{n-1} a_j b_{n+k-j}$ as convolutions and add them.)

3. **CLRS 30.2-8** The *chirp transform* of a vector $a = (a_0, a_1, \ldots, a_{n-1})$ is the vector $y = (y_0, y_1, \ldots, y_{n-1})$ where, $y_k = \sum_{j=0}^{n-1} a_j z^{kj}$ and z is any complex number. The DFT is a

special case of the chirp transform obtained by taking $z = \omega_n$. Show how to evaluate the chirp transform in time $O(n \log n)$ for any complex number z. (*Hint:* Use the equation

$$y_k = z^{k^2/2} \sum_{j=0}^{n-1} \left(a_j z^{j^2/2}\right) \left(z^{-(k-j)^2/2}\right)$$

to view the chirp transform as a convolution.