ESO207A: Data Structures and Algorithms

Practice Set *1b: Solutions* –

Recall the definition of convolution. 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 some normalization constant. Give an $O(n \log n)$ time algorithm to compute a' assuming that individual arithmetic operations can be computed in $O(1)$ time.

Solution. Define the vector

$$w = (w_{-k}, w_{-(k-1)}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{k-1}, w_k)$$

and let $w_j = e^{-j^2}$. So the vector $w = (e^{-k^2}, e^{-(k-1)^2}, \ldots, e^{-1}, 1, e^{-1}, \ldots, e^{-(k-1)^2}, e^{-k^2})$. So, with this notation,

$$a'_i = \frac{1}{C} \sum_{j=i-k}^{i+k} a_j w_{i-j} = \frac{1}{C} \sum_{s=-k}^{k} a_{i+s} w_s \ .$$

Define a vector b of dimension $2k+1$ such that $b_l = w_{k-l}$. The above sum can now be written as

$$\begin{aligned} a'_i &= a_{i+k} b_0 + a_{i+k-1} b_1 + \ldots + a_i b_k + \ldots + a_{i-k} b_{2k} \\ &= \sum_{(j,l): j+l=i+k} a_j b_l \ . \end{aligned}$$

This is essentially a convolution $a \otimes b$.

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.)

Solution. We will write the expression of c_k as the sum of two convolutions. Let $c_k = c'_k + d_k$, where, $c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_k b_0$, and $d_k = a_{k+1} b_{n-1} + a_{k+2} b_{n-2} + \ldots + a_{n-1} b_{k+1}$. Clearly, c' is a convolution $c' = a \otimes b$. Consider, $d_k = a_{k+1} b_{n-1} + a_{k+2} b_{n-2} + \ldots + a_{n-1} b_{k+1}$. Let $b'_j = b_{n-1-j}$ and $a'_j = a_{n-1-j}$ and $d'_{n-2-k} = d_k$. Then,

$$d_{n-2-k} = a'_{n-k-2} b'_0 + a'_{n-k-3} b_1 + \ldots + a'_0 b'_{n-k-2}$$

In other words, letting $u = n - 2 - k$

$$d'_u = a'_u b'_0 + a'_{u-1} b'_1 + \ldots + a'_0 b'_u$$

Thus, $d' = a' \otimes b'$. From d', d is easily calculated. Since convolutions are computed in $O(n \log n)$ time using FFT, the vector d can be computed in time $O(n \log n)$.

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.

Solution. We are given $a = (a_0, \ldots, a_{n-1})$ and a complex number z and we want y as defined above.

Following the hint above, define a vector f by $f_j = a_j z^{j^2/2}$ and the vector g by $g_l = z^{-l^2/2}$. Then,

$$\begin{aligned}(f \otimes g)_k &= \sum_j f_j g_{k-j} \\ &= \sum_j a_j z^{j^2/2} \cdot z^{-(k-j)^2/2} \\ &= \sum_j a_j z^{-k^2/2 + kj} \\ &= z^{-k^2/2} \sum_j a_j z^{kj} \quad .\end{aligned}$$

So $y_k = z^{k^2/2} (f \otimes g)_k$ is the chirp transform. Suppose we can calculate the vectors f and g in time $O(n \log n)$. Then, $f \otimes g$ is computed using FFT and its inverse in time $O(n \log n)$, which can then be multiplied by $z^{k^2/2}$ in time $O(n)$.

We show how to compute the vector h defined as $h_j = z^{j^2/2}$ in $O(n)$ time. The same ideas can be used to compute f and g in time $O(n)$. Compute $z^{1/2}$ and z^j, $j = 0, 1, \ldots, n$. Now $z^{j^2/2} = z^{((j-1)^2 + 2j + 1)/2} = z^{(j-1)^2/2 + j + 1/2} = z^{(j-1)^2/2} \cdot z^j \cdot z^{1/2}$. Thus, $z^{j^2/2}$ can be computed inductively from $z^{(j-1)^2/2}$ and z^j in time $O(1)$.

Caveat: We have made an assumption that $z^{j^2/2}$ is not large, this holds if $|z| \leq 1 + \frac{O(1)}{n^2}$.