Advanced Algorithms

Whenever you give an algorithm, also argue for its running time and correctness.

Convolution and FFT

1. The company Zamazon occasionally runs contests: If you can put exactly k items into your cart to get a total price of exactly T, then you don't have to pay at checkout. Zamazon sells n different items. You crawled the Zamazon website and now have a list of all items and their prices.

   (a) In this part of the exercise, assume that the item prices are real numbers. Give an algorithm that finds k items of total price T or outputs correctly that this is not possible. Multiple copies of the same item are allowed. You may spend $O(k \cdot n^{\lceil k/2 \rceil} \log n)$ arithmetic operations.

   (b) In this part of the exercise, assume that the item prices are integers smaller than $1000n$. Give an algorithm that decides correctly whether there are k items of total price T, in $O(k \cdot n \log n)$ arithmetic operations. Multiple copies of the same item are allowed.

2. The Fourier transform of a polynomial and its inverse change between the coefficient representation

   $$p(x) = a_0 + \ldots + a_{d-1}x^{d-1}$$

   of a polynomial p and its point-value representation

   $$(x_0, p(x_0)), \ldots, (x_{d-1}, p(x_{d-1}))$$

   at evaluation points $x_i = \omega_i^{(d)}$ for $i = 0, \ldots, d-1$. The FFT algorithm shows that either of these representations can be computed from the other using $O(n \log n)$ arithmetic operations. In this exercise, we show that even for arbitrarily chosen evaluation points x_i in the point-value representation, the coefficient representation can still be computed from the point-value representation—albeit at higher computational cost. That is, given a list of point-value pairs

   $$(x_0, y_0), \ldots, (x_d, y_d),$$

   with $x_i \neq x_j$ for $i \neq j$, show how to determine the coefficients $a_0, \ldots, a_d$ of a polynomial p of degree d such that $p(x_i) = y_i$ holds for all $i = 0 \ldots d$. You may use that the matrix

   $$\begin{pmatrix} x_0^0 & \ldots & x_0^d \\ \vdots & \ddots & \vdots \\ x_d^0 & \ldots & x_d^d \end{pmatrix}$$

   has full rank whenever $x_0 \ldots, x_d$ are pairwise distinct numbers. Which running time do you achieve?

3. Let n be a power of two, and let $\omega_0, \ldots, \omega_{n-1} \in \mathbb{C}$ be the n-th roots of unity, where

   $$\omega_k = \cos(2\pi \cdot k/n) + i \sin(2\pi \cdot k/n).$$

   Consider the matrix

   $$D = \begin{pmatrix} \omega_0^0 & \ldots & \omega_0^{n-1} \\ \vdots & \ddots & \vdots \\ \omega_{n-1}^0 & \ldots & \omega_{n-1}^{n-1} \end{pmatrix}.$$

   It can be checked that the columns $D_{\star,1}, \ldots, D_{\star,n}$ are a basis for the vector space $\mathbb{C}^n$, the so-called *Fourier basis*. Because the columns $D_{\star,1}, \ldots, D_{\star,n}$ are a basis, every vector $v \in \mathbb{C}^n$ has a unique representation in this basis. This means that there is a vector of coefficients $\hat{v} \in \mathbb{C}^n$ such that

   $$v = \sum_{j=1}^{n} \vec{v}(j) \cdot D_{\star,j}.$$

(a) Given as input v, compute the matrix-vector product Dv with $O(n \log n)$ arithmetic operations. It will help to look again at the previous exercise. (Note that computing Av for general matrices A takes at least $O(n^2)$ operations. The particular structure of the matrix D shows that this is not necessary; it is not even needed to provide D as part of the input.)

(b) Given as input v, compute the vector $\hat{v}$ with $O(n \log n)$ arithmetic operations.

(c) Bonus: Give a description of (the real part of) column $D_{\star,k}$ for $0 \leq k \leq n-1$ as an audio signal. What does D having full rank mean in terms of audio signals? What does the k-th entry of $\hat{v}$ mean in terms of audio?

Algorithms using matrix multiplication

1. In this exercise, we extend the triangle detection algorithm from the lecture. Let $G = (V, E)$ be an undirected graph on n vertices. Recall that ω denotes the optimal exponent of matrix multiplication. You may use a subroutine that multiplies $t \times t$ matrices in time $O(t^\omega)$.

   (a) A *triangle* is a vertex subset $T \subseteq V$ of size 3 such that $uv \in E$ for all $u, v \in T$ with $u \neq v$. Give an algorithm that *counts* triangles in $O(n^\omega)$ time.

   (b) A *k-clique* is a vertex subset $T \subseteq V$ of size k such that $uv \in E$ for all $u, v \in T$ with $u \neq v$. Give an algorithm that decides whether G contains a k-clique in $O(n^{\omega\lceil k/3 \rceil})$ time. (Hint: Split the clique into three equal-sized parts.)

   (c) Like above but *count* the cliques.