COL351: Analysis and Design of Algorithms

Tutorial Sheet - 8

October 15, 2022

Question 1 Given a sorted array of distinct integers $A[1,\ldots,n]$, you want to find out if there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.

Question 2 Let $A = \{a_1, a_2, \ldots, a_n\}$ be an input set of positive integers in range $[1, M]$. Your task is to compute the set $S = \{x + y + z \mid x, y, z \in A\}$ of triplet sums.

Show that if two polynomials of degree n can be multiplied in $O(n \log n)$ time, then the set S is computable in $O(M \log M)$ time.

Question 3 Prove the following:

(i) If ω is N^{th} primitive root of unity then $\{1, \omega, \omega^2, \ldots, \omega^{N-1}\}$ are distinct roots of $x^N - 1 = 0$.

(ii) If ω is N^{th} primitive root of unity then for every $1 \leqslant i \lneq N$, we have

$$1 + \omega^i + \omega^{2i} + \cdots + \omega^{i(N-1)} = 0.$$

(iii) If ω is a non-primitive N^{th} root of unity then there exists an $1 \leqslant i \lneq N$ satisfying

$$1 + \omega^i + \omega^{2i} + \cdots + \omega^{i(N-1)} \neq 0.$$

(iv) If ω is N^{th} primitive root of unity and N is power of two, then for any $1 \leqslant i \lneq N$, ω^i is N^{th} primitive root of unity if and only if i is odd. Use this to argue that ω^{-1} is also N^{th} primitive root of unity.

Question 4 Consider the following matrix of size $N \times N$.

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^j & \cdots & \omega^{N-1} \\ 1 & & & & & & \\ \vdots & & & & & & \\ 1 & \omega^i & (\omega^i)^2 & \cdots & (\omega^i)^{\,j} & \cdots & (\omega^i)^{\,N-1} \\ \vdots & & & & & & \\ 1 & & & & & & \end{bmatrix}$$

Prove that the matrix is non-invertible if ω is a non-primitive N^{th} root of unity.

Question 5 Suppose you have to multiply two n bit positive integers x and y. Observe that x can be written as $x_1 \cdot 2^{n/2} + x_0$, for appropriate x_0, x_1. Similarly, y can be written as $y_1 \cdot 2^{n/2} + y_0$, for appropriate y_0, y_1. Then,

$$\begin{aligned} xy &= x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{n/2} + x_0 y_0 \\ &= x_1 y_1 \cdot 2^n + (p - x_1 y_1 - x_0 y_0) \cdot 2^{n/2} + x_0 y_0 \end{aligned}$$

where, $p = (x_1 + x_0)(y_1 + y_0)$. Use this observation to design an $O(n^{\log_2 3}) = O(n^{1.59})$ time algorithm to multiply two n bit positive integers.