COL351: Analysis and Design of Algorithms

Tutorial Sheet - 9

October 22, 2022

Question 1 Compute the DFT of polynomial $x^3 + x^2 + 2x + 1$ using both FFT algorithm and by pre-multiplying the associated vector by Vandermonde matrix. Verify that the results are identical. (You should take ω to be $0 + i$).

Question 2 The hamming distance of two n-length arrays A, B is the number of positions where they mismatch, that is,

$$Ham\text{-}Dist(A, B) = \sum_{\substack{i \in [1,n] \\ A[i] \neq B[i]}} 1.$$

The cyclic shift M^i of an array M by a value $i < n$ is defined to be the concatenated array

$$M^i := M[i+1, n] \cdot M[1, i].$$

Design an $O(n \log n)$ time algorithm that given two n-length binary arrays A, B, finds an i in range $[0, n-1]$ for which $Ham\text{-}Dist(A, B^i)$ is minimized.

Question 3 Let A be an array of size n with integer entries. Design an $O(n)$ time algorithm to check if there exists an $x \in A$ such that count of x in A is at least $\lceil n/10 \rceil$.

Question 4 Let A be an array of size n. Your task is to decide if there is an element which is present more than $n/2$ times. The only operation by which you can access the elements of A is a function f, which given two indices i and j, outputs whether the objects at positions i and j in the array are identical or not. Design an $O(n \log n)$-time algorithm for this (where each call to f can be assumed to take $O(1)$ time).

Hint: Divide A into equal sub-arrays, and use divide and conquer: (i) If both sub-arrays do not have a majority then the original array does not have a majority element (why?) (ii) If either of the two parts returns a majority, then we need to use f to compare it with all other elements of A.

Question 5

1. Without using Master's theorem prove that the recurrence $T(n) = 2T(n/2) + n \log n$ satisfies the relation $T(n) = O(n \log^2 n)$. You can assume that $T(0), T(1)$ are some positive constants.

2. Suppose the time-complexity of an algorithm follows the recurrence relation $T(n) = T(n/2) + T(n/4) + 1$. Use recursion trees to prove that $\sqrt{n} \leqslant T(n) \leqslant n$.

   Hint: If we only look at complete levels, we find that the level sums form an ascending geometric series $T(n) = 1 + 2 + 4 + \cdots$, so the solution is dominated by the number of leaves. The recursion tree has $\log_4 n$ complete levels, so $T(n) \geqslant 2^{\log_4 n} = n^{\log_4 2} = \sqrt{n}$. On the other hand, every leaf has depth at most $\log_2 n$, so the total number of leaves is at most $2^{\log_2 n} = n$.