Problem 1: Majority element

An array $A[1 \ldots n]$ is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form “is $A[i] > A[j]$?” (Think of the array elements as GIF files, say.) However you can answer questions of the form: “is $A[i] = A[j]$?” in constant time.

(a) Show how to solve this problem in $O(n \log n)$ time. (Hint: Split the array $A$ into two arrays $A_1$ and $A_2$ of half the size. Does knowing the majority elements of $A_1$ and $A_2$ help you figure out the majority element of $A$? If so, you can use a divide-and-conquer approach.)

(b) Can you give a linear-time algorithm? (Hint: Here’s another divide-and-conquer approach:

- Pair up the elements of $A$ arbitrarily, to get $n/2$ pairs
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them.

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if and only if $A$ does.)

Problem 2: Find the missing integer

An array $A[1 \ldots n]$ contains all the integers from 0 to $n$ except one. It would be easy to determine the missing integer in $O(n \log n)$ time by using an auxiliary array $B[0 \ldots n]$ to record which numbers appear in $A$. In this problem, however, we cannot access an entire integer in $A$ with a single operation. The elements of $A$ are represented in binary, and the only operation we can use to access them is “fetch the $j$th bit of $A[i]$,” which takes constant time. Show that if we use only this operation, we can still determine the missing integer in $O(n)$ time. Do not forget to give the recurrence with ample justification.

Problem 3: Hadamard matrix

For a nonnegative integer $k$, the Hadamard matrix $H_k$ is defined as follows:

- $H_0$ is the $1 \times 1$ matrix (1).
• For $k > 0$, $H_k$ is the $2^k \times 2^k$ matrix

$$H_k = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}.$$ 

Show that if $\vec{v}$ is a column vector of length $n = 2^k$, then the matrix-vector product $H_k \vec{v}$ can be calculated in $O(n \log n)$ arithmetic operations. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time.

**Problem 4: Toom–Cook algorithm**

(20 = 10 + 10 Points)

In the class we have studied Karatsuba’s algorithm for integer multiplication by dividing the integers into two equal parts. The time complexity of this algorithm is $\Theta(n^{\log_2 3})$. We further studied the Toom-3 algorithm for integer multiplication, in which the integers to be multiplied are split into three equal parts, and showed that 9 multiplications can be reduced to 5 multiplications. The time complexity of Toom-3 algorithm is $\Theta(n^{\log_3 5})$. In this question, we will generalize Toom-3 algorithm to $r$ equal split of the integers to be multiplied. This generalization is called the Toom-Cook algorithm.

(a) Generalize Toom-3 algorithm for the case of $r$ equal split of the integers. Formulate the recurrence.

(b) Solve the recurrence to give the time complexity of Toom-Cook algorithm.

**Problem 5: More recurrences**

(15 = 5 +5 + 5 Points)

Solve these recurrences exactly to obtain closed-form solutions. Please do not give asymptotic bounds. When necessary, you can assume $n$ is exact power of 2. When initial conditions are not given (in (c)), you can use arbitrary constants.

(a) $T(n) = n(T(n/2))^2, T(1) = 1/3$

(b) $T(n) = T(n-1) + n^2, T(1) = 1$

(c) $T(n) = 2T(n/2) + n \log n$

**Problem 6: Polynomial multiplication by FFT**

(15 Points)

Suppose that you want to multiply two polynomials $x + 1$ and $x^2 + 1$ using the FFT. Choose an appropriate power of two, find the FFT of the two sequences, multiply the results componentwise, and compute the inverse FFT to get the final result. In your write-up, show all the steps involved.

**Problem 7: Cartesian sum**

(10 Points)

Consider two sets $A$ and $B$, each having $n$ integers in the range from 0 to 10$n$. We wish to compute the Cartesian sum of $A$ and $B$, defined by

$$C = \{ x + y \mid x \in A \land y \in B \}.$$

Note that the integers in $C$ are in the range from 0 to 20$n$. We want to find the elements of $C$ and the number of times each element of $C$ is realized as a sum of elements in $A$ and $B$. Show that this problem can be solved in $O(n \log n)$ time. (Hint: Represent $A$ and $B$ as polynomials of degree at most $10n$.)