IITM-CS2800-2025 : Design and Analysis of Algorithms

Practice Problem Sheet #3 Released On : Feb 13, 1PM

Part A (to be done during the tutorial hours)

1. Solve the following recurrence relations.
   (a) $T(n) = T(n/2) + 2T(n/3) + 3T(n/4) + n^2$.
   (b) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

2. We saw a linear time selection algorithm in class which is based on splitting the array into arrays of 5 elements each. Suppose we split the array into arrays of (a) 7 elements each and (b) 3 elements each. Derive a recurrence for the running time in each case, and write down asymptotic bounds.

3. You are given two sets of n points, one set $P = \{p_1, p_2, \ldots, p_n\}$ and another set $Q = \{q_1, q_2, \ldots, q_n\}$. Create a set of n line segments by connecting each point p_i to the corresponding point q_i. The task is to determine how many of the pairs of lines actually intersect. We consider this problem in two scenarios.
   (a) All points in the set P are on the line $y = 0$ and all points in line Q are on the line $y = 1$. Describe an algorithm that runs in time $O(n \log n)$ (Hint : You have seen this in class !).
   (b) All the poins in P and Q are on the *unit circle*. Describe and analyze a divide-and-conquer algorithm to determine how many pairs of these line segments intersect in $O(n \log^2 n)$ time.

4. An array $A[1, 2, \ldots, n]$ is said to have a majority element if strictly more than half of the elements are the same. Observe that there will exist at most one majority element in an array. The entries of the array need not necessarily be integers, and hence you cannot compare the elements. You are only allowed queries of the form "Is $A[i] = A[j]$ "?
   (a) Consider the following divide-and-conquer algorithm: First divide the array into two parts $A[1, 2, \ldots, n/2]$ and $A[n/2 + 1, \ldots, n]$ and find their majority element. How does finding the majority element of these two arrays help us obtain the majority element of A, if it exists. Write a pseudocode, and argue that your algorithm runs in time $O(n \log n)$.
   (b) Here is another divide-and-conquer strategy: First, pair the elements into groups of two arbitrarily. For each group of two, if both the elements are different then discard them. Otherwise, keep just one copy. Report the majority of the resulting list of

size at most half of the original. Show (1) termination (2) correctness (3) run-time analysis for this algorithm.

5. Is it asymptotically faster to square an n digit number than multiplying two n digit numbers? If you think it is true, give an algorithm. If you feel that it is incorrect, justify your answer.

Part B

1. The *Hadamard matrices*, denoted by $H_0, H_1, H_2, \ldots$, are defined as follows:

   - $H_0 = [1]$ is a 1×1 matrix.
   - H_k is a $2^k \times 2^k$ matrix defined as follows.

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

   (a) Write down the matrix H_3.

   (b) For $n = 2^k$, let $\mathbf{x}$ be an $n \times 1$ vector. Give an algorithm, that given k and x to obtain the product $H_k\mathbf{x}$. Note that the naive approach gives an $O(n^2)$-time algorithm.

   (c) Can you improve this using a divide-and-conquer algorithm. *(Note : The input only consists of k and x. The matrix H_k, which has n^2 entries, is not given as part of the input. Since you are aiming to improve an $O(n^2)$ algorithm, you cannot compute all entries of the matrix H_k. So your algorithm has to use the structure of H_k as shown in the above picture.)*

2. Given two numbers a and b, we would like to compute the greatest common divisor (gcd) of two positive integers: the largest integer which divides them both. We would like to solve this by using divide and conquer.

   (a) Show that the following rule is true:

   $$\gcd(a,b) = \begin{cases} 2 \cdot \gcd\left(\frac{a}{2}, \frac{b}{2}\right) & \text{if } a, b \text{ are even,} \\ \gcd\left(a, \frac{b}{2}\right) & \text{if } a \text{ is odd, } b \text{ is even,} \\ \gcd\left(\frac{a-b}{2}, b\right) & \text{if } a, b \text{ are odd.} \end{cases}$$

   (b) Give a divide-and-conquer algorithm for computing the greatest common divisor (gcd) based on the above. Write down the recurrence relation.