IITM-CS2800-2025 : Design and Analysis of Algorithms

Practice Problem Sheet #2 Released On : Feb 6, 1pm

1. Solve the following recurrence relations using Master's Theorem whever possible. Otherwise use an unfolding strategy.
   (a) $T(n) = 5T(n/2) + O(n)$
   (b) $T(n) = 4T(n/2) + O(n^2)$
   (c) $T(n) = 2T(n/4) + O(\sqrt{n})$
   (d) $T(n) = 5T(n/3) + O(n)$
   (e) $T(n) = T(\frac{n}{2}) + T(\frac{2n}{3}) + n$ and $T(1) = 1$
   (f) $T(n) = 32T(n/2) + n^6$ and $T(1) = 1$

2. Given A, B which are both $n \times n$ matrices. We want to compute the product matrix $C = A \times B$. We will learn a trick due to Strassen, similar in nature to Karastuba's trick, and a trick the shocked the world by a brand new algorithm for this problem.
   (a) Write down the algorithm that you have implemented in CS1100. Prove correctness and analyse it. Prove that
   (b) Design a divide and Conquer algorithm for matrix multiplication. Divide each of the 2 matrices into 4 parts each as follows.

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ and } \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \text{and the prdocuct matrix is } \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

   where A_{ij} and B_{ij} are $\frac{n}{2} \times \frac{n}{2}$ size matrices. Now, write down the product C of the A and B in terms of the smaller submatrices. Write this down as an algorithm and obtain a recurrence for it as $T(n) = aT(n/b) + f(n)$. Show that this gives again an $O(n^3)$ algorithm for the problem.
   (c) Verify the following the magical identity given by Strassen: The product matrix is

$$\begin{array}{rclcrcl} C_{11} & = & P_1 + P_4 - P_5 + P_7 & & C_{12} & = & P_3 + P_5 \\ C_{21} & = & P_2 + P_4 & & C_{22} & = & P_1 + P_3 - P_2 + P_6 \end{array}$$

   where

$$\begin{array}{rclcrcl} P_1 & = & (A_{11} + A_{22})(B_{11} + B_{22}) & & P_5 & = & (A_{11} + A_{12})B_{22} \\ P_2 & = & (A_{21} + A_{22})B_{11} & & P_6 & = & (-A_{11} + A_{21})(B_{11} + B_{12}) \\ P_3 & = & A_{11}(B_{12} - B_{22}) & & P_7 & = & (A_{12} - A_{22})(B_{21} + B_{22}) \\ P_4 & = & A_{22}(-B_{11} + B_{21}) & & & & \end{array} .$$

(d) Explain why if you implement the above in a recursive algorithm, the recurrence for the running time will be:

$$T(n) = 7T(n/2) + O(n^2)$$

And conclude (using masters theorem) that, $T(n) = \Theta(n^{\log_2 7}) = \Theta(n^{2.8073})$ which is faster than $O(n^3)$ algorithm.

3. **Tower of Hanoi:** You are given n disks of differing diameter, arranged in increasing order of diameter from top to bottom on the leftmost of three pegs. You are allowed to move a single disk at a time from one peg to another, but you must not place a larger disk on top of a smaller one. Your task is to move all disks to the rightmost peg via a sequence of these moves.
   (a) Solve the above problem using a recursive streategy and find how many moves the algorithm takes.
   (b) Show that the following algorithm solves this problem. Think of the pegs as being arranged in a circle, with clockwise moves being from peg 1 to peg 2 to peg 3 to peg 1. If n is odd, then start by moving the smallest disk one jump in the counterclockwise direction. Alternate between moving the smallest disk in this manner and the only other legal move available. If n is even, then start by moving the smallest disk one jump in the clockwise direction. Alternate between moving the smallest disk in this manner and the only other legal move available.

4. **Power Computation:** Given a number a, and an integer k, we want to compute a^k.
   (a) Design a trivial iterative algorithm for finding a^k. Analyse the running time (number of multiplications) in terms of k.
   (b) Can you do this by recursive approach? Design one by recursively computing $a^{k/2}$. First assume that k is a power of 2. Appropriately design this algorithm and write down a recurrence relation and solve it bound the running time. What happens when k is not a power of 2?

5. **Computing the power of 2:** Given n, we want write down the natural number 2^n.
   (a) Write down an algorithm that runs in time $O(n^2)$.
   (b) Describe and analyze a variant of Karatsuba's algorithm that multiplies any m-digit number and any n-digit number, for any $n \geq m$, in $O(nm^{\log_2 3-1})$ time.
   (c) Using the above algorithm (or directly using Karastuba's algorithm as a subroutine), improve the algorithm above that solves the problem of computing 2^n problem in in $O(n^{\log_2 3})$ time.

6. **Iterative Square Root Depth:** Given an integer n, the iterative square root depth is defined as the number of times the square root operation needs to be applied iteratively until the result is at most 1. Come up with a Divide and Conquer solution for this problem. For an integer n, what would be a tight asymptotic bound on the iterative square root depth?