Design and Analysis of Algorithms	**September, 2024**
Practice Set 1	
M.Tech CS, 2024-25	*Lecturer: Abhranil Chatterjee and Anupa Sunny*

Sorting and Searching

1. Consider a building with infinitely many floors. You need to find the highest floor h from which an egg can be dropped without breaking. Can you do it with $O(\log h)$ egg droppings?

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

3. Let $\langle a_1, \ldots, a_n \rangle$ be a sequence of n distinct numbers. We say that two indices $i < j$ form an *inversion* if $a_i > a_j$. Design and analyze an efficient algorithm to find out the number of inversions in A.

4. Given two sorted integer arrays (with all distinct numbers in them) of size n, we want to find the median of the union of the two arrays. Can you find it by accessing only $O(\log n)$ entries in the two arrays.

5. Given an array with n positive integers $[a_1, a_2, \ldots, a_n]$ and a target value S, find the minimum length subarray whose sum is at least S. Can you do it in $O(n \log n)$ time?

6. Prove that $\log n! = \Theta(n \log n)$.

7. Note that $\log n! = \log \left(\prod_{i=1}^{n} i\right) = \sum_{i=1}^{n} \log i \geq \int_1^n \log x \, dx$. Solve the integral to get a lower bound.

8. Given an integer a, check if it is of the form b^k for some unknown integers b and $k > 1$. Can you do this in time $O(\log^3 a)$?

9. True or false:

   - $2n + 3$ is $O(n^2)$.
   - $\sum_{i=1}^{n} i^2$ is $O(n^2)$.
   - $\sum_{i=1}^{n} 1/i$ is $O(\log n)$.
   - n^n is $O(2^n)$.
   - 2^{3n} is $O(2^n)$.

10. Show that any sequence of n integers can be sorted in $O(n + M)$ time, where

$$M = \max_i x_i - \min_i x_i.$$

For small M , this is linear time: why doesnt the $\Omega(n \log n)$ lower bound apply in this case?

11. Is binary search optimal? Justify your answer.

12. Given a natural number N, write an algorithm to check if it is a perfect square or not.

13. Consider an $n \times n$ matrix M where every row and column is sorted in increasing order. Given a number x, design an efficient algorithm to find the position of x in M.

14. Show that there is no comparison sort whose running time is linear for at least half of the $n!$ inputs of length n.

Divide and Conquer Technique

1. Let us try to apply the divide and conquer approach on the integer multiplication problem. Suppose we want to multiply two n-bit integers a and b. Expand the product in the base $2^{n/2}$. The Karatsuba algorithm computes the coefficients in the $2^{n/2}$ base expansion using *only three multiplications* of $n/2$ bit integers and a few additions/subtractions and shift operations? Complete the details of the algorithm and analyze the running time.

2. Can you find square of an n-bit integer a, using square subroutine on $2k - 1$ integers with n/k bits and a some additions/subtractions? Whats the running time you get? What if you take k as something like $n/2$? Does that give you a really fast algorithm?

3. Consider arbitrary eight numbers a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4. Define these seven expressions.

$$p_1 = (a_1 + a_4)(b_1 + b_4), \quad p_2 = (a_3 + a_4)b_1, \quad p_3 = a_1(b_2 - b_4), \quad p_4 = a_4(b_3 - b_1),$$
$$p_5 = (a_1 + a_2)b_4, \quad p_6 = (a_3 - a_1)(b_1 + b_2), \quad p_7 = (a_2 - a_4)(b_3 + b_4)$$

   (a) Define the following four terms:

$$q_1 = p_1 + p_4 - p_5 + p_7, \quad q_2 = p_3 + p_5$$
$$q_3 = p_2 + p_4, \quad q_4 = p_1 - p_2 + p_3 + p_6$$

We now consider the matrix multiplication algorithm. Given two $n \times n$ matrices A and B, their product C can be computed in $O(n^3)$ time. Recall that, a natural way to split any matrix as a 2×2 square matrix is following:

$$A = \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right].$$

Use this block decomposition to write the product in terms of A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4.

(b) Use the identities defined, to compute the product in $O(n^{\log_2 7})$ time.

4. A univariate polynomial $P(x)$ of degree $< n$ is of form:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}.$$

(a) How can you represent the polynomial in terms of coefficients and in terms of evaluation points?

(b) For any distinct n numbers α_0, α_1, $\ldots$, α_{n-1}, a *Vandermonde matrix* is defined as the following $n \times n$ matrix.

$$V = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \alpha_2 & \ldots & \alpha_{n-1} \\ \alpha_0^2 & \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_{n-1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{n-1} & \alpha_1^{n-1} & \alpha_2^{n-1} & \ldots & \alpha_{n-1}^{n-1} \end{bmatrix}$$

Show that V is invertible.

(c) Design an $O(n^3)$-time algorithm using the Vandermonde matrix to change one polynomial representation to the other.

(d) How can you choose the Vandermonde matrix wisely to improve the running time to $O(n \log n)$ [FFT algorithm].

(e) Suppose we want to convert the coefficient representation of $P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ to the evaluation vector representation evaluated at the fourth roots of unity $1, -1, i, -i$. Consider how FFT algorithm will compute these. We will represent the algorithm as a circuit, with the a gate labeled with α takes two numbers a and b as inputs and outputs two numbers $a + \alpha b$ and $a - \alpha b$. Write down the FFT circuit for the computation of the evaluation vector of the degree 3 polynomial at the 4th roots of unity.

5. Prove the Master theorem.