National Institute of Technology Karnataka
Department of Computer Science and Engineering
CS-700 Algorithms and Complexity

Assignment Sheet - 2 **Due Date:** 30 September 2024

Instructions:

- Do not solve the problems using Brute Force approach. Your algorithm should run in best known time complexity unless a different complexity is explicitly asked.
- If any specific methods/approaches are recommended for solving the problem, you should use those in solving the problem.
- Provide a detailed explanation of what algorithmic principle (eg: divide and conquer) is used and how it works.
- Include comments and documentation within your code to explain key parts of the implementation.

Problems:

1. Given a set of strings, find the longest common prefix.
   Example:
   - **Input**: {"apple", "ape", "april"}
   - **Output**: "ap"
2. Give an $O(n^2)$ algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.
   Example:
   - **Input**: $4, 8, 5, 7, 2, 9, 10, 1$
   - **Output**: $4, 5, 7, 9, 10$
3. Implement an algorithm to count the number of inversions in an array using the divide and conquer approach.
   An **inversion** in an array is a pair of indices (i, j) such that $i < j$ and arr$[i] >$ arr$[j]$.
   Example:

- **Input**: arr $= [2, 3, 8, 6, 1]$
- **Output**: 5
- **Explanation**: The array $[2, 3, 8, 6, 1]$ has the following inversions:
  - (2, 1), (3, 1), (8, 6), (8, 1), (6, 1)

  So, the total number of inversions is 5.

4. Implement an algorithm to find the k-th smallest element in an unsorted array with a time complexity of $O(n)$. Your goal is to identify the element that would occupy the k-th position if the array were sorted, without fully sorting the array. Use the divide and conquer approach to solve the problem.

   Example:

   - **Input**: arr $= [7, 10, 4, 3, 20, 15]$, $k = 4$
   - **Output**: 10

5. Given an $m \times n$ matrix where each row and each column is sorted in ascending order. Find the k-th smallest element in this matrix with running time strictly less than $O(n^2)$.

   Matrix Properties:

   - The matrix is sorted in non-decreasing order both row-wise and column-wise.
   - The size of the matrix is $m \times n$, where m and n are positive integers.
   - k is a positive integer where $1 \leq k \leq m \times n$.

   Note: Solve this problem using **binary search** approach
   Example:

   - **Input:**

     ```
     Matrix:
     1  2  2
     3  4  5
     4  5  7

     k = 3
     ```

   - **Output:**

     ```
     2
     ```

6. Given an integer array of length n. For each element in the array, determine how many elements to its right are smaller than that element. You need to solve this problem using divide and conquer approach with time complexity of $O(n \log n)$.
   Example:

- **Input**: arr $= [8, 2, 6, 3, 5]$
- **Output**: $[4, 0, 2, 0, 0]$
- **Explanation**: For 8: there are 4 numbers smaller than 8 to its right $(2, 6, 3, 5)$.
  For 2: there are 0 numbers smaller than 2 to its right.
  For 6: there are 2 numbers smaller than 8 to its right $(3, 5)$.

7. An array $A[1...n]$ 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.

   i) Show how to solve this problem in $O(nlogn)$ time. (Hint: Split the array A into two arrays $A1$ and $A2$ of half the size. Does knowing the majority elements of $A1$ and $A2$ help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

   ii) 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.)

8. Implement Strassen's algorithm for multiplying two square matrices. Test your implementation by comparing the results with those from the standard matrix multiplication method. Measure and compare the execution times of both methods for matrices of different sizes.

Figure 1: L-shaped triomino

9. Given a chessboard with dimensions $n \times n$, where $n = 2^k$ and $k > 0$. On this chessboard, exactly one square is missing (See Figure 2). Develop an algorithm to cover the entire chessboard (excluding the missing square) using L-shaped triominoes.

A *L-shaped triomino* is a 2×2 block with one cell of size 1×1 missing (See Figure 1).

Constraints:

i. Two triominoes should not overlap.

ii. Triminoes should not cover defective square.

iii. Triominoes should cover all other squares.

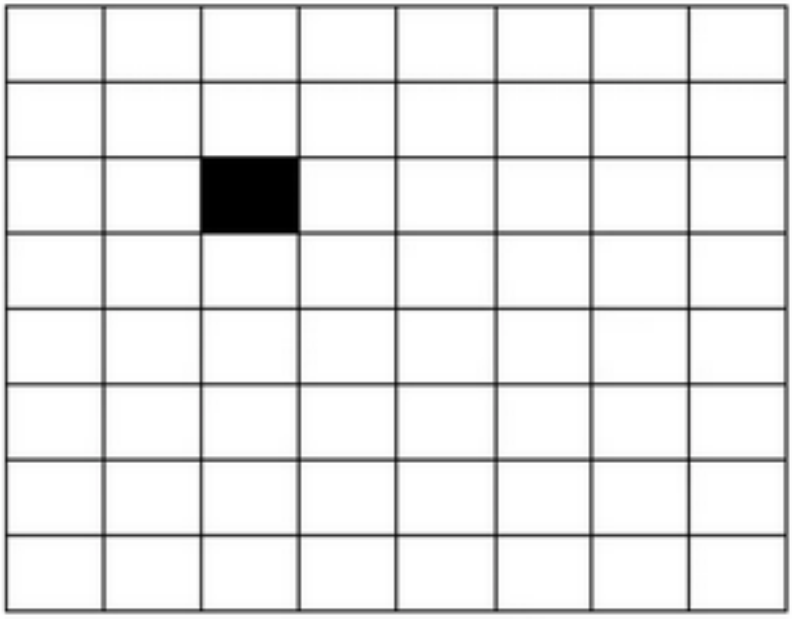

Figure 2: A 8×8 chessboard with one square missing