Instructions: This exam is open book. You can use any reference books in the exam. But no computers are allowed. Partial credits will be given in the marking. If you do not know the answer, give your best guess. You have 3 hours to complete the exam.

1 Find an integer in a matrix [10 points]

There is an integer matrix whose elements are in increasing order in each row and each column, i.e., \( A(i, j) \geq A(i - 1, j) \) and \( A(i, j) \geq A(i, j - 1) \). Design an efficient algorithm to check whether an integer is in the matrix.

2 A natural number not in an array [10 points]

Design an efficient algorithm to find a natural number not in a natural number array. To get full credit, your algorithm should have a \( O(\log(n)) \) complexity, where \( n \) is the length of the input array. [Hint: consider devising a divide and conquer algorithm].

3 2Sum [10 points]

Design a \( O(n) \) algorithm to find 2 numbers from an sorted integer array of length \( n \) so that their sum equals 0.

4 Find largest 1-block [10 points]

You are given an \( m \times n \) matrix \( M \) in which each entry is a 0 or 1. A solid block is a rectangular subset of \( M \) in which each entry is 1. Give a correct efficient algorithm to find a solid block in \( M \) with maximum area.

5 Find bottleneck edges [10 points]

In a directed graph, given a start node and an end node, design an algorithm to find the minimum number of edges that need to be removed so that the start node and end node are disconnected.

6 Find string in 2D pattern [10 points]

Given a 2D array that contains letters, find a string in the array. If the string is not in the pattern, report it. The successive letters in the string should be neighbors in the array. Here we define the neighbors as the letters having row difference of one or column difference of one. For example, CSBC is in the following 2D array:
7 Walking upstairs [10 points]

Design a polynomial complexity algorithm to compute the number of different ways to climb up n-step stairs if we can walk either one step or two steps each time. Analyze the complexity of your algorithm.

8 Sum of path weights [10 points]

There is a directed acyclic graph whose node set is V and edge set is E. The weight on each edge of the graph is non-negative. We define the weight of each directed path as the product of the weights of edges on the path. Design a \( O(|V| + |E|) \) algorithm to compute the sum of all the path weights from node \( v \) to node \( u \).

9 Building bridges [10 points]

Consider a 2-D map with a horizontal river passing through its center. There are \( n \) cities on the southern bank with x-coordinates \( a(1) \ldots a(n) \) and \( n \) cities on the northern bank with x-coordinates \( b(1) \ldots b(n) \). You want to connect as many north-south pairs of cities as possible with bridges such that no two bridges cross. When connecting cities, you can only connect city \( i \) on the northern bank to city \( i \) on the southern bank. [Hint: consider dynamic programming].

10 Largest subtree [10 points]

A subtree of a (rooted, ordered) binary tree \( T \) consists of a node and all its descendants. Design and analyze an efficient algorithm to compute the largest common subtree of two given binary trees \( T_1 \) and \( T_2 \), that is, the largest subtree of \( T_1 \) that is isomorphic to a subtree in \( T_2 \). The contents of the nodes are irrelevant; we are only interested in matching the underlying combinatorial structure.