1. Given an array of \( n \) elements, where \( n \) is a power of 2. We want an efficient algorithm to find the maximum and minimum elements of the array. (We want an efficient algorithm that minimizes the worst-case number of comparisons. An algorithm that uses \( 2(n - 1) \) comparisons is not acceptable.)

a) Using a divide-and-conquer technique, write a recursive function to find the largest and smallest elements of an array.

b) Let \( T(n) \) be the worst-case number of key-comparisons used. Write a recurrence for \( T(n) \). Solve the recurrence to find an exact expression for \( T(n) \).
2. Let \( S_1, S_2, \ldots, S_k \) be sets of integers, where \( n \) is the total number of elements in all sets combined, and all integers have values in the range 1 to \( n \). Let \( n_i \) be the number of elements in set \( S_i \). The number of elements in each set may vary from only a few to very many. Thus, the number of sets, \( k \), is not constant and may be \( O(n) \). We want to sort the elements in each set, so that at the end, each set \( S_i \) will have its own original elements but in sorted order.)

a) Suppose we apply bucket-sort to each set separately. What is the total time complexity of this algorithm (in terms of \( k \) and \( n \))? Explain.

b) Outline a more efficient algorithm to perform the entire job. (Describe the algorithm in terms of major steps.) Analyze the time complexity.

c) Illustrate the improved algorithm on the following example, where \( k = 3 \) and \( N = 8 \).

\[
S_1 = (8, 5, 7), S_2 = (6, 5, 8), S_3 = (5, 3).
\]
3. Given a sorted sequence of \( n \) integers, \( A[0..n-1] \), and an integer \( M \). We want an efficient \( O(n) \)-time algorithm to determine if there exists a pair of elements \( A[i], A[j] \) such that \( A[i] + A[j] = M \). (The algorithm should return 1 if such a pair exists, otherwise return 0.)

a) First outline the algorithm in simple words and illustrate on the following example:

\[
A = (2, 10, 12, 20, 25, 30, 60, 85, 90, 100, 120, 150, 200), \quad M = 90.
\]

b) Write the code for a function to implement this algorithm.

\[
\text{int PAIR(int A[], int n, int M)}
\]

\{

\}

c) Analyze the time complexity of this algorithm.
4. This problem deals with PRIM’s minimum-cost-spanning-tree (MST) algorithm for a weighted, connected, undirected graph with $n$ vertices and $e$ edges. Assume the graph is sparse and is represented by its adjacency lists.

(a) Briefly describe the implementation of the algorithm. (What data structures are used and how each iteration is implemented.)

(b) Briefly analyze the worst-case time-complexity of the algorithm.

(c) Illustrate how the algorithm works on the following example graph. Start with vertex 1 as the seed and show the result after each iteration.

```
   3
  1  ------  2
     \       / \    \
    10 \     / 6 \ 5
        \   /    \
           \ /    \
             3  ------  4
                   2
```