1. Given a sorted array of \( n \) elements that consists of only zeros and ones. That is, the array consists of some number of zeros, say \( k \), followed by \( n - k \) ones. We want to determine the number of zeros, using an asymptotically efficient algorithm. (A sequential algorithm with \( O(n) \) time complexity is not efficient and would not be acceptable.)

(a) Write the code for a recursive function that performs this task and returns the number of zeros in the array. Provide some comments and explanation about how the algorithm works.

```c
int Count (\( A[\ ] \), left, right )
```

(b) Analyze the worst-case time complexity of your algorithm.
2. The problem of k-way merge is to merge k sorted lists (of real-valued elements). Each list has \( n/k \) sorted elements, and we want to merge them into a single sorted list of \( n \) elements.

(a) A naive approach is to merge the first and second list into a sorted list of \( 2n/k \) elements, then merge the result with the third list, then with the fourth list, and so on until the whole job is done. Analyze the worst-case running time of this algorithm in terms of \( n \) and \( k \).

(b) Describe an efficient algorithm for this k-way merging problem. (Assume \( k \) is a power of 2.) Analyze the worst-case time complexity in terms of \( n \) and \( k \).
3. Given an array of $n$ elements with arbitrary random values, and an integer $k$ in the range $[1, n]$. We want to obtain the sequence of $k$ smallest elements in sorted order. (Note that we want the entire sequence of $k$ smallest elements.)

(a) You are given the linear-time SELECT algorithm, which uses the median-of-medians for pivot. This function returns the $k$th smallest element of the array. (Do NOT describe this function which is already given.)

```
SELECT (dtype A[], int n, int k)
```

Briefly describe how to USE this function (and possibly other known functions) to find the sorted sequence of $k$ smallest elements. Your algorithm must be asymptotically efficient. Analyze the worst-case time complexity (in terms of $n$ and $k$).

(b) Write the code for a tailored version of bubble-sort to perform the job of finding the smallest $k$ elements in sorted order. Analyze the worst-case time complexity (in terms of $n$ and $k$).

(c) Can Bucket-Sort or Radix-sort be used for this problem? Explain.
4. (a) Insert the following sequence of elements into an AVL tree, starting with an empty tree. Show the result after each balancing operation.

\[50, 10, 20, 70, 30, 25\]

(b) Delete element 5 in the following AVL tree.

```
                  50
               /     |
            20      80
           /  \
          10   70
         /  \
        5    60
       /  \
      45  69
     /  \
    52  55
```

5. Floyd's all-pairs-shortest-paths algorithm produces two matrices $A$ (cost) and $P$ (path), such that at the end, $A[i, j]$ is the cost of the shortest $(i, j)$ path, and $P[i, j]$ is the first vertex immediately after vertex $i$ on the shortest-path.

(a) Show the working of Floyd's algorithm for the following weighted graph. Show the matrix $A$ and $P$ after each iteration. (You may show the entries $A[i, j]/P[i, j]$ as a pair, rather than two physically separate matrices.)

![Weighted Graph Diagram]
(b) What is the time complexity of Floyd’s algorithm for each of the following cases.

i. Graph represented by its adjacency matrix.

ii. Graph represented by its adjacency lists.