Design and Analysis of Algorithms

# Week – 1 Assignment

1. You are given a sorted array A of size n. Write an iterative program to remove the duplicates from the array. For example, if A[] = {2, 7, 7, 11, 24, 24, 24, 29, 36, 36}, your output should be B[] = {2, 7, 11, 24, 29, 36}.
    a. Count the operations to get the closed form equation of running time (worst case).
    b. Submit the program for the problem https://leetcode.com/problems/remove-duplicates-from-sorted-array/ and submit the snapshot of acceptance as proof.

2. Consider an array A of size n. Split A[] into the two arrays Low[] and High[] such that Low[] contains all elements < A[0] and High[] contains all elements >= A[0].
    (a) Write an iterative algorithm to do this operation.
    (b) Count the operations to get the closed form of equation of running time T(n).

3. The $n^{th}$ number in the Fibonacci series Fib(n) is given by the recursive formula: Fib(n-1) + Fib(n-2). The first two numbers of the series are given by Fib(0) = Fib(1) = 1.
    a. Write a recursive program for computing Fib(n).
    b. Count the number of operations and determine the running time T(n).
    c. Solve the recurrence equation by approximating the equation to a simpler form.

4. Given two sorted lists A[1..n] and B[1..n], write an algorithm to merge them into a single sorted list C[1..2n]. For example, if A[] = {1,3,6,7} and B[] = {2,4,5,8}, then C[] = {1,2,3,4,5,6,7,8}.
    a. Write either an iterative or a recursive program (based on your comfort).
    b. Count the operations to get the closed form of equation of running time T(n).
    c. Submit the program for the problem https://leetcode.com/problems/merge-two-sorted-lists/ and submit the snapshot of acceptance as proof.

5. Given below are the two sorting algorithms. (a) Try the example in hand first and see how they work in each case. (b) Count the number of operations. (c) Write down the recurrence equations. (d) Solve them to derive the closed-form equations.

| **Algorithm: Sort1(A[1..n])**<br>1. **If** n == 1<br>**2.** **Return**<br>**3.** **Else**<br>4. pos = findMaxPos(A[1..n])<br>5. Swap(A[pos], A[n])<br>6. **Sort1**[A[1..n-1])<br><br>// Function call example<br>A[1..8] = {5,8,3,6,1,4,2,7}<br>Sort1(A[1..8]) | **Algorithm: Sort2(A[1..n])**<br>1. **If** n == 1<br>2. **Return** A[n]<br>**3.** **Else**<br>4. B[1..n/2] = **Sort2**[A[1..n/2])<br>5. C[1..n/2] = **Sort2**[A[n/2+1..n])<br>6. A[1..n] = merge(B[1..n/2], C[1..n/2])<br><br>// Function call example<br>A[1..8] = {5,8,3,6,1,4,2,7}<br>Sort1(A[1..8]) |
|---|---|
| findMaxPos() returns the position at which max element is present. Assume it take 'n' operations**.** | merge() combines two sorted arrays into a single sorted array. Assume it take n operations. (The algorithm has minor errors but are immaterial for the question.) |

Penalty for copy or any other form of malpractice: You will be awarded 0 marks