PROBLEMS OF THE DAY – 6

1.Merge Without Extra Space

Given two sorted arrays arr1[] and arr2[] in non-decreasing order. Merge them in sorted order without using any extra space. Modify arr1 so that it contains the first n elements and modify arr2 so that it contains the last m elements.

Examples:

Input: n = 4, m = 5, arr1[] = [1 3 5 7], arr2[] = [0 2 6 8 9]

Output: arr1[] = [0 1 2 3], arr2[] = [5 6 7 8 9]

Explanation: After merging the two non-decreasing arrays, we get, 0 1 2 3 5 6 7 8 9

Input: n = 2, m = 3, arr1[] = [10 12], arr2[] = [5 18 20]

Output: arr1[] = [5 10], arr2[] = [12 18 20]

Explanation: After merging two sorted arrays we get 5 10 12 18 20.

Expected Time Complexity: **O((n+m) log(n+m))**

Expected Auxiliary Space: **O(1)**

2.Minimum sum partition

Given an array arr of size n containing non-negative integers, the task is to divide it into two sets S1 and S2 such that the absolute difference between their sums is minimum and find the minimum difference.

Examples:

Input: N = 4, arr[] = {1, 6, 11, 5}

Output: 1

Explanation: Subset1 = {1, 5, 6}, sum of Subset1 = 12 . Subset2 = {11}, sum of Subset2 = 11.

Input: N = 2, arr[] = {1, 4}

Output: 3

Explanation: Subset1 = {1}, sum of Subset1 = 1. Subset2 = {4}, sum of Subset2 = 4.

Expected Time Complexity: **O(N*|sum of array elements|)**

Expected Auxiliary Space: **O(N*|sum of array elements|)**

3.Longest Prefix Suffix

Given a string of characters, find the length of the longest proper prefix which is also a proper suffix.

NOTE: Prefix and suffix can be overlapping but they should not be equal to the entire string.

Examples:

Input: str = "abab"

Output: 2

Explanation: "ab" is the longest proper prefix and suffix.

Input: str = "aaaa"

Output: 3

Explanation: "aaa" is the longest proper prefix and suffix.

Expected Time Complexity: **O(|str|)**

Expected Auxiliary Space: **O(|str|)**

4.nCr mod M

Given 2 integers n and r. You task is to calculate nCr%1000003.

Examples:

Input: n = 5, r = 2

Output: 10

Explanation: 5C2 = 5! / (2! * 3!) = 10

Input: n = 3, r = 2

Output: 3

Explanation: 3C2 = 3! / (2! * 1!) = 3

Expected Time Complexity: **O(m * logmn) where m = 1000003**

Expected Space Complexity: **O(m)**

5.Maximum Product Subarray

Given an array arr[] that contains positive and negative integers (may contain 0 as well). Find the maximum product that we can get in a subarray of arr.

Note: It is guaranteed that the output fits in a 64-bit integer.

Examples

Input: arr[] = [6, -3, -10, 0, 2]

Output: 180

Explanation: The subarray [6, -3, -10] gives max product as 180.

Input: arr[] = [2, 3, 4, 5, -1, 0]

Output: 120

Explanation: The subarray [2, 3, 4, 5] gives max product as 120.

Input: arr[] = [2, 3, 4]

Output: 24

Explanation: For an array with all positive elements, the result is product of all elements.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(1)**