**LAVANYAA A AIML**

1. **Union of arrays with duplicate elements**

import java.util.Arrays;

class UniqueUnion {

public static int countUniqueElements(int[] arr1, int[] arr2) {

Arrays.sort(arr1);

Arrays.sort(arr2);

int index1 = 0;

int index2 = 0;

int len1 = arr1.length;

int len2 = arr2.length;

int uniqueCount = 0;

int previous = Integer.MIN\_VALUE;

while (index1 < len1 && index2 < len2) {

if (arr1[index1] < arr2[index2]) {

if (arr1[index1] != previous) {

uniqueCount++;

previous = arr1[index1];

}

index1++;

} else if (arr2[index2] < arr1[index1]) {

if (arr2[index2] != previous) {

uniqueCount++;

previous = arr2[index2];

}

index2++;

} else {

if (arr1[index1] != previous) {

uniqueCount++;

previous = arr1[index1];

}

index1++;

index2++;

}

}

while (index1 < len1) {

if (arr1[index1] != previous) {

uniqueCount++;

previous = arr1[index1];

}

index1++;

}

while (index2 < len2) {

if (arr2[index2] != previous) {

uniqueCount++;

previous = arr2[index2];

}

index2++;

}

return uniqueCount;

}

}

TIME COMPLEXITY: O(n)

SPACE COMPLEXITY: O(1)

1. **Equilibrium Point**

class ArrayUtils {

public static int findEquilibriumIndex(int[] nums) {

int length = nums.length;

int[] prefixSum = new int[length];

int[] suffixSum = new int[length];

int totalSumLeft = 0, totalSumRight = 0;

for (int i = 0; i < length; i++) {

totalSumLeft += nums[i];

totalSumRight += nums[length - i - 1];

prefixSum[i] = totalSumLeft;

suffixSum[length - i - 1] = totalSumRight;

}

for (int i = 0; i < length; i++) {

if (prefixSum[i] == suffixSum[i]) {

return i + 1;

}

}

return -1;

}

}

TIME COMPLEXITY: O(n)

SPACE COMPLEXITY: O(n)