**Q1: Merge two arrays by satisfying given constraints**

**Given two sorted arrays X[] and Y[] of size m and n each where m >= n and X[] has exactly n vacant cells,merge elements of Y[] in their correct position in array X[], i.e., merge (X, Y) by keeping the sorted order.**

**For example,**

**Input: X[] = { 0, 2, 0, 3, 0, 5, 6, 0, 0 }**

**Y[] = { 1, 8, 9, 10, 15 } The vacant cells in X[] is represented by 0**

**Output: X[] = { 1, 2, 3, 5, 6, 8, 9, 10, 15 }**

**Code:**

import java.util.Arrays;

import java.util.Scanner;

public class MergeArrays {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input array X[]

System.out.print("Enter the size of array X: ");

int m = scanner.nextInt();

int[] X = new int[m];

System.out.println("Enter the elements of array X: ");

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

X[i] = scanner.nextInt();

}

// Input array Y[]

System.out.print("Enter the size of array Y: ");

int n = scanner.nextInt();

int[] Y = new int[n];

System.out.println("Enter the elements of array Y:");

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

Y[i] = scanner.nextInt();

}

scanner.close();

mergeArrays(X, Y);

// Sort the merged array

Arrays.sort(X);

System.out.println("Merged and sorted array: " + Arrays.toString(X));

}

public static void mergeArrays(int[] X, int[] Y) {

int m = X.length;

int n = Y.length;

int i = m - 1; // Index of last element in X[]

int j = n - 1; // Index of last element in Y[]

int lastIndex = m - 1;

// Move non-zero elements of X to the end

for (int k = m - 1; k >= 0; k--) {

if (X[k] != 0) {

X[lastIndex--] = X[k];

}

}

// Merge X and Y

i = lastIndex + 1;

int k = 0;

while (i < m && j >= 0) {

if (X[i] < Y[j]) {

X[k++] = X[i++];

} else {

X[k++] = Y[j--];

}

}

// Copy remaining elements of Y[] if any

while (j >= 0) {

X[k++] = Y[j--];

}

}

}

**Q2:Find maximum sum path involving elements of given arrays**

**Given two sorted arrays of integers, find a maximum sum path involving elements of both arrays whose sum is maximum. We can start from either array, but we can switch between arrays only through its common elements.**

**For example,**

**Input: X = { 3, 6, 7, 8, 10, 12, 15, 18, 100 }**

**Y = { 1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50 }**

**The maximum sum path is: 1 —> 2 —> 3 —> 6 —> 7 —> 9 —> 10 —> 12 —> 15 —> 16 —> 18 —> 100**

**The maximum sum is 199**

**Code:**

import java.util.ArrayList;

import java.util.Scanner;

public class MaximumSumPath {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of array X: ");

int m = scanner.nextInt();

int[] X = new int[m];

System.out.println("Enter the elements of array X (sorted):");

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

X[i] = scanner.nextInt();

}

System.out.print("Enter the size of array Y: ");

int n = scanner.nextInt();

int[] Y = new int[n];

System.out.println("Enter the elements of array Y (sorted):");

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

Y[i] = scanner.nextInt();

}

scanner.close();

ArrayList<Integer> maxPath = new ArrayList<>();

int maxSum = findMaximumSumPath(X, Y, maxPath);

System.out.println("The maximum sum path is:");

printMaximumSumPath(maxPath);

System.out.println("The maximum sum is: " + maxSum);

}

public static int findMaximumSumPath(int[] X, int[] Y, ArrayList<Integer> maxPath) {

int m = X.length;

int n = Y.length;

int i = 0, j = 0;

int sumX = 0, sumY = 0, maxSum = 0;

while (i < m && j < n) {

if (X[i] < Y[j]) {

sumX += X[i++];

} else if (X[i] > Y[j]) {

sumY += Y[j++];

} else { // Common element found, choose maximum sum path

maxSum += Math.max(sumX, sumY) + X[i];

maxPath.add(X[i]);

sumX = 0;

sumY = 0;

i++;

j++;

}

}

while (i < m) {

sumX += X[i++];

}

while (j < n) {

sumY += Y[j++];

}

maxSum += Math.max(sumX, sumY);

return maxSum;

}

public static void printMaximumSumPath(ArrayList<Integer> maxPath) {

for (int i = 0; i < maxPath.size() - 1; i++) {

System.out.print(maxPath.get(i) + " -> ");

}

System.out.println(maxPath.get(maxPath.size() - 1) + " -> End");

}

}

**Q3:Write a Java Program to count the number of words in a string using HashMap.**

**Code:**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class WordCountUsingHashMap {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

scanner.close();

Map<String, Integer> wordCountMap = countWords(input);

displayWordCount(wordCountMap);

}

public static Map<String, Integer> countWords(String input) {

Map<String, Integer> wordCountMap = new HashMap<>();

String[] words = input.trim().split("\\s+");

for (String word : words) {

word = word.toLowerCase();

word = word.replaceAll("[^a-zA-Z0-9]", "");

if (!word.isEmpty()) {

// Update word count in the HashMap

wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);

}

}

return wordCountMap;

}

public static void displayWordCount(Map<String, Integer> wordCountMap) {

System.out.println("Word Counts:");

for (Map.Entry<String, Integer> entry : wordCountMap.entrySet()) {

System.out.println(entry.getKey() + " : " + entry.getValue());

}

System.out.println("Total words: " + wordCountMap.size());

}

}

**Q4:Write a Java Program to find the duplicate characters in a string.**

**Code:**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class DuplicateCharacters {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

scanner.close();

Map<Character, Integer> charCountMap = countCharacters(input);

displayDuplicateCharacters(charCountMap);

}

public static Map<Character, Integer> countCharacters(String input) {

Map<Character, Integer> charCountMap = new HashMap<>();

for (char ch : input.toCharArray()) {

// Ignore spaces

if (ch != ' ') {

charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);

}

}

return charCountMap;

}

public static void displayDuplicateCharacters(Map<Character, Integer> charCountMap) {

System.out.println("Duplicate Characters:");

for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {

if (entry.getValue() > 1) {

System.out.println(entry.getKey() + " : " + entry.getValue());

}

}

}

}