public class Main {

public static void main(String[] args) {

int[] X = {0, 2, 0, 3, 0, 5, 6, 0, 0};

int[] Y = {1, 8, 9, 10, 15};

mergeArrays(X, Y);

System.out.println("Merged array: ");

for (int i : X) {

System.out.print(i + " ");

}

}

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

int m = X.length;

int n = Y.length;

int i = m - 1; // Start filling from the end of X

int j = n - 1; // Start iterating from the end of Y

while (j >= 0) {

if (i >= 0 && X[i] > Y[j]) {

X[i] = X[i]; // Keep the larger element from X (already in position)

i--;

} else {

X[i] = Y[j];

j--;

}

i--;

}

}

}

Output:

Merged array:

[0 2 0 3 1 8 9 10 15 ]

2. public class Main {

public static void main(String[] args) {

int[] arr1 = {3, 6, 7, 8, 10, 12, 15, 18, 100};

int[] arr2 = {1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50};

int maxSum = findMaxSumPath(arr1, arr2);

System.out.println("Maximum sum path: " + maxSum);

}

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

int m = arr1.length;

int n = arr2.length;

int i = 0, j = 0, maxSum = 0;

int prev = -1; // To keep track of previously visited common element

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

if (arr1[i] == arr2[j]) {

maxSum += arr1[i];

prev = arr1[i]; // Update common element

i++;

j++;

} else if (arr1[i] < arr2[j]) {

i++;

} else {

j++;

}

}

while (i < m) {

if (arr1[i] != prev) {

maxSum += arr1[i];

prev = arr1[i];

}

i++;

}

while (j < n) {

if (arr2[j] != prev) {

maxSum += arr2[j];

prev = arr2[j];

}

j++;

}

return maxSum;

}

}

Output:

Maximum sum path: 153

3.

public class Main {

public static int maxSumPath(int[] X, int[] Y) {

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

int i = 0, j = 0;

while (i < X.length && j < Y.length) {

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

sumX += X[i++];

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

sumY += Y[j++];

} else { // Common element found

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

sumX = sumY = 0;

i++;

j++;

}

}

// Add remaining elements if any

while (i < X.length) {

sumX += X[i++];

}

while (j < Y.length) {

sumY += Y[j++];

}

// Add the maximum sum of the remaining elements

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

return maxSum;

}

public static void main(String[] args) {

int[] X = { 3, 6, 7, 8, 10, 12, 15, 18, 100 };

int[] Y = { 1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50 };

System.out.println("Maximum sum path: " + maxSumPath(X, Y));

}

}

Maximum sum path: 199

import java.util.HashMap;

public class Main {

public static void main(String[] args) {

String str = "Hello world, how are you? Hello again!";

str = str.replaceAll("[^a-zA-Z ]", "").toLowerCase();

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

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

for (String word : words) {

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

}

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

for (String word : wordCountMap.keySet()) {

System.out.println(word + ": " + wordCountMap.get(word));

}

}

}

Output:

Word count:

how: 1

world: 1

are: 1

again: 1

hello: 2

you: 1