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 }

Answer:-

class MergeArrays {

void moveToEnd(int mPlusN[], int size)

{

int i, j = size - 1;

for (i = size - 1; i >= 0; i--) {

if (mPlusN[i] != -1) {

mPlusN[j] = mPlusN[i];

j--;

}

}

}

void merge(int mPlusN[], int N[], int m, int n)

{

int i = n;

int j = 0;

int k = 0;

while (k < (m + n))

{

if ((i < (m + n) && mPlusN[i] <= N[j])

|| (j == n)) {

mPlusN[k] = mPlusN[i];

k++;

i++;

}

{

mPlusN[k] = N[j];

k++;

j++;

}

}

void printArray(int arr[], int size)

{

int i;

for (i = 0; i < size; i++)

System.out.print(arr[i] + " ");

System.out.println("");

}

public static void main(String[] args)

{

MergeArrays mergearray = new MergeArrays();

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

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

int n = N.length;

int m = mPlusN.length - n;

mergearray.moveToEnd(mPlusN, m + n);

mergearray.merge(mPlusN, N, m, n);

mergearray.printArray(mPlusN, m + n);

}

}

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

Answer :-

class MaximumSumPath {

int max(int x, int y) { return (x > y) ? x : y; }

int maxPathSum(int ar1[], int ar2[], int m, int n)

{

int i = 0, j = 0;

int result = 0, sum1 = 0, sum2 = 0;

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

if (ar1[i] < ar2[j])

sum1 += ar1[i++];

else if (ar1[i] > ar2[j])

sum2 += ar2[j++];

else {

result += max(sum1, sum2) + ar1[i];

sum1 = 0;

sum2 = 0;

i++;

j++;

}

while (i < m)

sum1 += ar1[i++];

while (j < n)

sum2 += ar2[j++];

result += max(sum1, sum2);

return result;

}

public static void main(String[] args)

{

MaximumSumPath sumpath = new MaximumSumPath();

int ar1[] = { 2, 3, 7, 10, 12, 15, 30, 34 };

int ar2[] = { 1, 5, 7, 8, 10, 15, 16, 19 };

int m = ar1.length;

int n = ar2.length;

System.out.println(

"Maximum sum path is :"

+ sumpath.maxPathSum(ar1, ar2, m, n));

}

}

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

Answer :-

import java.io.\*;

import java.util.HashMap;

import java.util.Map;

class GFG {

public static void main(String[] args)

{

String str = "Alice is girl and Bob is boy";

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

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

for (String word : words) {

Integer integer = hashMap.get(word);

if (integer == null)

hashMap.put(word, 1);

else {

hashMap.put(word, integer + 1);

}

}

System.out.println(hashMap);

}

}

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

public class DuplicateCharacters {

public static void main(String[] args) {

String string1 = "Great responsibility";

int count;

char string[] = string1.toCharArray();

System.out.println("Duplicate characters in a given string: ");

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

count = 1;

for(int j = i+1; j <string.length; j++) {

if(string[i] == string[j] && string[i] != ' ') {

count++;

string[j] = '0';

}

}

if(count > 1 && string[i] != '0')

System.out.println(string[i]);

}

}