Merge two sorted arrays - GeeksforGeeks

Source: https://www.geeksforgeeks.org/merge-two-sorted-arrays/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Merge two sorted arrays Last Updated : 31 Jul, 2025 Given two sorted arrays arr1[] of size n and arr2[] of size m. Merge these two arrays. After the merge, the first n smallest elements of the combined sorted array should be stored in arr1[], and the remaining m largest elements should be stored in arr2[]. After the merging process, both arr1[] and arr2[] must remain sorted in non-decreasing order. Examples: Input : arr1[] = [1, 3, 4, 5], arr2[] = [2, 4, 6, 8] Output : arr1[] = [1, 2, 3, 4], arr2[] = [4 5, 6, 8] Explanation: Combined sorted array = [1, 2, 3, 4, 4, 5, 6, 8], array arr1[] contains smallest 4 elements: 1, 2, 3, 4, and array arr2[] contains the remaining 4 elements: 4, 5, 6, 8. Input : arr1[] = [5, 8, 9], arr2[] = [4, 7, 8] Output : arr1[] = [4, 5, 7], arr2[] = [8, 8, 9] Explanation: Combined sorted array = [4, 5, 7, 8, 8, 9], array arr1[] contains smallest 3 elements: 4, 5, 7, and array arr2[] contains the remaining 3 elements: 8, 8, 9. Try it on GfG Practice Table of Content [Naive Approach] Concatenate and Sort - O((n + m) log(n + m)) Time and O(n + m) Space [Expected Approach] Two-Pointer Merge - O(n + m) Time and O(n + m) Space [Naive Approach] Concatenate and Sort - O((n + m) log(n + m)) Time and O(n + m) Space The idea is to combine both sorted arrays into a single temporary array, then sort this combined array to get all elements in non-decreasing order. Once sorted, the first n smallest elements are copied back into arr1, and the remaining m largest elements are placed into arr2. C++ #include <iostream> #include <vector> #include <algorithm> using namespace std ; void mergeArrays (vector < int >& arr1 , vector < int >& arr2) { int n = arr1 . size (); int m = arr2 . size (); // temporary array to store all elements // from arr1 and arr2 vector < int > merged (n + m); // copy elements from arr1 and arr2 into merged array for (int i = 0 ; i < n ; ++ i) { merged [i] = arr1 [i]; } for (int j = 0 ; j < m ; ++ j) { merged [n + j] = arr2 [j]; } // dort the merged array sort (merged . begin (), merged . end ()); // distribute first n elements to arr1 for (int i = 0 ; i < n ; ++ i) { arr1 [i] = merged [i]; } // distribute remaining m elements to arr2 for (int j = 0 ; j < m ; ++ j) { arr2 [j] = merged [n + j]; } } int main () { vector < int > arr1 = { 1 , 3 , 5 , 7 }; vector < int > arr2 = { 2 , 4 , 6 , 8 }; mergeArrays (arr1 , arr2); for (int num : arr1) { cout << num << ' ' ; } cout << endl ; for (int num : arr2) { cout << num << ' ' ; } cout << endl ; return 0 ; } C #include <stdio.h> #include <stdlib.h> // function to compare two integers (used in qsort) int compare (void * a , void * b) { return (* (int *) a - * (int *) b); } // function to merge two sorted arrays in-place void mergeArrays (int arr1 [], int arr2 [], int n , int m) { // temporary array to store all elements // from arr1 and arr2 int * merged = (int *) malloc ((n + m) * sizeof (int)); // copy elements from arr1 and arr2 into merged array for (int i = 0 ; i < n ; ++ i) { merged [i] = arr1 [i]; } for (int j = 0 ; j < m ; ++ j) { merged [n + j] = arr2 [j]; } // sort the merged array qsort (merged , n + m , sizeof (int), compare); // distribute first n elements to arr1 for (int i = 0 ; i < n ; ++ i) { arr1 [i] = merged [i]; } // distribute remaining m elements to arr2 for (int j = 0 ; j < m ; ++ j) { arr2 [j] = merged [n + j]; } // free dynamically allocated memory free (merged); } int main () { int arr1 [] = { 1 , 3 , 5 , 7 }; int arr2 [] = { 2 , 4 , 6 , 8 }; int n = sizeof (arr1) / sizeof (arr1 [0]); int m = sizeof (arr2) / sizeof (arr2 [0]); mergeArrays (arr1 , arr2 , n , m); for (int i = 0 ; i < n ; ++ i) { printf ("%d " , arr1 [i]); } printf (" \n "); for (int i = 0 ; i < m ; ++ i) { printf ("%d " , arr2 [i]); } printf (" \n "); return 0 ; } Java import java.util.Arrays ; class GfG { static void mergeArrays (int [] arr1 , int [] arr2) { int n = arr1 . length ; int m = arr2 . length ; // temporary array to store all elements // from arr1 and arr2 int [] merged = new int [n + m] ; // copy elements from arr1 and arr2 into merged array for (int i = 0 ; i < n ; ++ i) { merged [i] = arr1 [i] ; } for (int j = 0 ; j < m ; ++ j) { merged [n + j] = arr2 [j] ; } // sort the merged array Arrays . sort (merged); // distribute first n elements to arr1 for (int i = 0 ; i < n ; ++ i) { arr1 [i] = merged [i] ; } // distribute remaining m elements to arr2 for (int j = 0 ; j < m ; ++ j) { arr2 [j] = merged [n + j] ; } } public

static void main (String [] args) { int [] arr1 = { 1 , 3 , 5 , 7 }; int [] arr2 = { 2 , 4 , 6 , 8 }; mergeArrays (arr1 , arr2); for (int num : arr1) { System . out . print (num + " "); } System . out . println (); for (int num : arr2) { System . out . print (num + " "); } System . out . println (); } } Python def mergeArrays (arr1 , arr2): n = len (arr1) m = len (arr2) # temporary array to store all elements # from arr1 and arr2 merged = [0] * (n + m) # copy elements from arr1 and arr2 # into merged array for i in range (n): merged [i] = arr1 [i] for j in range (m): merged [n + j] = arr2 [j] # sort the merged array merged . sort () # distribute first n elements to arr1 for i in range (n): arr1 [i] = merged [i] # distribute remaining m elements to arr2 for j in range (m): arr2 [j] = merged [n + j] if __name__ == "__main__" : arr1 = [1 , 3 , 5 , 7] arr2 = [2 , 4 , 6 , 8] mergeArrays (arr1 , arr2) print (* arr1) print (* arr2) C# using System ; class GfG { static void mergeArrays (int [] arr1 , int [] arr2) { int n = arr1 . Length ; int m = arr2 . Length ; // temporary array to store all elements // from arr1 and arr2 int [] merged = new int [n + m]; // copy elements from arr1 and arr2 // into merged array for (int i = 0 ; i < n ; ++ i) { merged [i] = arr1 [i]; } for (int j = 0 ; j < m ; ++ j) { merged [n + j] = arr2 [j]; } // sort the merged array Array . Sort (merged); // distribute first n elements to arr1 for (int i = 0 ; i < n ; ++ i) { arr1 [i] = merged [i]; } // distribute remaining m elements to arr2 for (int j = 0 ; j < m ; ++ j) { arr2 [j] = merged [n + j]; } } static void Main () { int [] arr1 = { 1 , 3 , 5 , 7 }; int [] arr2 = { 2 , 4 , 6 , 8 }; mergeArrays (arr1 , arr2); foreach (int num in arr1) { Console . Write (num + " "); } Console . WriteLine (); foreach (int num in arr2) { Console . Write (num + " "); } Console . WriteLine (); } } JavaScript function mergeArrays (arr1 , arr2) { let n = arr1 . length ; let m = arr2 . length ; // temporary array to store all elements // from arr1 and arr2 let merged = new Array (n + m); // copy elements from arr1 and arr2 // into merged array for (let i = 0 ; i < n ; ++ i) { merged [i] = arr1 [i]; } for (let j = 0 ; j < m ; ++ j) { merged [n + j] = arr2 [j]; } // sort the merged array merged . sort ((a , b) => a - b); // distribute first n elements to arr1 for (let i = 0 ; i < n ; ++ i) { arr1 [i] = merged [i]; } // distribute remaining m elements to arr2 for (let j = 0 ; j < m ; ++ j) { arr2 [j] = merged [n + j]; } } // Driver Code let arr1 = [1 , 3 , 5 , 7]; let arr2 = [2 , 4 , 6 , 8]; mergeArrays (arr1 , arr2); console . log (arr1 . join (' ')); console . log (arr2 . join (' ')); Output 1 2 3 4 5 6 7 8 [Expected Approach] Two Pointer Merge - O(n + m) Time and O(n + m) Space The idea is to use the two-pointer to merge both sorted arrays into a temporary array in linear time. We compare elements from arr1 and arr2 one by one and append the smaller element to the merged array. After merging, we copy the first n elements back to arr1 and the remaining m elements to arr2. Step-by-Step Implementation: Initialize two pointers: i = 0 for traversing arr1, j = 0 for traversing arr2 Create an empty array merged = [] Merge the elements into merged: While both i < n and j < m: => If arr1[i] <= arr2[j], append arr1[i] to merged and increment i => Else, append arr2[j] to merged and increment j Append remaining elements (if any): => While i < n, append arr1[i] to merged and increment i => While j < m, append arr2[j] to merged and increment j Distribute the merged elements back: => Copy the first n elements of merged to arr1 => Copy the remaining m elements to arr2 C++ #include <iostream> #include <vector> using namespace std ; void mergeArrays (vector < int >& arr1 , vector < int >& arr2) { int n = arr1 . size (); int m = arr2 . size (); int i = 0 , j = 0 ; // temporary array to store merged result vector < int > merged ; // merge elements in sorted order while (i < n && j < m) { if (arr1 [i] <= arr2 [j]) { merged . push_back (arr1 [i ++]); } else { merged . push_back (arr2 [j ++]); } } // copy remaining elements from arr1 while (i < n) merged . push_back (arr1 [i ++]); // copy remaining elements from arr2 while (j < m) merged . push_back (arr2 [j ++]); // copy first n to arr1 for (int k = 0 ; k < n ; ++ k) { arr1 [k] = merged [k]; } // copy remaining m to arr2 for (int k = 0 ; k < m ; ++ k) { arr2 [k] = merged [n + k]; } } int main () { vector < int > arr1 = { 1 , 3 , 5 , 7 }; vector < int > arr2 = { 2 , 4 , 6 , 8 }; mergeArrays (arr1 , arr2); for (int num : arr1) cout << num << ' ' ; cout << endl ; for (int num : arr2) cout << num << ' ' ; cout << endl ; return 0 ; } C #include <stdio.h> #include <stdlib.h> void mergeArrays (int arr1 [], int arr2 [], int n , int m) { int * merged = (int *) malloc ((n + m) * sizeof (int)); int i = 0 , j = 0 , k = 0 ; // merge elements in sorted order while (i < n && j < m) { if (arr1 [i] <= arr2 [j]) { merged [k ++] = arr1 [i ++]; } else { merged [k ++] = arr2 [j ++]; } } // copy remaining elements from arr1 while (i < n) merged [k ++] = arr1 [i ++]; // copy remaining elements from arr2 while (j < m) merged [k ++] = arr2 [j ++]; // copy first n to arr1 for (i = 0 ; i < n ; ++ i) arr1 [i] = merged [i]; // copy remaining m to arr2 for (j = 0 ; j < m ; ++ j) arr2 [j] = merged [n + j]; free (merged); } int main () { int arr1 [] = { 1 , 3 , 5 , 7 }; int arr2 [] = { 2 , 4 , 6 , 8 }; int n = sizeof (arr1) / sizeof (arr1 [0]); int m = sizeof (arr2) / sizeof (arr2 [0]); mergeArrays (arr1 , arr2 , n , m); for (int i = 0 ; i < n ; ++ i) printf ("%d " , arr1 [i]); printf (" \n "); for (int i = 0 ; i < m ; ++ i) printf ("%d " , arr2 [i]); printf (" \n "); return 0 ; } Java class GfG { static void mergeArrays (int [] arr1 , int [] arr2) { int n = arr1 . length , m = arr2 . length ; int i = 0 , j = 0 ; int [] merged = new int [n + m] ; // merge elements in sorted order int k = 0 ; while (i < n && j < m) { if (arr1 [i] <= arr2 [j]) { merged [k ++] = arr1 [i ++] ; } else { merged [k ++] = arr2 [j ++] ; } } // copy

remaining elements from arr1 while (i < n) merged [k ++] = arr1 [i ++] ; // copy remaining elements from arr2 while (j < m) merged [k ++] = arr2 [j ++] ; // copy first n to arr1 for (i = 0 ; i < n ; ++ i) arr1 [i] = merged [i] ; // copy remaining m to arr2 for (j = 0 ; j < m ; ++ j) arr2 [j] = merged [n + j] ; } public static void main (String [] args) { int [] arr1 = { 1 , 3 , 5 , 7 }; int [] arr2 = { 2 , 4 , 6 , 8 }; mergeArrays (arr1 , arr2); for (int num : arr1) System . out . print (num + " "); System . out . println (); for (int num : arr2) System . out . print (num + " "); System . out . println (); } } Python def mergeArrays (arr1 , arr2): n = len (arr1) m = len (arr2) i = j = 0 # temporary array to store merged result merged = [] # merge elements in sorted order while i < n and j < m : if arr1 [i] <= arr2 [j]: merged . append (arr1 [i]) i += 1 else : merged . append (arr2 [j]) j += 1 # copy remaining elements from arr1 while i < n : merged . append (arr1 [i]) i += 1 # copy remaining elements from arr2 while j < m : merged . append (arr2 [j]) j += 1 # copy first n to arr1 for i in range (n): arr1 [i] = merged [i] # copy remaining m to arr2 for j in range (m): arr2 [j] = merged [n + j] if __name__ == "__main__" : arr1 = [1 , 3 , 5 , 7] arr2 = [2 , 4 , 6 , 8] mergeArrays (arr1 , arr2) print (* arr1) print (* arr2) C# using System ; class GfG { static void mergeArrays (int [] arr1 , int [] arr2) { int n = arr1 . Length , m = arr2 . Length ; int i = 0 , j = 0 ; int [] merged = new int [n + m]; int k = 0 ; // merge elements in sorted order while (i < n && j < m) { if (arr1 [i] <= arr2 [j]) { merged [k ++] = arr1 [i ++]; } else { merged [k ++] = arr2 [j ++]; } } // copy remaining elements from arr1 while (i < n) merged [k ++] = arr1 [i ++]; // copy remaining elements from arr2 while (j < m) merged [k ++] = arr2 [j ++]; // copy first n to arr1 for (i = 0 ; i < n ; ++ i) arr1 [i] = merged [i]; // copy remaining m to arr2 for (j = 0 ; j < m ; ++ j) arr2 [j] = merged [n + j]; } static void Main () { int [] arr1 = { 1 , 3 , 5 , 7 }; int [] arr2 = { 2 , 4 , 6 , 8 }; mergeArrays (arr1 , arr2); foreach (int num in arr1) Console . Write (num + " "); Console . WriteLine (); foreach (int num in arr2) Console . Write (num + " "); Console . WriteLine (); } } JavaScript function mergeArrays (arr1 , arr2) { let n = arr1 . length , m = arr2 . length ; let i = 0 , j = 0 ; let merged = []; // merge elements in sorted order while (i < n && j < m) { if (arr1 [i] <= arr2 [j]) { merged . push (arr1 [i ++]); } else { merged . push (arr2 [j ++]); } } // copy remaining elements from arr1 while (i < n) merged . push (arr1 [i ++]); // copy remaining elements from arr2 while (j < m) merged . push (arr2 [j ++]); // copy first n to arr1 for (let k = 0 ; k < n ; ++ k) arr1 [k] = merged [k]; // copy remaining m to arr2 for (let k = 0 ; k < m ; ++ k) arr2 [k] = merged [n + k]; } // Driver Code let arr1 = [1 , 3 , 5 , 7]; let arr2 = [2 , 4 , 6 , 8]; mergeArrays (arr1 , arr2); console . log (arr1 . join (' ')); console . log (arr2 . join (' ')); Output 1 2 3 4 5 6 7 8 Problems related to Merging Two Sorted Arrays Merge two sorted arrays with O(1) extra space Merge k sorted arrays Union of Two Sorted Arrays Intersection of Two Sorted Arrays Merge 2 sorted linked list in reverse order Sort last M elements Merge 2 sorted linked list in reverse order Merge Sort for Doubly Linked List Comment Article Tags: Article Tags: Sorting DSA array-merge