Efficiently merging two sorted arrays with O(1) extra space - GeeksforGeeks

Source: https://www.geeksforgeeks.org/efficiently-merging-two-sorted-arrays-with-o1-extra-space/

Efficiently merging two sorted arrays with O(1) extra space Last Updated : 16 May, 2024 Given two sorted arrays, we need to merge them in O((n+m)*log(n+m)) time with O(1) extra space into a sorted array, when n is the size of the first array, and m is the size of the second array. Example: Input: ar1[] = {10}; ar2[] = {2, 3}; Output: ar1[] = {2} ar2[] = {3, 10} Input: ar1[] = {1, 5, 9, 10, 15, 20}; ar2[] = {2, 3, 8, 13}; Output: ar1[] = {1, 2, 3, 5, 8, 9} ar2[] = {10, 13, 15, 20} Recommended: Please solve it on " PRACTICE " first, before moving on to the solution. We have discussed a quadratic time solution in the below post. In this post, a better solution is discussed. Approach: The idea is that we start comparing elements that are far from each other rather than adjacent. For every pass, we calculate the gap and compare the elements towards the right of the gap. Every pass, the gap reduces to the ceiling value of dividing by 2. Examples: First example: a1[] = {3 27 38 43}, a2[] = {9 10 82} Start with gap = ceiling of n/2 = 4 [This gap is for whole merged array] 3 27 38 43 9 10 82 3 27 38 43 9 10 82 3 10 38 43 9 27 82 gap = 2: 3 10 38 43 9 27 82 3 10 38 43 9 27 82 3 10 38 43 9 27 82 3 10 9 43 38 27 82 3 10 9 27 38 43 82 gap = 1: 3 10 9 27 38 43 82 3 10 9 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 Output : 3 9 10 27 38 43 82 Second Example: a1[] = {10 27 38 43 82}, a2[] = {3 9} Start with gap = ceiling of n/2 (4): 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 3 38 43 82 27 9 10 3 9 43 82 27 38 gap = 2: 10 3 9 43 82 27 38 9 3 10 43 82 27 38 9 3 10 43 82 27 38 9 3 10 43 82 27 38 9 3 10 27 82 43 38 9 3 10 27 38 43 82 gap = 1 9 3 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 3 9 10 27 38 43 82 Output : 3 9 10 27 38 43 82 Algorithm: 1. Define a function to find the next gap. This function takes the current gap as input, and returns the next gap to be used in the merge step. 2. Define a merge function that takes two sorted arrays arr1 and arr2, and their respective sizes n and m as inputs. This function will merge the two arrays into arr1. 3. Initialize a variable gap to n + m. 4. Repeat the following steps while gap is greater than 0: a. Calculate the next gap using the nextGap function. b. Compare and swap adjacent elements in the first array arr1, up to the index (n - 1), with a gap of the calculated next gap. c. Compare and swap elements in both arrays arr1 and arr2, starting at index i and j respectively, with a gap of the calculated next gap, until i is less than n and j is less than m. d. If j is less than m, compare and swap adjacent elements in the second array arr2, up to the index (m - 1), with a gap of the calculated next gap. 5. Output the sorted array arr1 and arr2. Below is the implementation of the above idea: C++ #include <iostream> #include <vector> using namespace std ; // Function to find next gap int nextGap (int gap) { if (gap <= 1) return 0 ; return (gap / 2) + (gap % 2); } void merge (vector < int >& arr1 , vector < int >& arr2 , int n , int m) { int gap = n + m ; gap = nextGap (gap); while (gap > 0) { int i = 0 ; // comparing elements in the first array while (i + gap < n) { if (arr1 [i] > arr1 [i + gap]) { swap (arr1 [i], arr1 [i + gap]); } i ++ ; } // comparing elements in both arrays int j = gap > n ? gap - n : 0 ; while (i < n && j < m) { if (arr1 [i] > arr2 [j]) { swap (arr1 [i], arr2 [j]); } i ++ ; j ++ ; } if (j < m) { // comparing elements in the second array j = 0 ; while (j + gap < m) { if (arr2 [j] > arr2 [j + gap]) { swap (arr2 [j], arr2 [j + gap]); } j ++ ; } } gap = nextGap (gap); } } // Driver code int main () { vector < int > a1 = { 10 , 27 , 38 , 43 , 82 }; vector < int > a2 = { 3 , 9 }; int n = a1 . size (); int m = a2 . size (); // Function Call merge (a1 , a2 , n , m); cout << "First Array: " ; for (int i = 0 ; i < n ; i ++) cout << a1 [i] << " " ; cout << endl ; cout << "Second Array: " ; for (int i = 0 ; i < m ; i ++) cout << a2 [i] << " " ; cout << endl ; return 0 ; } Java import java.util.Arrays ;

```
public class MergeSortedArrays { // Function to find next gap public static int nextGap ( int gap ) { if (
gap <= 1 ) return 0 ; return ( gap / 2 ) + ( gap % 2 ); } public static void merge ( int [] arr1 , int [] arr2 , int
n , int m ) { int gap = n + m ; gap = nextGap ( gap ); while ( gap > 0 ) { int i = 0 ; // comparing elements in
the first array while ( i + gap < n ) { if ( arr1 [ i ] > arr1 [ i + gap ] ) { int temp = arr1 [ i ] ; arr1 [ i ] = arr1 [ i
+ gap ] ; arr1 [ i + gap ] = temp ; } i ++ ; } // comparing elements in both arrays int j = gap > n ? gap - n :
0 ; while ( i < n && j < m ) { if ( arr1 [ i ] > arr2 [ j ] ) { int temp = arr1 [ i ] ; arr1 [ i ] = arr2 [ j ] ; arr2 [ j ] =
temp ; } i ++ ; j ++ ; } if ( j < m ) { // comparing elements in the second array j = 0 ; while ( j + gap < m ) {
if ( arr2 [ j ] > arr2 [ j + gap ] ) { int temp = arr2 [ j ] ; arr2 [ j ] = arr2 [ j + gap ] ; arr2 [ j + gap ] = temp ; } j
++ ; } } gap = nextGap ( gap ); } } // Driver code public static void main ( String [] args ) { int [] a1 = { 10 ,
27 , 38 , 43 , 82 }; int [] a2 = { 3 , 9 }; int n = a1 . length ; int m = a2 . length ; // Function Call merge ( a1 ,
a2 , n , m ); System . out . print ( "First Array: " ); for ( int i = 0 ; i < n ; i ++ ) System . out . print ( a1 [ i ] +
" " ); System . out . println (); System . out . print ( "Second Array: " ); for ( int i = 0 ; i < m ; i ++ ) System
. out . print ( a2 [ i ] + " " ); System . out . println (); } } Python # Merging two sorted arrays with O(1) #
extra space # Function to find next gap. def nextGap ( gap ): if ( gap <= 1 ): return 0 return ( gap // 2 ) +
( gap % 2 ) def merge ( arr1 , arr2 , n , m ): gap = n + m gap = nextGap ( gap ) while gap > 0 : #
comparing elements in # the first array. i = 0 while i + gap < n : if ( arr1 [ i ] > arr1 [ i + gap ]): arr1 [ i ],
arr1 [ i + gap ] = arr1 [ i + gap ], arr1 [ i ] i += 1 # comparing elements in both arrays. j = gap - n if gap >
n else 0 while i < n and j < m : if ( arr1 [ i ] > arr2 [ j ]): arr1 [ i ], arr2 [ j ] = arr2 [ j ], arr1 [ i ] i += 1 j += 1 if
( j < m ): # comparing elements in the # second array. j = 0 while j + gap < m : if ( arr2 [ j ] > arr2 [ j +
gap ]): arr2 [ j ], arr2 [ j + gap ] = arr2 [ j + gap ], arr2 [ j ] j += 1 gap = nextGap ( gap ) # Driver code if
__name__ == "__main__" : a1 = [ 10 , 27 , 38 , 43 , 82 ] a2 = [ 3 , 9 ] n = len ( a1 ) m = len ( a2 ) #
Function Call merge ( a1 , a2 , n , m ) print ( "First Array: " , end = "" ) for i in range ( n ): print ( a1 [ i ],
end = " " ) print () print ( "Second Array: " , end = "" ) for i in range ( m ): print ( a2 [ i ], end = " " ) print ()
# This code is contributed # by ChitraNayal C# // C# program for Merging two sorted arrays // with O(1)
extra space using System ; class GFG { // Function to find next gap. static int nextGap ( int gap ) { if (
gap <= 1 ) return 0 ; return ( gap / 2 ) + ( gap % 2 ); } private static void merge ( int [] arr1 , int [] arr2 , int
n , int m ) { int i , j , gap = n + m ; for ( gap = nextGap ( gap ); gap > 0 ; gap = nextGap ( gap )) { //
comparing elements in the first // array. for ( i = 0 ; i + gap < n ; i ++ ) if ( arr1 [ i ] > arr1 [ i + gap ]) { int
temp = arr1 [ i ]; arr1 [ i ] = arr1 [ i + gap ]; arr1 [ i + gap ] = temp ; } // comparing elements in both
arrays. for ( j = gap > n ? gap - n : 0 ; i < n && j < m ; i ++ , j ++ ) if ( arr1 [ i ] > arr2 [ j ]) { int temp = arr1 [
i ]; arr1 [ i ] = arr2 [ j ]; arr2 [ j ] = temp ; } if ( j < m ) { // comparing elements in the // second array. for ( j
= 0 ; j + gap < m ; j ++ ) if ( arr2 [ j ] > arr2 [ j + gap ]) { int temp = arr2 [ j ]; arr2 [ j ] = arr2 [ j + gap ]; arr2
[ j + gap ] = temp ; } } } } // Driver code public static void Main () { int [] a1 = { 10 , 27 , 38 , 43 , 82 }; int []
a2 = { 3 , 9 }; // Function Call merge ( a1 , a2 , a1 . Length , a2 . Length ); Console . Write ( "First Array:
" ); for ( int i = 0 ; i < a1 . Length ; i ++ ) { Console . Write ( a1 [ i ] + " " ); } Console . WriteLine ();
Console . Write ( "Second Array: " ); for ( int i = 0 ; i < a2 . Length ; i ++ ) { Console . Write ( a2 [ i ] + " "
); } } } // This code is contributed by Sam007. JavaScript // Javascript program for Merging two sorted
arrays // with O(1) extra space // Function to find next gap. function nextGap ( gap ) { if ( gap <= 1 )
return 0 ; return parseInt ( gap / 2 , 10 ) + ( gap % 2 ); } function merge ( arr1 , arr2 , n , m ) { let i , j , gap
= n + m ; for ( gap = nextGap ( gap ); gap > 0 ; gap = nextGap ( gap )) { // comparing elements in the
first // array. for ( i = 0 ; i + gap < n ; i ++ ) if ( arr1 [ i ] > arr1 [ i + gap ]) { let temp = arr1 [ i ]; arr1 [ i ] =
arr1 [ i + gap ]; arr1 [ i + gap ] = temp ; } // comparing elements in both arrays. for ( j = gap > n ? gap - n :
0 ; i < n && j < m ; i ++ , j ++ ) if ( arr1 [ i ] > arr2 [ j ]) { let temp = arr1 [ i ]; arr1 [ i ] = arr2 [ j ]; arr2 [ j ] =
temp ; } if ( j < m ) { // comparing elements in the // second array. for ( j = 0 ; j + gap < m ; j ++ ) if ( arr2 [
j ] > arr2 [ j + gap ]) { let temp = arr2 [ j ]; arr2 [ j ] = arr2 [ j + gap ]; arr2 [ j + gap ] = temp ; } } } } let a1 = [
10 , 27 , 38 , 43 , 82 ]; let a2 = [ 3 , 9 ]; // Function Call merge ( a1 , a2 , a1 . length , a2 . length );
console . log ( "First Array: " ); for ( let i = 0 ; i < a1 . length ; i ++ ) { console . log ( a1 [ i ] + " " ); }
console . log ( "</br>" ); console . log ( "Second Array: " ); for ( let i = 0 ; i < a2 . length ; i ++ ) { console .
log ( a2 [ i ] + " " ); } PHP <?php // Merging two sorted arrays // with O(1) extra space // Function to find
next gap. function nextGap ( $gap ) { if ( $gap <= 1 ) return 0 ; return ( $gap / 2 ) + ( $gap % 2 ); }
function merge ( $arr1 , $arr2 , $n , $m ) { $i ; $j ; $gap = $n + $m ; for ( $gap = nextGap ( $gap ); $gap
> 0 ; $gap = nextGap ( $gap )) { // comparing elements // in the first array. for ( $i = 0 ; $i + $gap < $n ;
$i ++ ) if ( $arr1 [ $i ] > $arr1 [ $i + $gap ]) { $tmp = $arr1 [ $i ]; $arr1 [ $i ] = $arr1 [ $i + $gap ]; $arr1 [ $i
+ $gap ] = $tmp ; } // comparing elements // in both arrays. for ( $j = $gap > $n ? $gap - $n : 0 ; $i < $n
&& $j < $m ; $i ++ , $j ++ ) if ( $arr1 [ $i ] > $arr2 [ $j ]) { $tmp = $arr1 [ $i ]; $arr1 [ $i ] = $arr2 [ $j ];
$arr2 [ $j ] = $tmp ; } if ( $j < $m ) { // comparing elements in // the second array. for ( $j = 0 ; $j + $gap <
$m ; $j ++ ) if ( $arr2 [ $j ] > $arr2 [ $j + $gap ]) { $tmp = $arr2 [ $j ]; $arr2 [ $j ] = $arr2 [ $j + $gap ];
```

$arr2 [$j + $gap] = $tmp ; } } } echo "First Array: " ; for ($i = 0 ; $i < $n ; $i ++) echo $arr1 [$i] . " " ; echo " \n Second Array: " ; for ($i = 0 ; $i < $m ; $i ++) echo $arr2 [$i] . " " ; echo " \n " ; } // Driver code $a1 = array (10 , 27 , 38 , 43 , 82); $a2 = array (3 , 9); $n = sizeof ($a1); $m = sizeof ($a2); // Function Call merge ($a1 , $a2 , $n , $m); // This code is contributed // by mits. ?> Output First Array: 3 9 10 27 38 Second Array: 43 82 Time Complexity: O(n + m), as the gap is considered to be n+m. Auxiliary Space: O(1) Another method in O(m+n) time complexity: Here we use the below technique: Suppose we have a number A and we want to convert it to a number B and there is also a constraint that we can recover number A any time without using other variable.To achieve this we choose a number N which is greater than both numbers and add B*N in A. so A --> A+B*N To get number B out of (A+B*N) we divide (A+B*N) by N. so (A+B*N)/N = B. To get number A out of (A+B*N) we take modulo with N. so (A+B*N)%N = A. -> In short by taking modulo we get old number back and taking divide we new number. We first find the maximum element of both arrays and increment it by one to avoid collision of 0 and maximum element during modulo operation. The idea is to traverse both arrays from starting simultaneously. Let's say an element in a is a[i] and in b is b[j] and k is the position at where the next minimum number will come. Now update value a[k] if k<n else b[k-n] by adding min(a[i],b[j])*maximum_element. After updating all the elements divide all the elements by maximum_element so we get the updated array back. Below is the implementation of the above idea: C++ #include <bits/stdc++.h> using namespace std ; void mergeArray (int a [], int b [], int n , int m) { int mx = 0 ; // Find maximum element of both array for (int i = 0 ; i < n ; i ++) { mx = max (mx , a [i]); } for (int i = 0 ; i < m ; i ++) { mx = max (mx , b [i]); } // increment by one to avoid collision of 0 and maximum // element of array in modulo operation mx ++ ; int i = 0 , j = 0 , k = 0 ; while (i < n && j < m && k < (n + m)) { // recover back original element to compare int e1 = a [i] % mx ; int e2 = b [j] % mx ; if (e1 <= e2) { // update element by adding multiplication // with new number if (k < n) a [k] += (e1 * mx); else b [k - n] += (e1 * mx); i ++ ; k ++ ; } else { // update element by adding multiplication // with new number if (k < n) a [k] += (e2 * mx); else b [k - n] += (e2 * mx); j ++ ; k ++ ; } } // process those elements which are left in array a while (i < n) { int el = a [i] % mx ; if (k < n) a [k] += (el * mx); else b [k - n] += (el * mx); i ++ ; k ++ ; } // process those elements which are left in array b while (j < m) { int el = b [j] % mx ; if (k < n) a [k] += (el * mx); else b [k - n] += (el * mx); j ++ ; k ++ ; } // finally update elements by dividing // with maximum element for (int i = 0 ; i < n ; i ++) a [i] = a [i] / mx ; // finally update elements by dividing // with maximum element for (int i = 0 ; i < m ; i ++) b [i] = b [i] / mx ; return ; } // Driver Code int main () { int a [] = { 3 , 5 , 6 , 8 , 12 }; int b [] = { 1 , 4 , 9 , 13 }; int n = sizeof (a) / sizeof (int); // Length of a int m = sizeof (b) / sizeof (int); // length of b // Function Call mergeArray (a , b , n , m); cout << "First array : " ; for (int i = 0 ; i < n ; i ++) cout << a [i] << " " ; cout << endl ; cout << "Second array : " ; for (int i = 0 ; i < m ; i ++) cout << b [i] << " " ; cout << endl ; return 0 ; } Java import java.io.* ; class GFG { static void mergeArray (int a [] , int b [] , int n , int m) { int mx = 0 ; // Find maximum element of both array for (int i = 0 ; i < n ; i ++) { mx = Math . max (mx , a [i]); } for (int i = 0 ; i < m ; i ++) { mx = Math . max (mx , b [i]); } // Increment one two avoid collision of // 0 and maximum element of array in // modulo operation mx ++ ; int i = 0 , j = 0 , k = 0 ; while (i < n && j < m && k < (n + m)) { // Recover back original element // to compare int e1 = a [i] % mx ; int e2 = b [j] % mx ; if (e1 <= e2) { // Update element by adding // multiplication with new number if (k < n) a [k] += (e1 * mx); else b [k - n] += (e1 * mx); i ++ ; k ++ ; } else { // Update element by adding // multiplication with new number if (k < n) a [k] += (e2 * mx); else b [k - n] += (e2 * mx); j ++ ; k ++ ; } } // Process those elements which are // left in array a while (i < n) { int el = a [i] % mx ; if (k < n) a [k] += (el * mx); else b [k - n] += (el * mx); i ++ ; k ++ ; } // Process those elements which are // left in array a while (j < m) { int el = b [j] % mx ; if (k < n) b [k] += (el * mx); else b [k - n] += (el * mx); j ++ ; k ++ ; } // Finally update elements by dividing // with maximum element for (int in = 0 ; in < n ; in ++) a [in] = a [in] / mx ; // Finally update elements by dividing // with maximum element for (int in = 0 ; in < m ; in ++) b [in] = b [in] / mx ; return ; } // Driver code public static void main (String [] args) { int a [] = { 3 , 5 , 6 , 8 , 12 }; int b [] = { 1 , 4 , 9 , 13 }; // Length of a int n = a . length ; // length of b int m = b . length ; // Function Call mergeArray (a , b , n , m); System . out . print ("First array : "); for (int i = 0 ; i < n ; i ++) System . out . print (a [i] + " "); System . out . println (); System . out . print ("Second array : "); for (int i = 0 ; i < m ; i ++) System . out . print (b [i] + " "); System . out . println (); } } // This code is contributed by yashbeersingh42 Python # Merging two sorted arrays with O(1) # extra space # Function to find next gap. def mergeArray (a , b , n , m): mx = 0 # Find maximum element of both array for i in range (n): mx = max (mx , a [i]) for i in range (m): mx = max (mx , b [i]) # Increment one two avoid collision of # 0 and maximum element of array in # modulo operation mx += 1 i = 0 j = 0 k = 0 while (i < n & j < m & k < (n + m)): # Recover back original element # to compare e1 = (a [i] % mx) e2 = (b [j] % mx) if (e1 <= e2): # Update element by adding # multiplication with new number if (k < n): a [k] += (e1 * mx) else : b [k - n] += (e1 * mx) i += 1 k += 1 else : # Update element by adding # multiplication with new number if (k < n): a [k] += (e2 * mx) else : b [k - n] += (e2 * mx) j += 1 ; k += 1 ; # Process those elements which are # left in array a while (i < n): el = (a [i] % mx) if (k < n): a [k] += (el * mx) else : b [k - n] += (el * mx) i += 1 ; k += 1 ; # Process those elements which are # left in array b while (j < m): el = (b [j] % mx) if (k < n): a [k] += (el * mx) else : b [k - n] += (el * mx) j += 1 ; k += 1 ; # Finally update elements by dividing # with maximum element for In in range (n): a [In] = int (float (a [In]) / float (mx)) # Finally update elements by dividing # with maximum element for In in range (m): b [In] = int (float (b [In]) / float (mx)) return # Driver code if __name__ == "__main__" : a = [3 , 5 , 6 , 8 , 12] b = [1 , 4 , 9 , 13] n = len (a) m = len (b) # Function Call mergeArray (a , b , n , m) print ("First Array: " , end = "") for i in range (n): print (a [i], end = " ") print () print ("Second Array: " , end = "") for i in range (m): print (b [i], end = " ") print () # This code is contributed by Aarti_Rathi C# using System ; public class GFG { static void mergeArray (int [] a , int [] b , int n , int m) { int mx = 0 ; // Find maximum element of both array for (int I = 0 ; I < n ; I ++) { mx = Math . Max (mx , a [I]); } for (int I = 0 ; I < m ; I ++) { mx = Math . Max (mx , b [I]); } // Increment one two avoid collision of // 0 and maximum element of array in // modulo operation mx ++ ; int i = 0 , j = 0 , k = 0 ; while (i < n && j < m && k < (n + m)) { // Recover back original element // to compare int e1 = a [i] % mx ; int e2 = b [j] % mx ; if (e1 <= e2) { // Update element by adding // multiplication with new number if (k < n) a [k] += (e1 * mx); else b [k - n] += (e1 * mx); i ++ ; k ++ ; } else { // Update element by adding // multiplication with new number if (k < n) a [k] += (e2 * mx); else b [k - n] += (e2 * mx); j ++ ; k ++ ; } } // Process those elements which are // left in array a while (i < n) { int el = a [i] % mx ; if (k < n) a [k] += (el * mx); else b [k - n] += (el * mx); i ++ ; k ++ ; } // Process those elements which are // left in array a while (j < m) { int el = b [j] % mx ; if (k < n) b [k] += (el * mx); else b [k - n] += (el * mx); j ++ ; k ++ ; } // Finally update elements by dividing // with maximum element for (int In = 0 ; In < n ; In ++) a [In] = a [In] / mx ; // Finally update elements by dividing // with maximum element for (int In = 0 ; In < m ; In ++) b [In] = b [In] / mx ; return ; } // Driver code static public void Main (){ int [] a = { 3 , 5 , 6 , 8 , 12 }; int [] b = { 1 , 4 , 9 , 13 }; // Length of a int n = a . Length ; // length of b int m = b . Length ; // Function Call mergeArray (a , b , n , m); Console . Write ("First array : "); for (int i = 0 ; i < n ; i ++) Console . Write (a [i] + " "); Console . WriteLine (); Console . Write ("Second array : "); for (int i = 0 ; i < m ; i ++) Console . Write (b [i] + " "); Console . WriteLine (); } } // This code is contributed by avanitrachhadiya2155 JavaScript function mergeArray (a , b , n , m) { let mx = 0 ; // Find maximum element of both array for (let i = 0 ; i < n ; i ++) { mx = Math . max (mx , a [i]); } for (let i = 0 ; i < m ; i ++) { mx = Math . max (mx , b [i]); } // Increment by one to avoid collision // of 0 and maximum element of array // in modulo operation mx ++ ; let i = 0 , j = 0 , k = 0 ; while (i < n && j < m && k < (n + m)) { // Recover back original element // to compare let e1 = a [i] % mx ; let e2 = b [j] % mx ; if (e1 <= e2) { // Update element by adding // multiplication with new number if (k < n) a [k] += (e1 * mx); else b [k - n] += (e1 * mx); i ++ ; k ++ ; } else { // Update element by adding // multiplication with new number if (k < n) a [k] += (e2 * mx); else b [k - n] += (e2 * mx); j ++ ; k ++ ; } } // Process those elements which // are left in array a while (i < n) { let el = a [i] % mx ; if (k < n) a [k] += (el * mx); else b [k - n] += (el * mx); i ++ ; k ++ ; } // Process those elements which // are left in array b while (j < m) { let el = b [j] % mx ; if (k < n) a [k] += (el * mx); else b [k - n] += (el * mx); j ++ ; k ++ ; } // Finally update elements by dividing // with maximum element for (let i = 0 ; i < n ; i ++) a [i] = parseInt (a [i] / mx); // Finally update elements by dividing // with maximum element for (let i = 0 ; i < m ; i ++) b [i] = parseInt (b [i] / mx); return ; } // Driver Code var a = [3 , 5 , 6 , 8 , 12]; var b = [1 , 4 , 9 , 13]; // Length of a var n = a . length ; // Length of b var m = b . length ; // Function Call mergeArray (a , b , n , m); console . log ("First array : "); for (let i = 0 ; i < n ; i ++) console . log (a [i] + " "); console . log ('<br>') console . log ("Second array : "); for (let i = 0 ; i < m ; i ++) console . log (b [i] + " "); // This code is contributed by Potta Lokesh Output First array : 1 3 4 5 6 Second array : 8 9 12 13 Time Complexity: O(m+n) The space complexity of the given program is O(n+m), where n and m are the lengths of the input arrays a and b, respectively. This is because two additional arrays of size n and m are created to store the merged elements, and no additional space is used apart from that. The input arrays are modified in place, so they do not contribute to the space complexity. We have discussed a simple and efficient solution in the below post. In this post, a better and simple solution is discussed. The idea is: We will traverse the first array and compare it with the first element of the second array. If the first element of the second array is smaller than the first array, we will swap and then sort the second array. First, we have to traverse array1 and then compare it with the first element of array2. If it is less than array1 then we swap both. After swapping we are going to sort the array2 again so that the smallest element of the array2 comes at the first position and we can again swap with the array1 To sort the array2 we will store the first element of array2 in a variable and left shift all the elements and store the first element in array2 in the last. Implementation: C++ #include <bits/stdc++.h> using namespace std ; void mergeArray (int arr1 [], int arr2 [], int n , int m) { // Now traverse the array1 and if // arr2 first element // is less than arr1 then swap for (int i = 0 ; i < n ; i ++) { if (arr1 [i] > arr2 [0]) { // Swap int temp = arr1 [i]; arr1 [i] = arr2 [0]; arr2 [0] = temp ; // After swapping we have to sort the array2 // again so that it can be again swap with // arr1 // We will store the firstElement of array2 // and left shift all the element and store // the firstElement in arr2[k-1] int firstElement = arr2 [0]; int k ; for (k = 1 ; k < m && arr2 [k] < firstElement ; k ++) { arr2 [k - 1] = arr2 [k]; } arr2 [k - 1] = firstElement ; } } // Read the arr1 for (int i = 0 ; i < n ; i ++) { cout << arr1 [i] << " " ; } cout << endl ; // Read the arr2 for (int i = 0 ; i < m ; i ++) { cout << arr2 [i] << " " ; } } // Driver Code int main () { int arr1 [] = { 1 , 3 , 5 , 7 }; int arr2 [] = { 0 , 2 , 6 , 8 , 9 }; int n = sizeof (arr1) / sizeof (arr1 [0]), m = sizeof (arr2) / sizeof (arr2 [0]); mergeArray (arr1 , arr2 , n , m); } // This code is contributed by yashbeersingh42 Java import java.io.* ; class GFG { public static void mergeArray (int [] arr1 , int [] arr2) { // length of first arr1 int n = arr1 . length ; // length of second arr2 int m = arr2 . length ; // Now traverse the array1 and if // arr2 first element // is less than arr1 then swap for (int i = 0 ; i < n ; i ++) { if (arr1 [i] > arr2 [0]) { // swap int temp = arr1 [i] ; arr1 [i] = arr2 [0] ; arr2 [0] = temp ; // after swapping we have to sort the array2 // again so that it can be again swap with // arr1 // we will store the firstElement of array2 // and left shift all the element and store // the firstElement in arr2[k-1] int firstElement = arr2 [0] ; int k ; for (k = 1 ; k < m && arr2 [k] < firstElement ; k ++) { arr2 [k - 1] = arr2 [k] ; } arr2 [k - 1] = firstElement ; } } // read the arr1 for (int i : arr1) { System . out . print (i + " "); } System . out . println (); // read the arr2 for (int i : arr2) { System . out . print (i + " "); } } // Driver Code public static void main (String [] args) { int [] arr1 = { 1 , 3 , 5 , 7 }; int [] arr2 = { 0 , 2 , 6 , 8 , 9 }; mergeArray (arr1 , arr2); } } Python def mergeArray (arr1 , arr2): # length of first arr1 n = len (arr1); # length of second arr2 m = len (arr2); # Now traverse the array1 and if # arr2 first element # is less than arr1 then swap for i in range (n): if (arr1 [i] > arr2 [0]): # swap temp = arr1 [i]; arr1 [i] = arr2 [0]; arr2 [0] = temp ; # after swapping we have to sort the array2 # again so that it can be again swap with # arr1 # we will store the firstElement of array2 # and left shift all the element and store # the firstElement in arr2[k-1] firstElement = arr2 [0]; k = 1 ; for p in range (1 , m): if (arr2 [k] < firstElement): arr2 [k - 1] = arr2 [k]; k += 1 ; arr2 [k - 1] = firstElement ; # read the arr1 #for (i : arr1): # print(i, end=""); print (arr1); print (arr2); # read the arr2 #for (i : arr2): # print(i, end=""); # Driver Code if __name__ == '__main__' : arr1 = [1 , 3 , 5 , 7]; arr2 = [0 , 2 , 6 , 8 , 9]; mergeArray (arr1 , arr2); # This code is contributed by Rajput-Ji C# using System ; class GFG { public static void mergeArray (int [] arr1 , int [] arr2) { // Length of first arr1 int n = arr1 . Length ; // Length of second arr2 int m = arr2 . Length ; // Now traverse the array1 and if // arr2 first element // is less than arr1 then swap for (int i = 0 ; i < n ; i ++) { if (arr1 [i] > arr2 [0]) { // Swap int temp = arr1 [i]; arr1 [i] = arr2 [0]; arr2 [0] = temp ; // After swapping we have to sort the array2 // again so that it can be again swap with // arr1 // We will store the firstElement of array2 // and left shift all the element and store // the firstElement in arr2[k-1] int firstElement = arr2 [0]; int k ; for (k = 1 ; k < m && arr2 [k] < firstElement ; k ++) { arr2 [k - 1] = arr2 [k]; } arr2 [k - 1] = firstElement ; } } // Read the arr1 foreach (int i in arr1) { Console . Write (i + " "); } Console . WriteLine (); // Read the arr2 foreach (int i in arr2) { Console . Write (i + " "); } } // Driver Code static public void Main () { int [] arr1 = { 1 , 3 , 5 , 7 }; int [] arr2 = { 0 , 2 , 6 , 8 , 9 }; mergeArray (arr1 , arr2); } } // This code is contributed by rag2127 JavaScript function mergeArray (arr1 , arr2 , n , m) { // Now traverse the array1 and if // arr2 first element // is less than arr1 then swap for (let i = 0 ; i < n ; i ++) { if (arr1 [i] > arr2 [0]) { // Swap let temp = arr1 [i]; arr1 [i] = arr2 [0]; arr2 [0] = temp ; // After swapping we have to sort the array2 // again so that it can be again swap with // arr1 // We will store the firstElement of array2 // and left shift all the element and store // the firstElement in arr2[k-1] let firstElement = arr2 [0]; let k ; for (k = 1 ; k < m && arr2 [k] < firstElement ; k ++) { arr2 [k - 1] = arr2 [k]; } arr2 [k - 1] = firstElement ; } } // Read the arr1 for (let i = 0 ; i < n ; i ++) { console . log (arr1 [i] + " "); } console . log ("<br>"); // Read the arr2 for (let i = 0 ; i < m ; i ++) { console . log (arr2 [i] + " "); } } // Driver Code let arr1 = [1 , 3 , 5 , 7]; let arr2 = [0 , 2 , 6 , 8 , 9]; let n = 4 , m = 5 ; mergeArray (arr1 , arr2 , n , m); //This code is contributed by Mayank Tyagi Output 0 1 2 3 5 6 7 8 9 Time Complexity: O(n*m) Space Complexity: O(1) Comment Article Tags: Article Tags: DSA Microsoft Goldman Sachs Snapdeal Zoho Linkedin Quikr Synopsys Brocade Juniper Networks Amdocs + 7 More