***Merge function of Merge sort***

Given two sorted arrays, the task is to merge them in a sorted manner.  
**Examples:**

**Input**: arr1[] = { 1, 3, 4, 5}, arr2[] = {2, 4, 6, 8}   
**Output**: arr3[] = {1, 2, 3, 4, 4, 5, 6, 8}  
**Input**: arr1[] = { 5, 8, 9}, arr2[] = {4, 7, 8}   
**Output**: arr3[] = {4, 5, 7, 8, 8, 9}

**Method 1 (O(n1 \* n2) Time and O(n1+n2) Extra Space)**

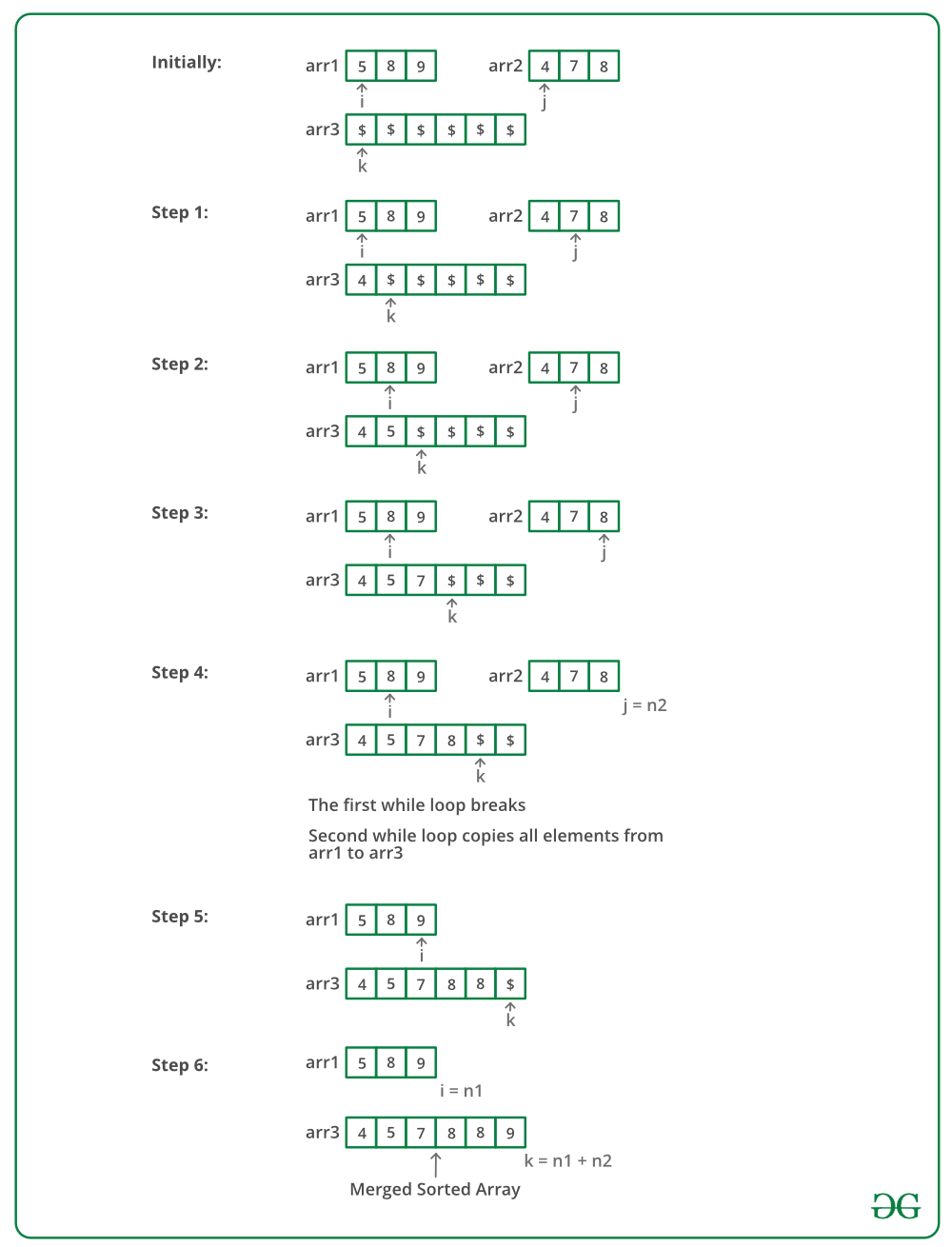
1. Create an array arr3[] of size n1 + n2.
2. Copy all n1 elements of arr1[] to arr3[]
3. Traverse arr2[] and one by one insert elements (like [insertion sort](https://www.geeksforgeeks.org/insertion-sort/)) of arr3[] to arr1[]. This step take O(n1 \* n2) time.

We have discussed implementation of above method in [Merge two sorted arrays with O(1) extra space](https://www.geeksforgeeks.org/merge-two-sorted-arrays-o1-extra-space/)

**Method 2 (O(n1 + n2) Time and O(n1 + n2) Extra Space)**   
The idea is to use Merge function of [Merge sort](https://www.geeksforgeeks.org/merge-sort/).

1. Create an array arr3[] of size n1 + n2.
2. Simultaneously traverse arr1[] and arr2[].   
   * Pick smaller of current elements in arr1[] and arr2[], copy this smaller element to next position in arr3[] and move ahead in arr3[] and the array whose element is picked.
3. If there are remaining elements in arr1[] or arr2[], copy them also in arr3[].

Below image is a dry run of the above approach:



Below is the implementation of the above approach: 

C++Java

// Java program to merge two sorted arrays

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class MergeTwoSorted

{

// Merge arr1[0..n1-1] and arr2[0..n2-1]

// into arr3[0..n1+n2-1]

public static void mergeArrays(int[] arr1, int[] arr2, int n1,

int n2, int[] arr3)

{

int i = 0, j = 0, k = 0;

// Traverse both array

while (i<n1 && j <n2)

{

// Check if current element of first

// array is smaller than current element

// of second array. If yes, store first

// array element and increment first array

// index. Otherwise do same with second array

if (arr1[i] < arr2[j])

arr3[k++] = arr1[i++];

else

arr3[k++] = arr2[j++];

}

// Store remaining elements of first array

while (i < n1)

arr3[k++] = arr1[i++];

// Store remaining elements of second array

while (j < n2)

arr3[k++] = arr2[j++];

}

public static void main (String[] args)

{

int[] arr1 = {1, 3, 5, 7};

int n1 = arr1.length;

int[] arr2 = {2, 4, 6, 8};

int n2 = arr2.length;

int[] arr3 = new int[n1+n2];

mergeArrays(arr1, arr2, n1, n2, arr3);

System.out.println("Array after merging");

for (int i=0; i < n1+n2; i++)

System.out.print(arr3[i] + " ");

}

}

/\* This code is contributed by Mr. Somesh Awasthi \*/

**Output:**

Array after merging

1 2 3 4 5 6 7 8

**Time Complexity :** O(n1 + n2)   
**Auxiliary Space :** O(n1 + n2)

**Method 3: Using Maps (O(nlog(n) + mlog(m)) Time and O(N) Extra Space)**

1. Insert elements of both arrays in a map as keys.
2. Print the keys of the map.

Below is the implementation of above approach. 

CPPJava

// Java program to merge two sorted arrays

//using maps

import java.io.\*;

import java.util.\*;

class GFG {

// Function to merge arrays

static void mergeArrays(int a[], int b[], int n, int m)

{

// Declaring a map.

// using map as a inbuilt tool

// to store elements in sorted order.

Map<Integer,Boolean> mp = new TreeMap<Integer,Boolean>();

// Inserting values to a map.

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

{

mp.put(a[i], true);

}

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

{

mp.put(b[i], true);

}

// Printing keys of the map.

for (Map.Entry<Integer,Boolean> me : mp.entrySet())

{

System.out.print(me.getKey() + " ");

}

}

// Driver Code

public static void main (String[] args)

{

int a[] = {1, 3, 5, 7}, b[] = {2, 4, 6, 8};

int size = a.length;

int size1 = b.length;

// Function call

mergeArrays(a, b, size, size1);

}

}

// This code is contributed by rag2127

**Output:**

1 2 3 4 5 6 7 8

**Time Complexity:**O( nlog(n) + mlog(m) )   
**Auxiliary Space:**O(N)