Tutorial Link https://codequotient.com/tutorials/Merge 2 Sorted Arrays/608f60e068a47dda668684ce

TUTORIAL

Merge 2 Sorted Arrays

Chapter

Topics

Given two sorted arrays A and B, the task is to merge them and create one single sorted array.

Examples:

Input :

```
A[] = {1, 5, 8}
B[] = {6, 9}
```

Output :

```
Merged Array = {1, 5, 6, 8, 9}
```

Solution Approach - 1

The naive approach is to create an array of size n+m, then insert the elements of both arrays in that array and finally sort that array.

Algorithm:

1. Create an array C[] of size n + m, where n and m is the size of array A[] and B[] respectively.
2. Traverse the two given arrays one after another and copy all their elements in C[].
3. Then sort the final array C[] using any sorting algorithm.

Clearly, the time complexity of the first step will be **O(n+m)**, and for the second step using any sorting algorithm can give us the best time complexity of **O((n+m)log(n+m))**.

Now the question arises why do we actually need to sort when it is given that the two arrays A and B are already sorted. So in the next approach, we will see how to merge two sorted arrays without explicitly doing any sorting.

Time Complexity: O((n+m) + (n+m)log(n+m))

Space Complexity: O(n+m)

C++

```cpp
#include <bits/stdc++.h>
using namespace std;

void mergeArrays(int A[], int B[], int n, int m, int C[]) {
   int j = 0;
   for (int i=0; i<n; i++)
       C[j++] = A[i];
   for (int i=0; i<m; i++)
       C[j++] = B[i];
   int N = n+m;
   sort(C, C+N);
}

int main(){
   int A[] = {1, 5, 8};
   int n = sizeof(A) / sizeof(A[0]);
   int B[] = {6, 9};
   int m = sizeof(B) / sizeof(B[0]);
   int C[n+m];
```

```
    mergeArrays(A, B, n, m, C);
    for (int i = 0; i < (n + m); i++)
        cout << C[i] << " ";
    return 0;
}


```

Solution Approach 2

The idea is to use the Merge operation of the Merge Sort.

Algorithm:

1. Create an array C[] of size n + m, and initialize a pointer k = 0 that will help us insert the elements in this final array.
2. Then start traversing the two given arrays A, B simultaneously using two pointers i, j. Every time compare A[i], B[j] and whichever is smaller, pick that element and copy to the kth position in C[] and increment the value of k as well as the pointer for the array whose element is picked.
3. Finally, once we've copied all elements from one array, we'll copy the remaining from the other into the merged array C[].

Time Complexity: O(n+m)

Space Complexity: O(n+m)

Javascript

```
function mergeArrays(A,B){
  let C = [];
  let i=0,j = 0;
  while(i<A.length && j<B.length){
    if(A[i] < B[j]){
      C.push(A[i++])
    }else{
      C.push(B[j++])
    }
  }

  while(i<A.length){
    C.push(A[i++])

  }
  while(j<B.length){
    C.push(B[j++])
  }
  return C;
}

function main(){
  let A = [1,5,8]
  let B = [6,9]
  let C = mergeArrays(A,B)
  console.log(C.join(' '))
}

main()
```

C

```c
#include <stdio.h>

void mergeArrays(int A[], int B[], int n, int m, int C[]){
   int i = 0, j = 0, k = 0;
   while (i < n && j < m) {
       if (A[i] <= B[j])
           C[k++] = A[i++];
       else
           C[k++] = B[j++];
   }
   while (i < n)
       C[k++] = A[i++];
   while (j < m)
       C[k++] = B[j++];
}
int main(){
   int A[] = {1, 5, 8};
   int n = sizeof(A) / sizeof(A[0]);
   int B[] = {6, 9};
   int m = sizeof(B) / sizeof(B[0]);
   int C[n+m];
   mergeArrays(A, B, n, m, C);
   for (int i = 0; i < (n + m); i++)
```

```
       printf("%d ", C[i]);
 return 0;
}
```

Java

```
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main {
    public static void mergeArrays(int[] A, int[] B, int n, int m, int[] C) {
        int i = 0, j = 0, k = 0;

        while (i<n && j <m){
            if (A[i] < B[j])
                C[k++] = A[i++];
            else
                C[k++] = B[j++];
        }

        while (i < n)
            C[k++] = A[i++];

        while (j < m)
            C[k++] = B[j++];
    }

    public static void main (String[] args) {
        int[] A = {1, 5, 8};
        int n = A.length;

        int[] B = {6, 9};
        int m = B.length;

        int[] C = new int[n+m];

        mergeArrays(A, B, n, m, C);

        for (int i=0; i < n+m; i++)
            System.out.print(C[i] + " ");
    }
}

```

Python 3

```python
def mergeArrays(a,b):
  c= []
  i=0;j=0
  while(i<len(a) and j<len(b)):
    if(a[i]<b[j]):
      c.append(a[i])
      i+=1
    else:
      c.append(b[j])
      j+=1
  while(i<len(a)):
    c.append(a[i])
    i+=1;
  while(j<len(b)):
    c.append(b[j])
    j+=1
  return c

if __name__=='__main__':
  A = [1,5,8]
  B = [6,9]
  C = mergeArrays(A,B)
  print(' '.join( str(i) for i in C))
```

C++

```cpp
#include <iostream>
using namespace std;
void mergeArrays(int A[], int B[], int n, int m,int C[]) {
   int i = 0, j = 0, k = 0;
   while (i<n && j <m) {
       if (A[i] < B[j])
           C[k++] = A[i++];
       else
           C[k++] = B[j++];
   }
   while (i < n)
       C[k++] = A[i++];
    while (j < m)
```

```
        C[k++] = B[j++];
}

int main() {
    int A[] = {1, 5, 8};
    int n = sizeof(A) / sizeof(A[0]);
    int B[] = {6, 9};
    int m = sizeof(B) / sizeof(B[0]);
    int C[n+m];
    mergeArrays(A, B, n, m, C);
    for (int i = 0; i < (n + m); i++)
        cout<<C[i]<<" ";
    return 0;
}

```

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/cVmv3VvBTp0" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>