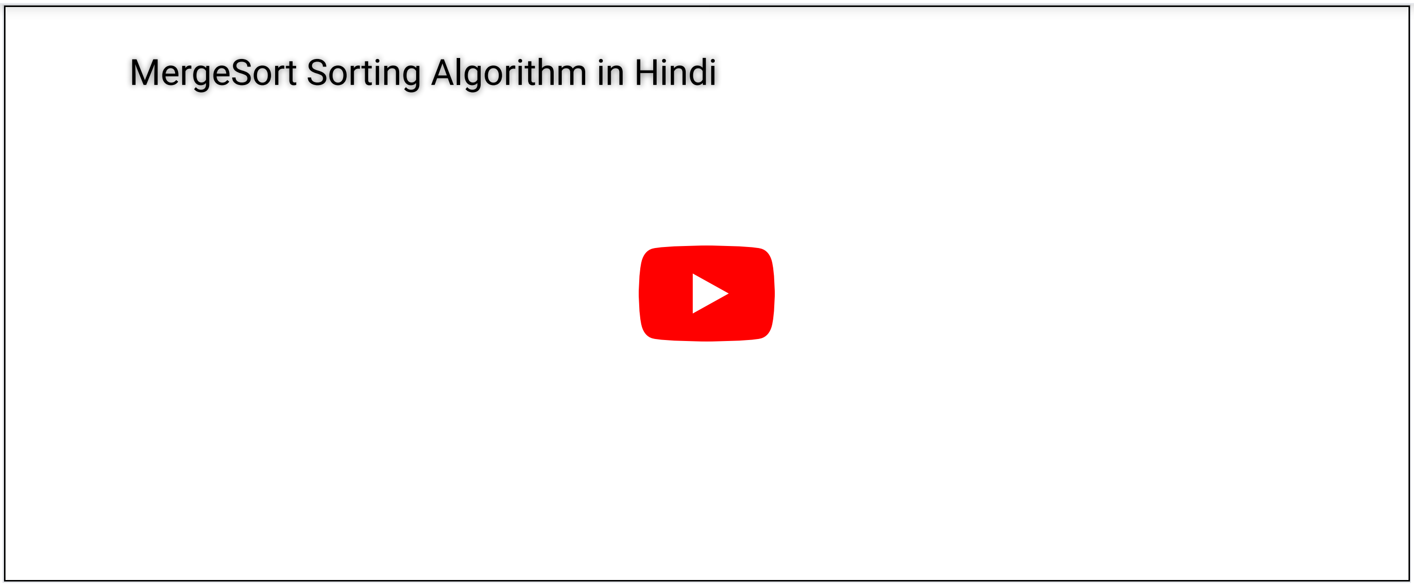


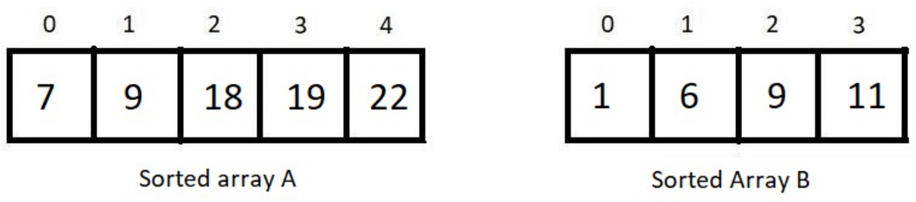
MergeSort Sorting Algorithm in Hindi

We have so far covered all our sorting algorithms except one or two. We learned about the bubble sort, the insertion sort, the selection sort, and the quicksort. Now it's time to move onto our next sorting algorithm, the merge sort algorithm. You will understand it very easily once I explain the working of the algorithm using a few intuitive examples to you.

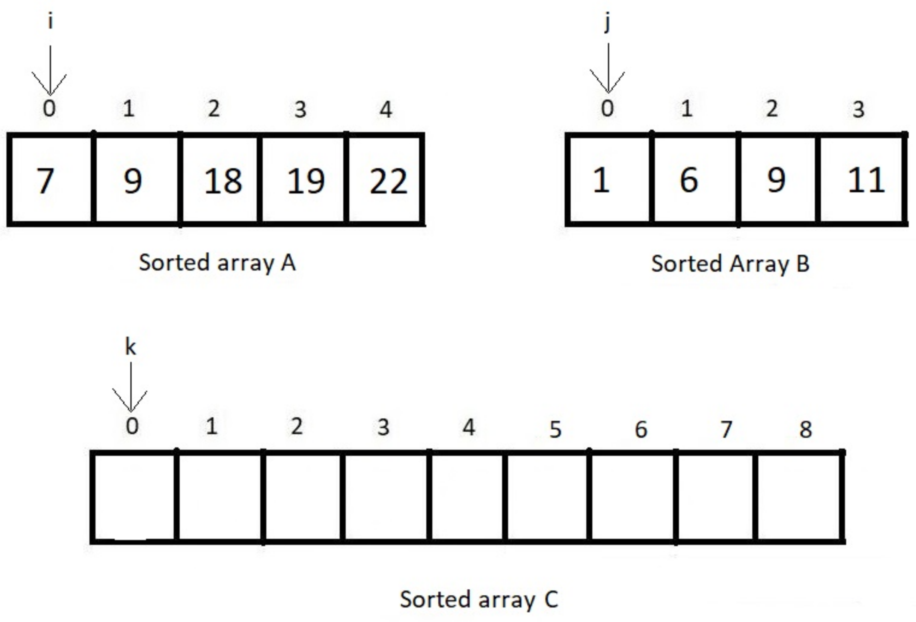
But before we proceed, I would like to give you the reason why we call it the **merge** sort algorithm. In this algorithm, we divide the arrays into subarrays and subarrays into more subarrays until the size of each subarray becomes 1. Since arrays with a single element are always considered sorted, this is where we merge. Merging two sorted subarrays creates another sorted subarray. I'll show you first how merging two sorted subarrays works.

Merging Procedure:

Suppose we have two sorted arrays, A and B, of sizes 5 and 4, respectively.

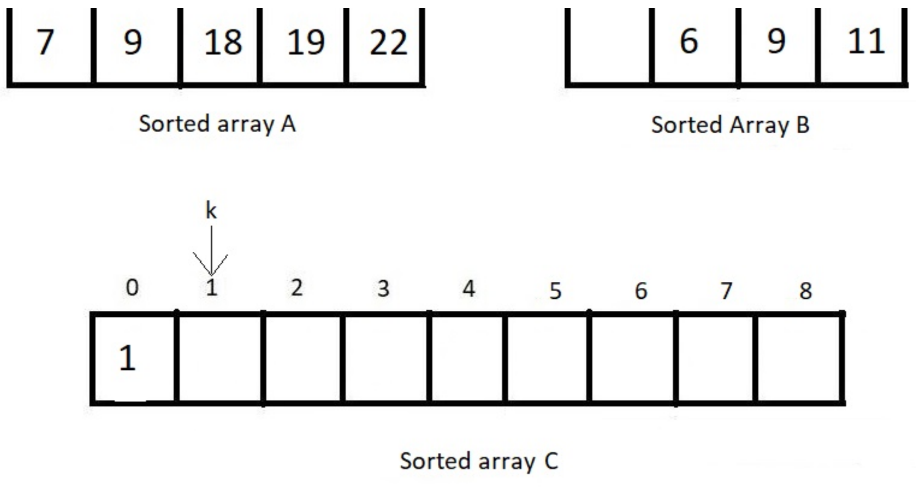


1. And we apply merging on them. Then the first job would be to create another array C with size being the sum of both the raw arrays' sizes. Here the sizes of A and B are 5 and 4, respectively. So, the size of array C would be 9.
2. Now, we maintain three index variables i, j, and k. i looks after the first element in array A, j looks after the first element in array B, and k looks after the position in array C to place the next element in.

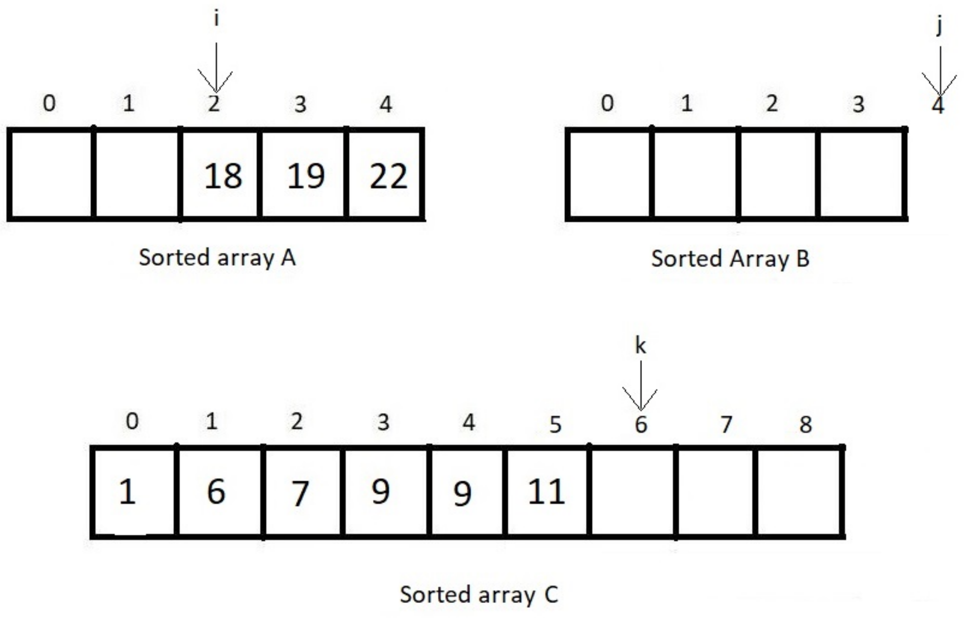


3. Initially, all i, j, and k are equal to 0.
4. Now, we compare the elements at index *i* of array A and index *j* of array B and see which one is smaller. Fill in the smaller element at index *k* of array C and increment *k* by 1. Also, increment the index variable of the array we fetched the smaller element from.
5. Here, A[i] is greater than B[j]. Hence we fill C[k] with B[j] and increase k and j by 1.

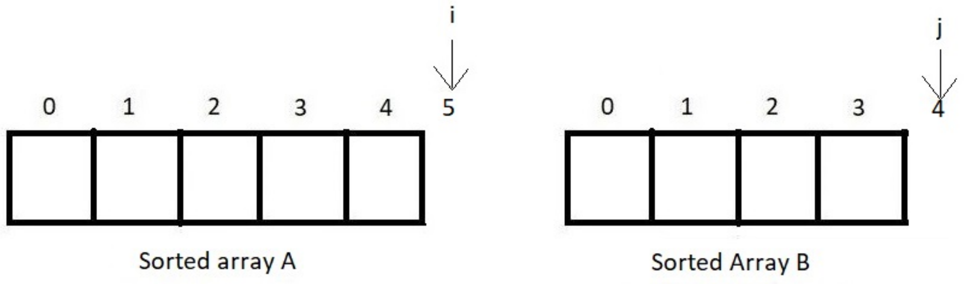
i
j
0 1 2 3 4
0 1 2 3

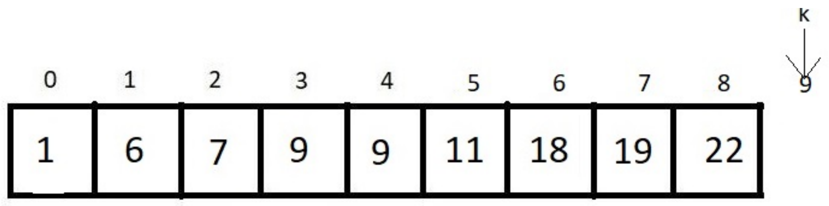


6. We continue doing step 5 until one of the arrays, A or B, gets empty.



Here, array B inserted all its elements in the merged array C. Since we are only left with the elements of element A, we simply put them in the merged array as they are. This will result in our final merged array C.



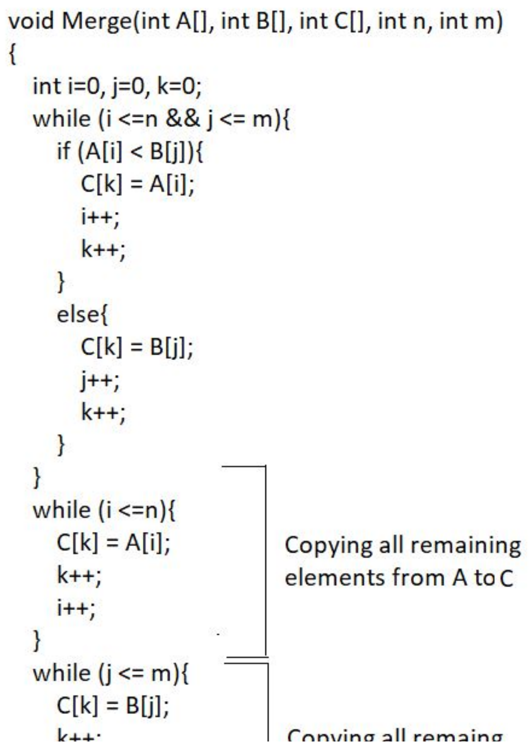


Sorted array C

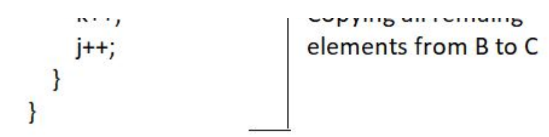
I hope you understand the merging procedure. This is an important concept in learning the merge sort algorithm. Be sure not to skip this. Even the programming part of the merge procedure is not that tough. You just follow these steps:

1. Take both the arrays and their sizes to be merged as the parameters of the merge function. By summing the sizes of the two arrays, we can create one larger array.
2. Create three index variables *i,j & k.* And initialize all of them with 0.
3. And then run a while loop with the condition that both the index variables *i and j* don't exceed their respective array limits.
4. Now, at each run, see if A[i] is smaller than B[j], if yes, make C[k] = A[i] and increase both *i and k* by 1, else C[k] = B[j] and both *j and k* are incremented by 1.
5. And when the loop finishes, either array A or B or both get finished. And now you run two while loops for each array A and B, and insert all the remaining elements as they are in the array C. And you are done merging.

The pseudocode for the above procedure has been attached below.



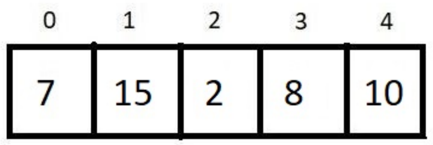
```
void Merge(int A[], int B[], int C[], int n, int m)
{
   int i=0, j=0, k=0;
   while (i <=n && j <= m){
      if (A[i] < B[j]){
         C[k] = A[i];
         i++;
         k++;
      }
      else{
         C[k] = B[j];
         j++;
         k++;
      }
   }
   while (i <=n){
      C[k] = A[i];
      k++;
      i++;
   }
   while (j <= m){
      C[k] = B[j];
      k++;
```



```
            j++;
        }
    }
```

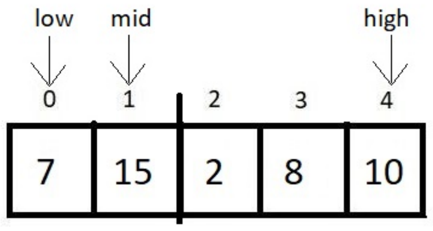
elements from B to C

Now, this would quite not be our situation when sorting an array using the merge sort. We wouldn't have two different arrays A and B, rather a single array having two sorted subarrays. Now, I'd show you how to merge two sorted subarrays of a single array in the array itself.
Suppose there is an array A of 5 elements and contains two sorted subarrays of length 2 and 3 in itself.

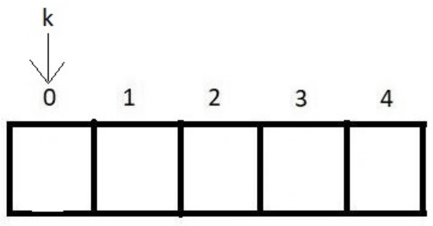


Unsorted Array A

To merge both the sorted subarrays and produce a sorted array of length 5, we will create an auxiliary array B of size 5. Now the process would be more or less the same, and the only change we would need to make is to consider the first index of the first subarray as *low* and the last index of the second subarray as *high*. And mark the index prior to the first index of the second subarray as *mid.*

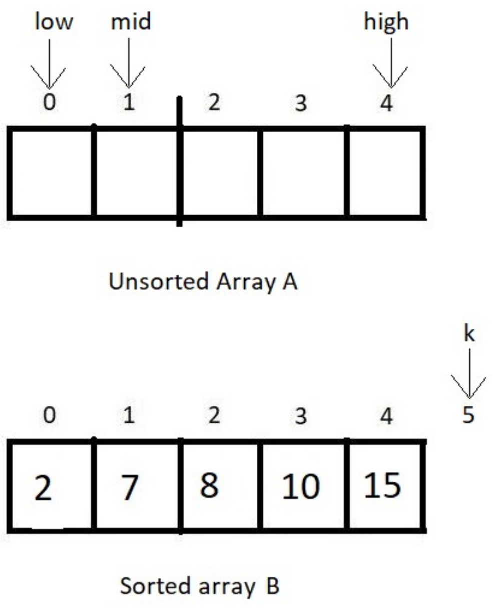


Unsorted Array A

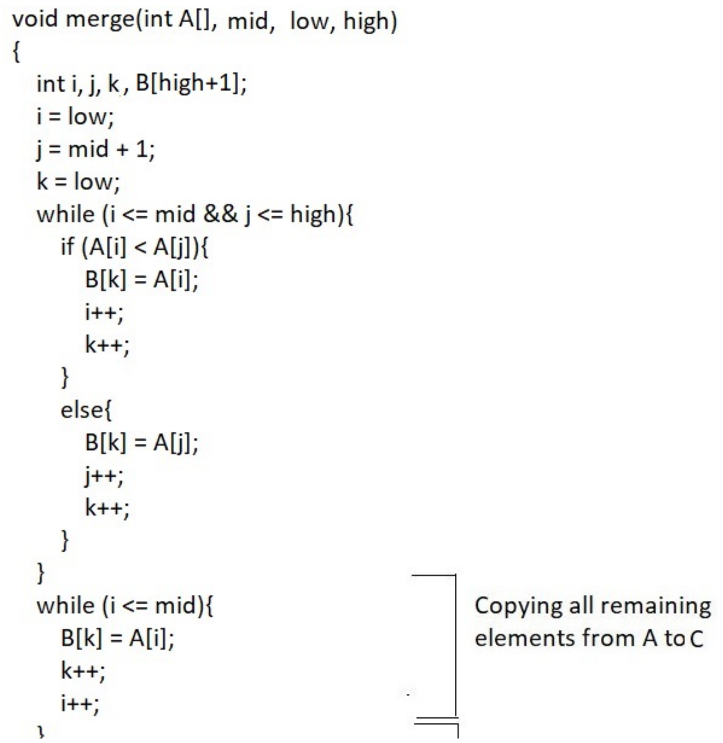


Sorted array B

Previously we had index variables *i, j, and k* initialised with 0 of their respective arrays. But here, i gets initialised with *low,* j gets initialised with *mid+1,* and k gets initialised with *low* only. And similar to what we did earlier, i runs from *low to mid*, j runs from *mid+1 to high*, and until and unless they both get all their elements merged, we continue filling elements in array B.

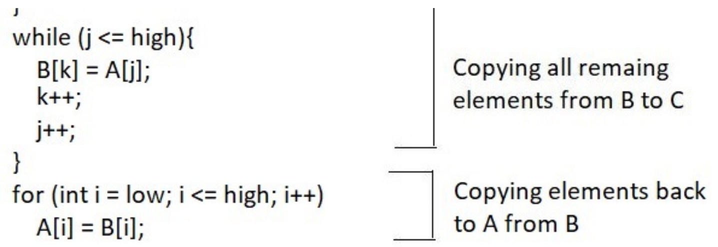


After all the elements get filled in array C, we revert back to our original array A and fill the sorted elements again from low to high, making our array merge-sorted.



```
void merge(int A[], mid, low, high)
{
   int i, j, k, B[high+1];
   i = low;
   j = mid + 1;
   k = low;
   while (i <= mid && j <= high){
      if (A[i] < A[j]){
         B[k] = A[i];
         i++;
         k++;
      }
      else{
         B[k] = A[j];
         j++;
         k++;
      }
   }
   while (i <= mid){
      B[k] = A[i];
      k++;
      i++;
   }
```

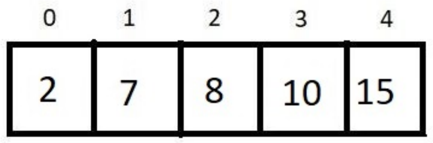
Copying all remaining elements from A to C



```
while (j <= high){
    B[k] = A[j];
    k++;
    j++;
}
for (int i = low; i <= high; i++)
    A[i] = B[i];
```

Copying all remaing elements from B to C
Copying elements back to A from B

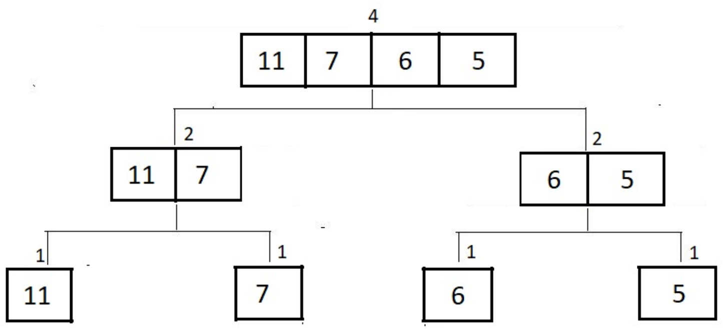
There were few changes we had to make in the pseudocode.



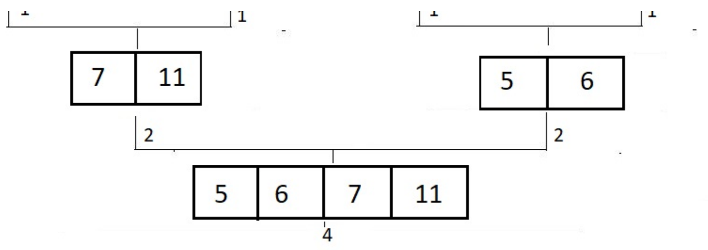
Merge sorted Array A

So that was our merging segment. That was the best I could do to make things clear to you. I hope you all understood everything I said. But things have not finished yet. This was just the merging part yet the core of the lecture. It's just the easy part left.

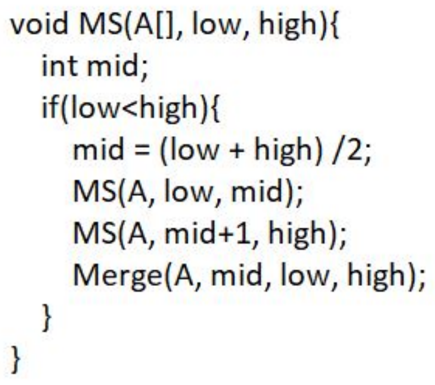
Whenever you receive an unsorted array, you break the array into fragments till the size of each subarray becomes 1. Let this be clearer via an illustration.



So, we divided the array until there are all subarrays of just length 1. Since any array/subarray of length 1 is always sorted, we just need to merge all these singly-sized subarrays into a single entity. Visit the merging procedure below.



And this is how our array got merge sorted. To achieve this divided merging and sorting, we create a recursive function merge sort and pass our array and the *low and high* index into it. This function divides the array into two parts: one from *low* to *mid* and another from *mid+1* to *high.* Then, it recursively calls itself passing these divided subarrays. And the resultant subarrays are sorted. In the next step, it just merges them. And that's it. Our array automatically gets sorted. Pseudocode for the merge sort function is illustrated below.



```
void MS(A[], low, high){
   int mid;
   if(low<high){
      mid = (low + high) /2;
      MS(A, low, mid);
      MS(A, mid+1, high);
      Merge(A, mid, low, high);
   }
}
```

We'll see the programming segment in the next lecture, and things will naturally start making more sense. You will get a better insight of the merge sort function by moving onto the next lecture. Make sure the Merge function is crystal clear to you. I have made a lot of effort above to make things easy for you. Go through the lectures again, and practice merging two sorted subarrays on your own.
I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial, where we'll learn how to code the merge sort algorithm. Till then, keep coding.

Previous Next

CodeWithHarry