**Title: Merge Sort**

**Abstract :** The strategy behind Merge Sort is to change the problem of sorting into the problem of merging two sorted sub-lists into one. If the two halves of the array were sorted, then merging them carefully could complete the sort of the entire list. Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sub-lists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.

**Methodology:** Like [QuickSort](https://www.geeksforgeeks.org/quick-sort/), Merge Sort is a [Divide and Conquer](https://www.geeksforgeeks.org/divide-and-conquer-introduction/) algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one. See following C implementation for details.

MergeSort(arr[], l, r)

If r > l

1. Find the middle point to divide the array into two halves:

middle m = (l+r)/2

2. Call mergeSort for first half:

Call mergeSort(arr, l, m)

3. Call mergeSort for second half:

Call mergeSort(arr, m+1, r)

4. Merge the two halves sorted in step 2 and 3:

Call merge(arr, l, m, r)

The following diagram from [wikipedia](http://en.wikipedia.org/wiki/File:Merge_sort_algorithm_diagram.svg" \t "_blank) shows the complete merge sort process for an example array {38, 27, 43, 3, 9, 82, 10}. If we take a closer look at the diagram, we can see that the array is recursively divided in two halves till the size becomes 1. Once the size becomes 1, the merge processes comes into action and starts merging arrays back till the complete array is merged.



**Discussion:**

* In computer science, merge sort is an efficient, general-purpose, comparison-based sorting algorithm.
* Most implementations produce a stable sort, which means that the order of equal elements is the same in the input and output.
* Merge sort is a divide and conquer algorithm that was invented by John von Neumann in 1945.
* [**Worst complexity**](https://www.google.com/search?safe=active&q=merge+sort+worst+complexity&stick=H4sIAAAAAAAAAOPgE-LQz9U3MDWIL9bSyU620s8uiM8p1y_OLyrJzEuPT8xJzy_KLMnItSrPLyouiU_Ozy3ISa3ILKlcxCqdm1qUnqoAUqoAllVAyAIA0qYH21gAAAA&sa=X&ved=2ahUKEwiIsbGGqfrhAhXe6XMBHW_HBgEQ6BMoADAnegQIDBAK)**:**n\*log(n)
* [**Average complexity**](https://www.google.com/search?safe=active&q=merge+sort+average+complexity&stick=H4sIAAAAAAAAAOPgE-LQz9U3MDWIL9bSy0620s8uiM8p1y_OLyrJzEuPT8xJzy_KLMnItUosSy1KTE-NT87PLchJrcgsqVzEKpubWpSeqgBSrACVV0DIAwDF8dy6XAAAAA&sa=X&ved=2ahUKEwiIsbGGqfrhAhXe6XMBHW_HBgEQ6BMoADAoegQIDBAN)**:**n\*log(n)
* [**Best complexity**](https://www.google.com/search?safe=active&q=merge+sort+best+complexity&stick=H4sIAAAAAAAAAOPgE-LQz9U3MDWIL9bSzk620s8uiM8p1y_OLyrJzEuPT8xJzy_KLMnItUpKLS6JT87PLchJrcgsqVzEKpWbWpSeqgBSqQCSVEBIAgBcmmtUVgAAAA&sa=X&ved=2ahUKEwiIsbGGqfrhAhXe6XMBHW_HBgEQ6BMoADApegQIDBAQ)**:**n\*log(n)
* [**Space complexity**](https://www.google.com/search?safe=active&q=merge+sort+space+complexity&stick=H4sIAAAAAAAAAOPgE-LQz9U3MDWIL9ZSyk620s8uiM8p1y_OLyrJzEuPT8xJzy_KLMnItcpNzc0vqlzEKp2bWpSeqgBSoFBckJicqpCcn1uQk1qRWVIJAKQn5-FOAAAA&sa=X&ved=2ahUKEwiIsbGGqfrhAhXe6XMBHW_HBgEQ6BMoADAqegQIDBAT)**:**n
* Merge sort runs in O (n log n) running time.
* It is very efficient sorting algorithm with near optimal number of comparison.
* [Recursive algorithm](https://www.thecrazyprogrammer.com/2014/02/how-to-convert-recursive-function-or-algorithm-to-non-recursive.html) used for merge sort comes under the category of divide and conquer technique.
* An array of n elements is split around its center producing two smaller arrays.
* After these two arrays are sorted independently, they can be merged to produce the final sorted array.
* The process of splitting and merging can be carried recursively till there is only one element in the array. An array with 1 element is always sorted.

**Source Code:**

|  |
| --- |
| #include<stdio.h>  void mergesort(int a[],int i,int j);  void merge(int a[],int i1,int j1,int i2,int j2);    int main()  {  int a[30],n,i;  printf("Enter no of elements:");  scanf("%d",&n);  printf("Enter array elements:");    for(i=0;i<n;i++)  scanf("%d",&a[i]);    mergesort(a,0,n-1);    printf("\nSorted array is :");  for(i=0;i<n;i++)  printf("%d ",a[i]);    return 0;  }  void mergesort(int a[],int i,int j)  {  int mid;    if(i<j)  {  mid=(i+j)/2;  mergesort(a,i,mid); //left recursion  mergesort(a,mid+1,j); //right recursion  merge(a,i,mid,mid+1,j); //merging of two sorted sub-arrays  }  }    void merge(int a[],int i1,int j1,int i2,int j2)  {  int temp[50]; //array used for merging  int i,j,k;  i=i1; //beginning of the first list  j=i2; //beginning of the second list  k=0;    while(i<=j1 && j<=j2) //while elements in both lists  {  if(a[i]<a[j])  temp[k++]=a[i++];  else  temp[k++]=a[j++];  }    while(i<=j1) //copy remaining elements of the first list  temp[k++]=a[i++];    while(j<=j2) //copy remaining elements of the second list  temp[k++]=a[j++];    //Transfer elements from temp[] back to a[]  for(i=i1,j=0;i<=j2;i++,j++)  a[i]=temp[j];  } |

