

# Merge Sort

- Merge Sort एक  $O(n \log n)$  तुलना आधारित छँटाई कलन विधि है.
- Merge Sort एक विभाजन और जीत कलन विधि है.
- यह 1945 में जॉन वॉन न्यूमैन द्वारा आविष्कार किया गया था.
- Bottom-up merge sort की एक विस्तृत वर्णन और विश्लेषण गोल्ड्सटाइन और न्यूमैन की एक रिपोर्ट में छपी 1948 में.

## कलन विधि

इस प्रकार के रूप धारणा, एक मर्ज सॉर्ट काम करता है :

1. N उप सूचियों में unsorted सूची में विभाजित, प्रत्येक युक्त 1 तत्व (1 तत्व की एक सूची हल माना जाता है).
2. बार बार शेष केवल 1 उप सूची जब तक वहाँ नई छाँटे गए उप सूचियों का उत्पादन करने के लिए उप सूचियों विलय. इस छांटी गई सूची हो जाएगा.

सबसे बुरी स्थिति प्रदर्शन

$O(n \log n)$

सर्वश्रेष्ठ मामले प्रदर्शन

$O(n \log n)$

औसत के मामले प्रदर्शन

$O(n \log n)$

सबसे बुरी स्थिति अंतरिक्ष जटिलता

$O(n)$  auxiliary

Source : [www.wikipedia.com](http://www.wikipedia.com)

## CODES (C)

```
#include<stdio.h>

int arr[20];           // array to be sorted

int main()
{
    int n,i;
    printf("Enter the size of array\n"); // input the elements
    scanf("%d",&n);
    printf("Enter the elements:");
    for(i=0; i<n; i++)
        scanf("%d",&arr[i]);
    merge_sort(arr,0,n-1); // sort the array
    printf("Sorted array:"); // print sorted array
    for(i=0; i<n; i++)
        printf("%d",arr[i]);
    return 0;
}

int merge_sort(int arr[],int low,int high)
{
    int mid;
    if(low<high) {
        mid=(low+high)/2;
        // Divide and Conquer
        merge_sort(arr,low,mid);
        merge_sort(arr,mid+1,high);
        // Combine
        merge(arr,low,mid,high);
    }
    return 0;
}
```

```

int merge(int arr[],int l,int m,int h)
{
    int arr1[10],arr2[10]; // Two temporary arrays to
    hold the two arrays to be merged
    int n1,n2,i,j,k;
    n1=m-l+1;
    n2=h-m;

    for(i=0; i<n1; i++)
        arr1[i]=arr[l+i];
    for(j=0; j<n2; j++)
        arr2[j]=arr[m+j+1];

    arr1[i]=9999; // To mark the end of each temporary array
    arr2[j]=9999;

    i=0;
    j=0;
    for(k=l; k<=h; k++) { //process of combining two sorted
arrays
        if(arr1[i]<=arr2[j])
            arr[k]=arr1[i++];
        else
            arr[k]=arr2[j++];
    }

    return 0;
}

```

## CODES (JAVA)

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

class MergeSort {

    static public void DoMerge(int []
numbers, int left, int mid, int right)
    {
        int [] temp = new int[25];
        int i, left_end, num_elements, tmp_pos;

        left_end = (mid - 1);
        tmp_pos = left;
        num_elements = (right - left + 1);

        while ((left <= left_end) && (mid <= right))
        {
            if (numbers[left] <= numbers[mid])
                temp[tmp_pos++] = numbers[left++];
            else
                temp[tmp_pos++] = numbers[mid++];
        }

        while (left <= left_end)
            temp[tmp_pos++] = numbers[left++];

        while (mid <= right)
            temp[tmp_pos++] = numbers[mid++];

        for (i = 0; i < num_elements; i++)
        {
            numbers[right] = temp[right];
            right--;
        }
    }
}
```

```

    static public void MergeSort_Recursive(int []
numbers, int left, intright)
    {
        int mid;

        if (right > left)
        {
            mid = (right + left) / 2;
            MergeSort_Recursive(numbers, left, mid);
            MergeSort_Recursive(numbers, (mid + 1), right);

            DoMerge(numbers, left, (mid+1), right);
        }
    }

public static void main(String[] args)
{
    int[] numbers = { 47, 18, 23, 19, 1, 72, 87, 44, 21 };
    int len = 9;

    System.out.println("Output:");

    MergeSort_Recursive(numbers, 0, len - 1);
    for (int i = 0; i < 9; i++)
        System.out.println(numbers[i]);
}
}

```