# GeeksQuiz

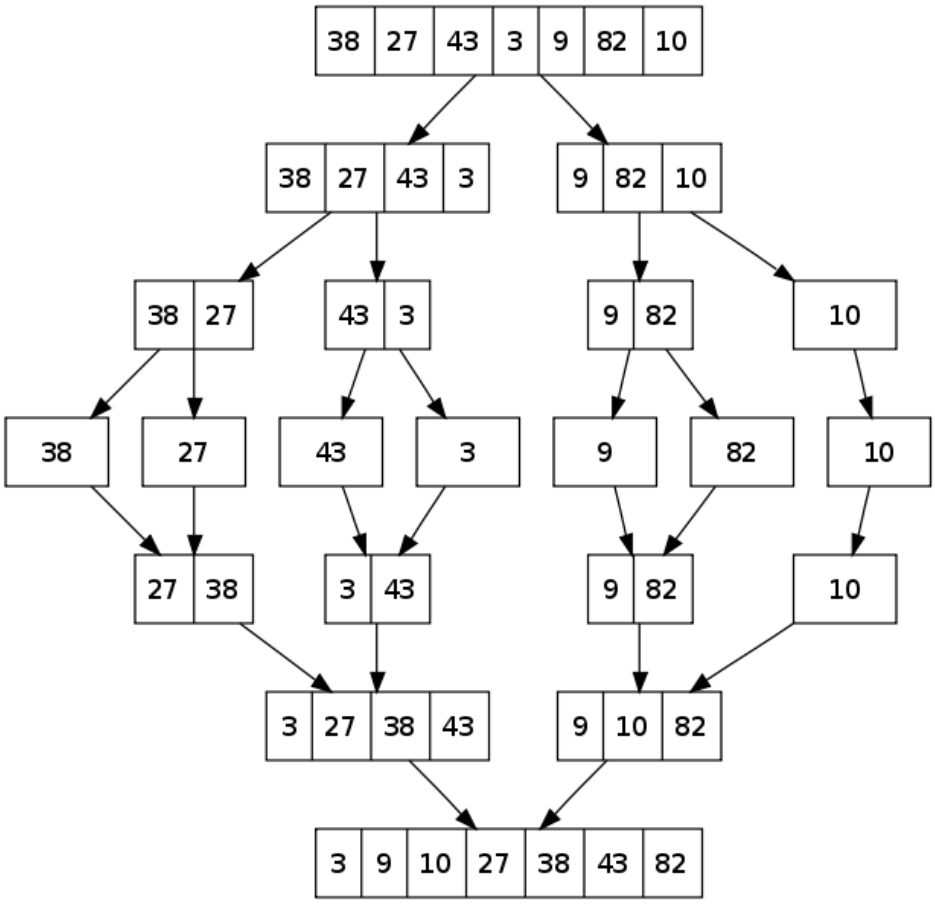
Computer science mock tests for geeks

# Merge Sort

MergeSort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. **The merg() 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 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.




```c
/* C program for merge sort */
#include<stdlib.h>
#include<stdio.h>

/* Function to merge the two haves arr[l..m] and arr[m+1..r] of array arr[] */
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for(i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for(j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
```

```
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there are any */
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there are any */
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the sub-array
  of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = l+(r-l)/2; //Same as (l+r)/2, but avoids overflow for large l and h
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);
        merge(arr, l, m, r);
    }
}


/* UITLITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

/* Driver program to test above functions */
int main()
{
    int arr[] = {12, 11, 13, 5, 6, 7};
    int arr_size = sizeof(arr)/sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

Output:

```
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```

**Time Complexity:** Sorting arrays on different machines. Merge Sort is a recursive algorithm and time complexity can be expressed as following recurrence relation.
T(n) = 2T(n/2) + $\Theta(n)$
The above recurrence can be solved either using Recurrence Tree method or Master method. It falls in case II of Master Method and solution of the recurrence is $\Theta(nLogn)$.
Time complexity of Merge Sort is $\Theta(nLogn)$ in all 3 cases (worst, average and best) as merge sort always divides the array in two halves and take linear time to merge two halves.

**Auxiliary Space:** O(n)

**Algorithmic Paradigm:** Divide and Conquer

**Sorting In Place:** No in a typical implementation

**Stable:** Yes

**Applications of Merge Sort**
**1)** Merge Sort is useful for sorting linked lists in O(nLogn) time. Other nlogn algorithms like Heap Sort, Quick Sort (average case nLogn) cannot be applied to linked lists.
**2)** Inversion Count Problem
**3)** Used in External Sorting

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Category: Searching and Sorting

**Archita** • 2 years ago

Can yoi please give me the explanation of why in Merge sort we divide the list into 2 parts only ??? Why not 3 or 4 ?

7 • Reply • Share ›

**geeksquizgeeksquiz** Mod ➚ Archita • 2 years ago

If we divide in more parts, we may end up with more merge operations. See http://stackoverflow.com/quest...

4 • Reply • Share ›

**jimmy** • a year ago

i think ,in the merge() function the second array
for(j = 0; j <= n2; j++)
R[j] = arr[m + 1+ j];
j should not be equal to n2.

what do you think?

4 • Reply • Share ›

**GeeksforGeeks** Mod ➚ jimmy • a year ago

Thanks for pointing this out. We have updated the code.

• Reply • Share ›

**Faisal** ➚ GeeksforGeeks • a year ago

The new code won't compile because you cannot allocate arrays on the stack without knowing their length before hand.

• Reply • Share ›

**GeeksforGeeks** Mod ➚ Faisal • 10 months ago

This is allowed in all C compilers except few legacy compilers like Turbo C. For example, try the new code with gcc.

• Reply • Share ›

**Techie1991** • 5 months ago

Why can't be this done inplace? Why need the two auxiliary arrays?

2 • Reply • Share ›

**Jenny_P** • 9 months ago

I'm trying to implement this code but am getting compile errors... Could someone please tell me how to avoid a error C2057: expected constant expression when declaring a temp array?

(Is Visual Studio 2010 a 'Legacy' compiler?)

2 • Reply • Share ›

**Ravi Prakash Giri** • a year ago

What is the need of j<=n2 for copying data into array R[ ]? Cann't it be done by simply j

1 • Reply • Share ›

**Vinay A.F** • 5 days ago

the expression int m = l+(r-1)/2 in method mergeSort() should be changed to m = (l+(r-1))/2

• Reply • Share ›

**Sivakumar Sundaramurthi** • a month ago

Hello Friends.....I hav a doubt, I need to use merge sort for sorting the elements in odd positions of an array. What will be the time complexity??....if it remains O(nlogn).....Wat is the diff b/w sorting the Whole array and the sorting the elements in odd positions of an array???.....Please Help me...:)

• Reply • Share ›

**Govinda raj** • 2 months ago

Can we have other algorithm for comparison sort which take less than O(nlogn) time using divide and conquer?

• Reply • Share ›

**RICKY KUMAR** • 3 months ago

for array int arr[]={2,1,4,51,34,52,2,3,1};

this is not working

• Reply • Share ›

**MythBuster** • 4 months ago

int m = l+(r-l)/2;
mergeSort(arr, l, m);

For n = 6, l = 3, r = 5, the supposedly overflow-safe expression, l + (r - 1) / 2 yields the same m always and hence the same r(and also l remains unchanged) in the subsequent mergesort() call, and ends up in smashing the stack due to infinite recursion. Someone

needs to look into this. Replacing l + (r - 1) / 2 with the usual (l + r) / 2 fixed the recursion.

• Reply • Share ›

**dk** • 5 months ago

change into /* UTILITY FUNCTIONS */ from /* UITLITY FUNCTIONS */

• Reply • Share ›

**pkzz** • 7 months ago

First the list is divided in 2 parts until 2 elements are left.....can someone explain after merging (11,12) in sorted order how does the merge sort function returns (13,5) for next iteration

• Reply • Share ›

**Guest** • 7 months ago

```
4 void merge( int pArray[],

5 int array_one[],int lone,

6 int array_two[],int ltwo)

7 {

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

9 int tempOne = array_one[i], tempTwo = array_two[j];

10 int length = lone+ltwo;

11 printf("length=%d\n",length);

12 while(i < lone && j < ltwo)

13 {

14 if(array_one[i] < array_two[i])
```

see more

• Reply • Share ›

**Guest** • 7 months ago

Improved merge function:

```
3 void merge( int pArray[],

4 int array_one[],int lone,

5 int array_two[],int ltwo)
```

```
6 {

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

8 int tempOne = array_one[i], tempTwo = array_two[j];

9 int length = lone+ltwo;

10 printf("length=%d\n",length);

11 while(iCount < length)

12 {
```

see more

• Reply • Share ›

**Pratik** • 7 months ago

Why does this statement work.
/* create temp arrays */
int L[n1], R[n2];
& and this doesnot work..
scanf("%d",n);
int a[n]

I mean if we are able to dynamically create array why do we need malloc

• Reply • Share ›

**Ayush Jain** • 10 months ago

**@GeeksforGeeks** For worst case of merge sort, refer this link :
http://stackoverflow.com/a/245...

• Reply • Share ›

**Rishabh Gupta** Ayush Jain • 10 months ago

Also refer this link also

http://stackoverflow.com/users...

• Reply • Share ›

**Jibran** • 10 months ago

How do you arrive at the recurance relation ?

• Reply • Share ›

**Shishir** • 10 months ago

This code doesn't seem to work. I have used the exact same code but its not working. You can check the code here https://gist.github.com/Shad0w...

This is output of the program:

Length is 6
Original array
12 11 13 5 6 7
Sorted array
12 12 6 7 13 6 7

• Reply • Share ›

**lambda** • 10 months ago

merge L1 and L2 can be concisely.

```
while(k <= end){
if(j > n2-1 || (i < n1 && L1[i] < L2[j])){
arr[k] = L1[i];
i++;
}
else{
arr[k] = L2[j];
j++;
}
k++;
}
```

• Reply • Share ›

**Manav** • a year ago

T(n) = 2T(n/2) + . It falls in case 2 as c is 1 and is also 1. So the solution is how is ans nlogn can anyone explain please

• Reply • Share ›

**GeeksforGeeks** Mod Manav • a year ago

Please see http://www.geeksforgeeks.org/a... for more details of solving recurrences using Master Method.

• Reply • Share ›

**amitav shaw** • a year ago

it works with j< n2 also.. Can you explain why its j <=n2 in the for loop for copying elements in R. What I think is it unnecessarily creates an extra element in the array, but since the code doesn't hit that part.. its harmless. There could be a possible bug or may be I'm wrong.

• Reply • Share ›

**sameer** • a year ago

can we create an array with some variable passed in it..!!

i mean not the constant.
like int L[n1]..??
0_o

Reply • Share ›

**Rajesh PS** ➚ sameer • 4 months ago
no we cant check below

Reply • Share ›

**Vin** • a year ago
You have to free R and L in the end of merge function

Reply • Share ›

**GeeksforGeeks** Mod ➚ Vin • a year ago
Thanks for pointing this out. We have changed the code to use auto arrays instead of dynamically allocated arrays.

Reply • Share ›

**Trish** • 2 years ago
In the function "void mergesort" , we can define m= l+(h-l)/2 instead of m=l+h/2 because if l and r get last two indexes and that too of very high values,then we might have an overflow.

Reply • Share ›

**geeksquizgeeksquiz** Mod ➚ Trish • 2 years ago
Trish: Thanks for inputs. We have updated the post. Keep it up!

Reply • Share ›

**Mohamed Fasil** ➚ geeksquizgeeksquiz • 2 years ago
updation has compilation error for h variable... it shd be r and not h

Reply • Share ›

**GeeksforGeeks** Mod ➚ Mohamed Fasil • 2 years ago
Thanks for pointing this out. We have changed h to r.

Reply • Share ›

**Rajesh PS** ➚ GeeksforGeeks • 4 months ago
//pls.check this error.. i ve corrected BELOW the code ..because it won't work in visual c++ compiler

```
void merge(int arr[], int l, int m, int r)

{

int i, j, k, n1, n2, L[10], R[10];
```

```
int i, j, k,n1,n2,L[10],R[10];

n1 = m - l + 1;

n2 = r - m;

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

L[i] = arr[l + i];

for(j = 0; j < n2; j++)

R[j] = arr[m + 1+ j];
```

see more

• Reply • Share ›

**rajesh** GeeksforGeeks • 4 months ago

please check this line:int L[n1], R[n2];
it show error.....where it
cannot allocate an array of constant size 0
expected constant expression

• Reply • Share ›