Merge 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 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.



**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) + Θ (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 Θ (n log n).  
Time complexity of Merge Sort is Θ (n log n) in all 3 cases (worst, average and best) as merge sort always divides the array into two halves and take linear time to merge two halves.

**Algorithm**:

1. Algorithm MergeSort(low,high)

2. //a[low:high] is a global array to be sorted

3. //Small(P) is true if there is only one element

4. //to sort. In this case the list is already sorted.

5. {

6. if (low<high) then //if there are more than one element

7. {

8. //Divide P into subproblems

9. //find where to split the set

10. mid = [(low+high)/2];

11. //solve the subproblems.

12. mergesort (low,mid);

13. mergesort(mid+1,high);

14. //combine the solutions .

15. merge(low,mid,high);

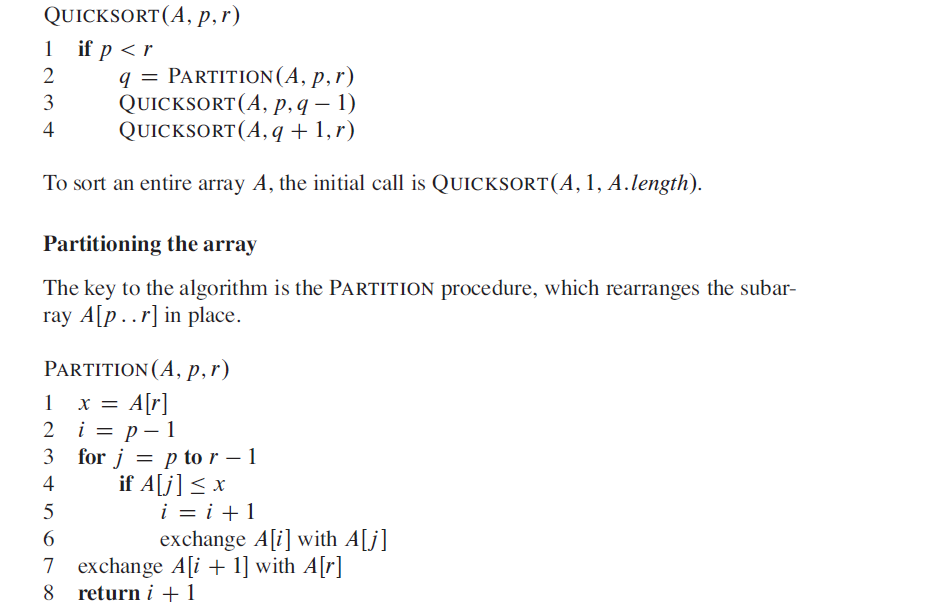
16. }

17. }

**QUICKSORT**

Quicksort is a sorting algorithm whose worst-case running time is http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/bound.gif(*n*2) on an input array of *n* numbers. In spite of this slow worst-case running time, quicksort is often the best practical choice for sorting because it is remarkably efficient on the average: its expected running time is http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/bound.gif(*n* lg *n*), and the constant factors hidden in the http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/bound.gif(*n* lg *n*) notation are quite small. It also has the advantage of sorting in place (see page 3), and it works well even in virtual memory environments.

The following procedure implements quicksort.



**Selection sort**

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of Ο(n2), where **n** is the number of items.

## **How Selection Sort Works?**

Consider the following depicted array as an example.

Unsorted Array

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

Selection Sort

So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of the sorted list.

Selection Sort

For the second position, where 33 is residing, we start scanning the rest of the list in a linear manner.

Selection Sort

We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.

Selection Sort

After two iterations, two least values are positioned at the beginning in a sorted manner.

Selection Sort

The same process is applied to the rest of the items in the array.

Following is a pictorial depiction of the entire sorting process −



Now, let us learn some programming aspects of selection sort.

### **Algorithm**

**Step 1** − Set MIN to location 0

**Step 2** − Search the minimum element in the list

**Step 3** − Swap with value at location MIN

**Step 4** − Increment MIN to point to next element

**Step 5** − Repeat until list is sorted

### **Pseudocode**

procedure selection sort

list : array of items

n : size of list

for i = 1 to n - 1

/\* set current element as minimum\*/

min = i

/\* check the element to be minimum \*/

for j = i+1 to n

if list[j] < list[min] then

min = j;

end if

end for

/\* swap the minimum element with the current element\*/

if indexMin != i then

swap list[min] and list[i]

end if

end for

end procedure

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.geeksforgeeks.org/merge-sort/>
2. <http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap08.htm>
3. <https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm>

**Lecture Video:**

1. <https://youtu.be/6pV2IF0fgKY>
2. <https://youtu.be/7h1s2SojIRw>
3. <https://youtu.be/uEUXGcc2VXM>
4. https://youtu.be/xWBP4lzkoyM

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**