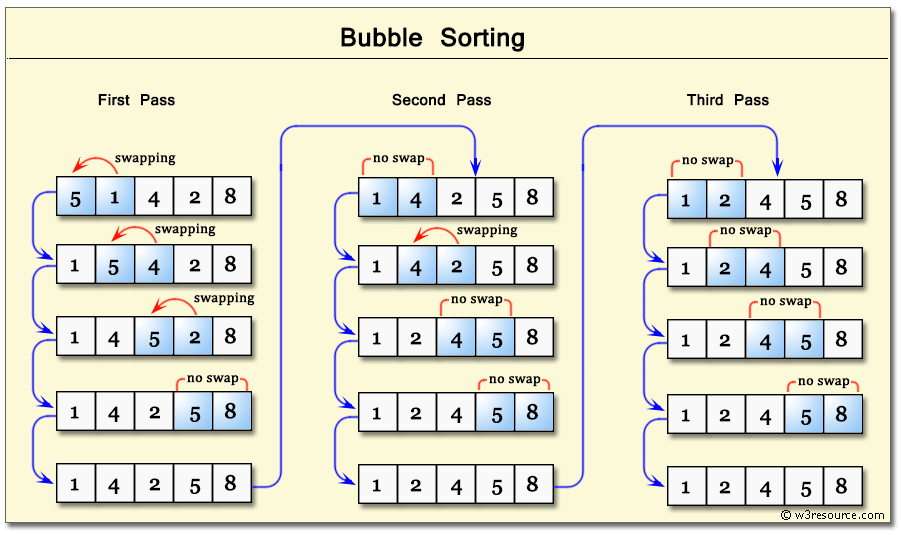
**Sorting Algorithms:**

We have implemented **Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort, Heap Sort, Radix Sort, Shell Sort.**

# **Bubble Sort**

# Bubble sort is a simple sorting algorithm in which each element is compared with adjacent element and swapped if their position is incorrect. It is named as bubble sort because same as like bubbles the lighter elements come up and heavier elements settle down.

Both worst case and average case complexity is O (n2).



**Insertion Sort**

**Insertion sort** is a sorting algorithm that builds a final sorted array one element at a time.

Insertion sort has an average and worst-case running time of O (n2).



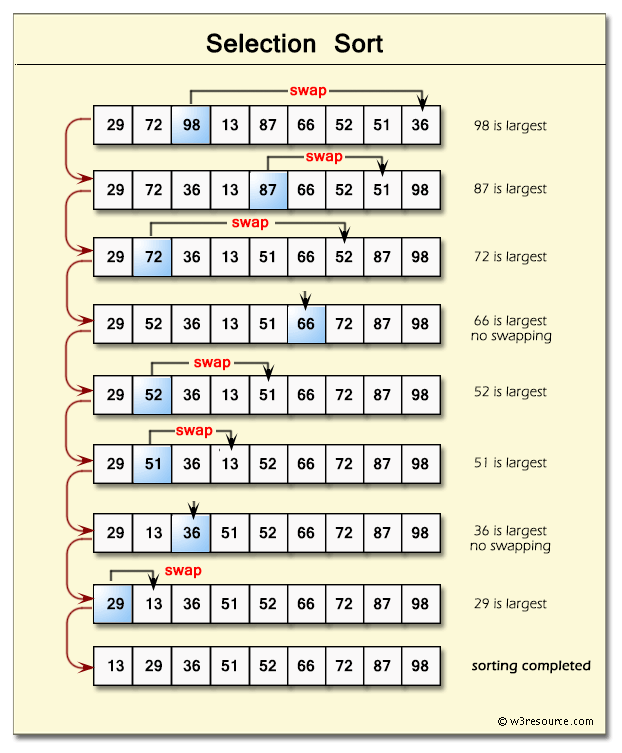
**Selection Sort**

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.  
2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

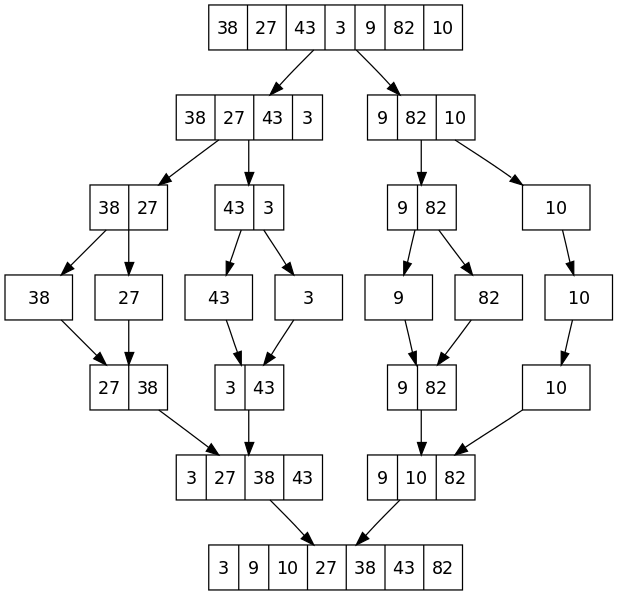
Time Complexity is same for Best Case and Worst Case = O(n^2)



**Merge Sort**

In Divide & Conquer algorithm design paradigm, we divide the problems in sub-problems recursively then solve the sub-problems, & at last combine the solutions to find the final result.  
One thing to keep in mind while dividing the problems into sub-problems is that, the structure of sub-problems should not change as of the original problem.  
Divide & Conquer algorithm has 3 steps:  
1. Divide: Breaking the problem into subproblems  
2. Conquer: Recursively solving the subproblems  
3. Combine: Combining the solutions to get the final result.

Time Complexity for Best Case and Worst Case = O(n log(n))



**Quick Sort**

Quicksort is a divide and conquer algorithm.

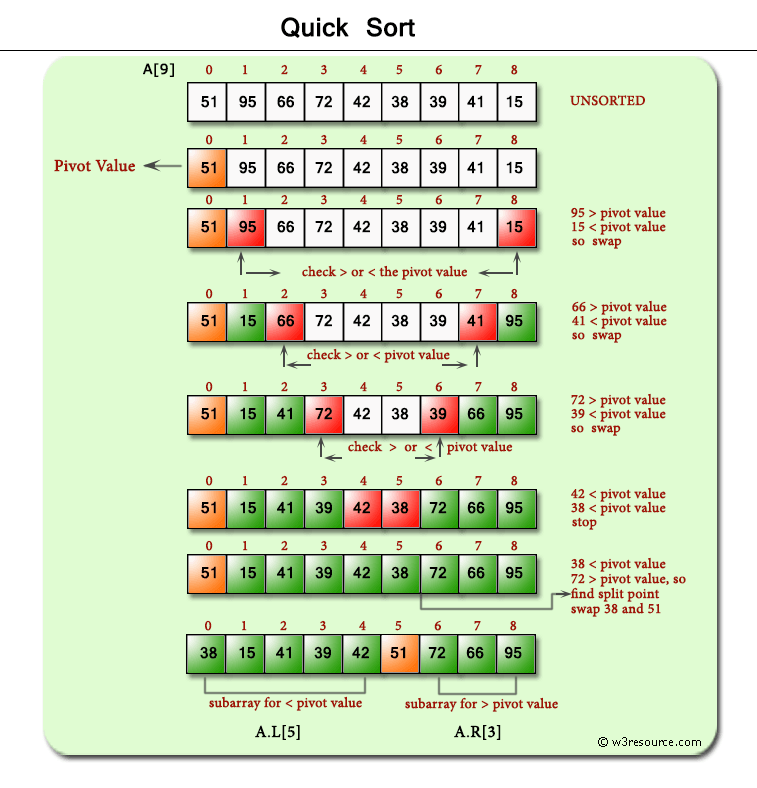
The steps are:

1) Pick an element from the array, this element is called as pivot element.

2) Divide the unsorted array of elements in two arrays with values less than the pivot come in the first sub array, while all elements with values greater than the pivot come in the second sub-array (equal values can go either way). This step is called the partition operation.

3) Recursively repeat the step 2(until the sub-arrays are sorted) to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values. The same logic we have implemented in the following C program.

Time Complexity of Quick sort in best case = Ω (n log(n)), Worst Case = O(n^2)



**Heap Sort**

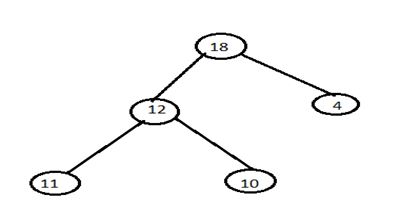
## What is Heap?

Heap is a complete binary tree in which every parent node be either greater or lesser than its child nodes. Heap can be of two types that are:

**1) Max heap**

A max heap is a tree in which value of each node is greater than or equal to the value of its children node.

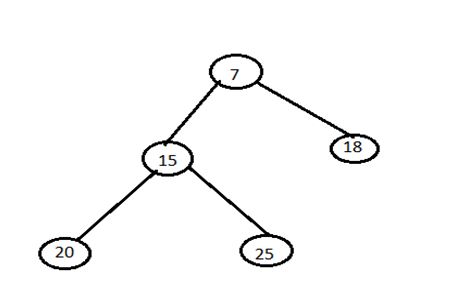
**Example of max heap:**



**2) Min heap**

A min heap is a tree in which value of each node is less than or equal to the value of its children node.

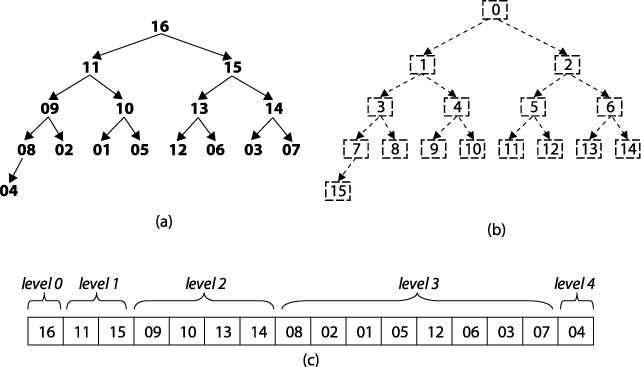
**Example of min heap:**



### **Working of Heap Sort**

1. The heap sort begins by building a heap out of the data set and then removing the largest item and placing it at the end of the sorted array.
2. After removing the largest item, it reconstructs the heap and removes the largest remaining item and places it in the next open position from the open position from the end of the sorted array.
3. This is repeated until there is no item left in the heap and the sorted array is full.
4. Elementary implementation requires two arrays one to hold the heap and the other to hold the sorted element.

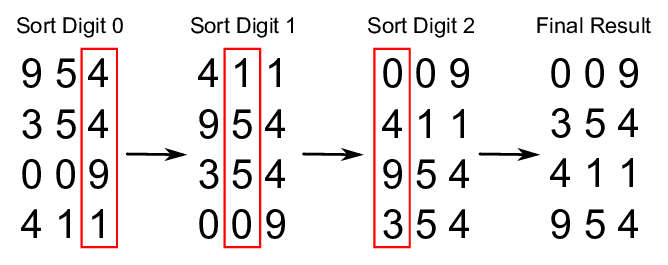
Time Complexity for Best Case and Worst Case = O (n log(n))



**Radix Sort**

**Radix sort** is an integer sorting algorithm that sorts data with integer keys by grouping the keys by individual digits that share the same significant position and value (place value). Radix sort uses counting sort as a subroutine to sort an array of numbers. Because integers can be used to represent strings (by hashing the strings to integers), radix sort works on data types other than just integers.

Time Complexity for Best Case and Worst case = O (nk)

****

**Shell Sort**

Shell sort is a highly efficient sorting algorithm and is based on insertion sort algorithm. This algorithm avoids large shifts as in case of insertion sort, if the smaller value is to the far right and has to be moved to the far left.

Time Complexity for Best Case =O (n\* logn) and Worst case= O (n\* log2n).



**Contribute**

If you have a Sorting Technique that is not in repo feel free to contribute.

**Anything else that seems useful.**