Sorting Techniques

1. Selection Sort;

5,1,4,2,8

ASC (Greedy)

Sorting the number by its index is 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.

Code:

minIndex;

for(curr=0;curr<n-1;curr++)

{

for(i=curr+1;i<n;i++)

{

if(a[minIndex]>a[i])

{

minIndex=I;

}

}

swap(a[curr],a[minIndex]);

}

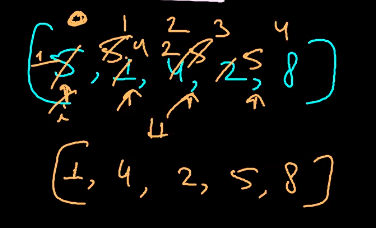
Time Complexity = O(n)

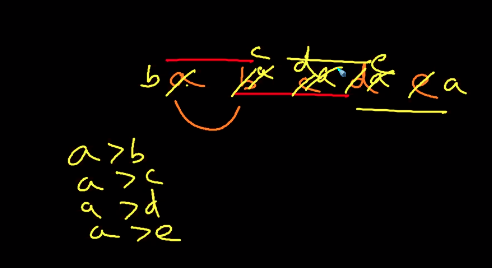
1. Bubble sort:

5,1,4,2,8

Doing Adjacent comparison

Swapping the adjacent element checking its size.



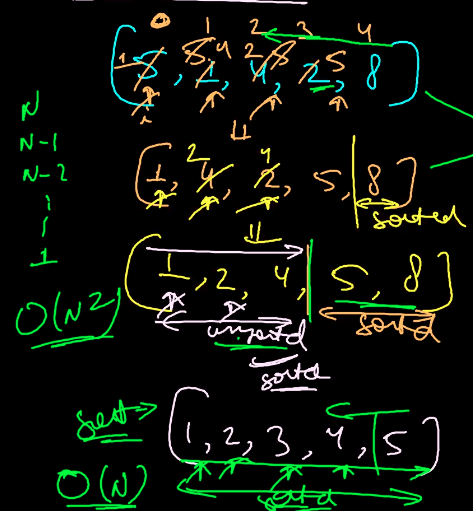


Initially 5 and 1 is compared and swapped and 5 and 4 is compare and swapped again 5 and 2 is compared and swapped. Again 5 and 8 is compared and will be swapped is the 5 is greater than that number. It will go till the last element of the array.

Finally, after adjusting the element of the array we get the sorted array, 1,2,3,5,8 as the result.

Time Complexity: O(n) for best case

O(n^2) for worst case of bubble sort.



Code:

for(i=0;i<n-1;i++){

for(j=0;j<(n-i-1);j++){

if(a[j]>a[j+1]){

swap(arr[j],arr[j+1]);

}

}

}

1. Merge Sort:

If N=7;

We need to divide and conquer.

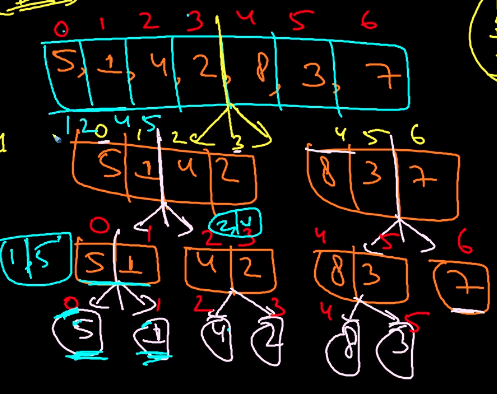
Rules needs to be followed.

1. Divide array in 2 parts.
2. Merge 2 sorted array.

We need to declare 3 variable to do that.

low,mid and high.

Time Complexity: O(n)



**Code;**

Void ms(vector<int>& a, int low, int high)

If(low>high)

Return ;

Mid=low+((high-low)/2);

Ms(a,low,mid);

Ms(a,mid+1,high);

Merge2SoertedList(a,low,mid,high);

}

Time Complexity: O(N log N)

How to Sort more efficiently; (insertion sort)

Lets take a number I and compare it to the numbers of the array.

Take a number as reference and compare it with other number. If x<i<y in this order.

Example;

5,1,4,2,8,3,7

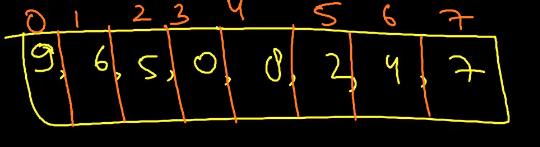
5 is taken into an array and 1 is compared with 5, 1<5 so 1 is placed before 5;

Set the index to the last value or size of the array.

Again 4 is compared with the 5, 4<5 so 4 is inserted to the left of 5 and again 4 is compared with 1, 1<4 so 4 is placed to the right of the 1. It goes till the end. We get the sorted elements at the last.

**Quick Sort:**

Let us assume an array of element



In quick sort we need to select a pivot:

How to select the pivot element in the array -> Pick the last element as the pivot.

It is like “<=(left) pivot <= (right) ”

