Todays Content

1. Sorting Intro
2. Bubble Sort
3. Selection Sort
4. Insertion Sort

Sorting: Arranging data in inc/dec order based on parameter

	0	1	2	3	4	5	
Ex1: A[] = {	2	3	9	12	17	19	} : Inc based on value
Ex2: A[] = {	19	6	5	2	-1	-19	} : Dec based on value
Ex3: A[] = {	2	7	4	9	6	10	} : Inc based on factors
#factors	2	2	3	3	4	4	

Sorting library? TC: $O(N \log_2 N)$ To sort N elements.

1. To sort int arr[]:

```
sort(arr, arr+n); # It will sort arr[] from arr[0] to arr[N-1] in Inc
sort(arr, arr+n, greater<int>()); # It will sort arr[] in Dec
# include <function>  # for greater
```

2. To sort vector<int> v;

```
sort( v.begin(), v.end()); #Sort in increasing order
sort(v.begin(), v.end(), greater<int>()); # Sort in Decreasing Order
# include <function>  # for greater
```

Stable Sorting: When 2 data points have same value, their relative order should be same, before & after sorting, that type of sorting algo is called stable sorting Algo.

fn: ar[5] = { 1 5 2 6 2 } #Sort inc based on value;

#Sort1 = { 1 2 2 5 6 } #Sort1 is stable

#Sort2 = { 1 2 2 5 6 } #Sort2 is unstable

Inplace Sorting:

Sorting Algo with SC: O(1) It is considered as Inplace Sorting.

All Sorting Algorithms

#Basic:	Selection Sort	& Bubble Sort	& Insertion Sort
TC :	$O(N^2)$	$O(N^2)$	$O(N^2)$
Stable Sort :	*	✓	✓
Inplace Sort :	✓	✓	✓

#Optimized:	Merge Sort	Quick Sort	Heap Sort
TC :			
Stable Sort :			
Inplace Sort :			

#Linear:	Count Sort	Radix Sort
TC :		
Stable Sort :		
Inplace Sort :		

Selection Sort:

At every step:

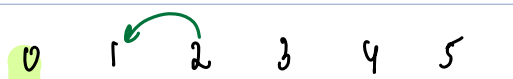
Iterate on arr[] select min & bring to it's correct position.

	0	1	2	3	4	5
arr[] = {	9	8	4	10	6	2 }

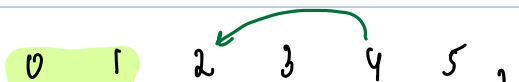
mini: Index with min value

MinInd Swap(arr[] arr[]);

	0	1	2	3	4	5		
i=0 j:[0 5] {	9	8	4	10	6	2 }		
mini=	0	1	2	2	2	5	5	swap(arr[0] arr[5])

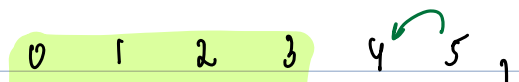


	0	1	2	3	4	5		
i=1 j:[1 5] {	2	8	4	10	6	9 }		
mini=		1	2	2	2	2	2	swap(arr[1] arr[2])

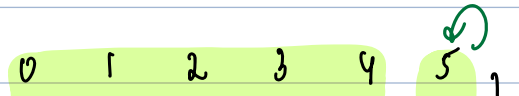


	0	1	2	3	4	5		
i=2 j:[2 5] {	2	4	8	10	6	9 }		
mini=			2	2	4	4	4	swap(arr[2] arr[4])

	0	1	2	3	4	5		
i=3 j:[3 5] {	2	4	6	10	8	9 }		
mini=				3	4	4	4	swap(arr[3] arr[4])



	0	1	2	3	4	5		
i=4 j:[4 5] {	2	4	6	8	10	9 }		
mini=					4	5	5	swap(arr[4] arr[5])



	0	1	2	3	4	5		
i=5 j:[5 5] {	2	4	6	8	9	10 }		
mini=						5	5	swap(arr[5] arr[5])

Pseudo Code:

```
i : 0 to N-1 :
    Iterate j : i to N-1 & calculate mini :
    swap arr[mini] & arr[i];
```

```
void selectim(int[] ar, int N){ TC: O(N^2   SC: O(1)
    for(int i=0; i<N; i++){
        # i: j: [i..N-1] & calculate mini;
        int mini = i;
        for(int j=i; j<N; j++){
            # ar[mini] && ar[j]
            if(ar[j] < ar[mini]){
                mini = j;
            }
        }
        swap ar[i] & ar[mini]
    }
}
```

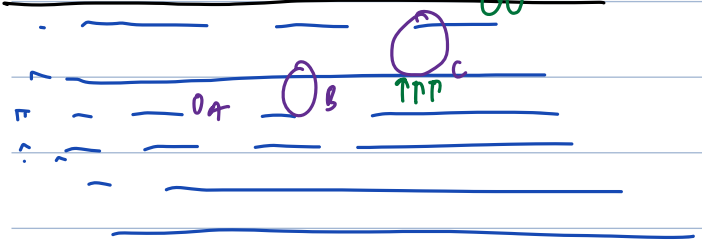
#Selectim Sort:

	0	1	2	3	4	5	MinInd	Swap(ar[] ar[]);
i=0 j:[0 5] {	6	5	6	3	8	7 }		
mini =	0	1	1	3	3	3	3	swap(ar[0] ar[3])

	0	1	2	3	4	5		
i=1 j:[1 5] {	3	5	6	6	8	7 }		
mini =		1	1	1	1	1	1	swap(ar[1] ar[1])

	0	1	2	3	4	5	
i= j:[ ] {	3	5	6	6	8	7 }	# Issue: Relative order between 6 6 is changed
mini =							

#Idea Bubble Sort: Bigger bubble will come out on top.



Bubble Sort : At each iteration

Compare adj elements, if not in correct order swap

Increasing

Note:

	0	1	2	3	4	5	
ar[] :	9	8	4	10	6	2	#small optim
i=0:	8	4	9	6	2	10	i=0, j<5; j==5 stop
i=1:	4	8	6	2	9	10	i=1, j<4, j<5-1;
i=2:	4	6	2	8	9	10	i=2, j<3; j<5-2;
i=3:	4	2	6	8	9	10	i=3 j<2; j<5-3;
i=4:	2	4	6	8	9	10	i → j < N-i-1;
i=5:	2	4	6	8	9	10	

```
void bubbleSort(int[] ar, int N){ TC: O(N^2)  SC: O(1)
    for(int i=0; i<N; i++){

        for(int j=0; j<N-i-1; j++){
            if(ar[j] > ar[j+1]){ # j = N-1 : ar[N-1] > ar[N] : Error.
                swap ar[j] & ar[j+1];
            }
        }
    }
```

#Stability : # Relative order maintained in small elements

ar[] = { 6 3 5 6 4 }

i=0 { 3 5 6 4 6 }

i=1 { 3 5 4 6 6 }

i=2 { 3 4 5 6 6 }

3

TODO: Check inplace or not.

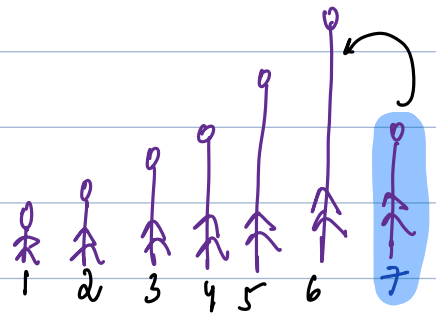
Insertion Srt:

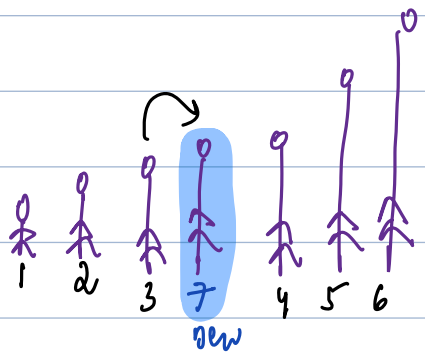
We insert 1 element in enisting sorted data to make entire data sorted.

while (prev persn > new persn && if no prev ele we stop) {

We swap

If new persn at start position, no prev persn so we can stop





Insertion Sort:

Dry Run:

	Sort	Insert		0	1	2	3	4	5
i=	[ ]	ar[ ]	ar[ ] = {	10	9	4	8	6	2 }
i=	[ ]	ar[ ]							
i=	[ ]	ar[ ]							
i=	[ ]	ar[ ]							
i=	[ ]	ar[ ]							
i=	[ ]								

```
void InsertSort(int[] ar, int N){   TC: O(  )  SC: O( )
    for(int i=0; i<N-1; i++){
        # i = [0..i] is sorted      [0 1 2 .. i] | i+1
        int j=i+1; // Insert in sorted data [0...i];
        while( j!=0 && ar[j-1]>ar[j]){   # j=0 {ar[-1]>ar[0]}
            swap ar[j-1] & ar[j]
            j--;
        }
    }
}
```