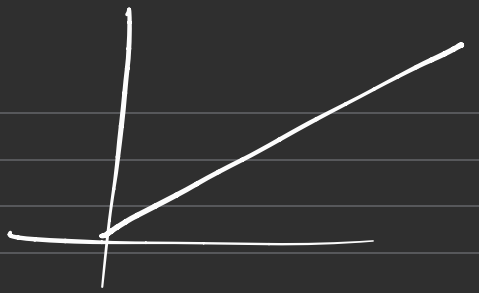
"रास्ते खुद बन जाते हैं,
हौसलों के कदम जब उठते हैं।"

Sorting

Sorting in algorithms means arranging elements in a specific order, usually either ascending (smallest to largest) or descending (largest to smallest).

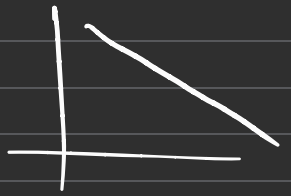
Increasing

1 2 3 4 5



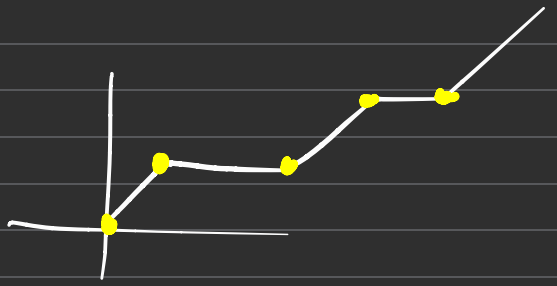
Decreasing

5 4 3 2 1



Ascending

1 1 2 3 3 4



Descending

4 3 3 2 1 1

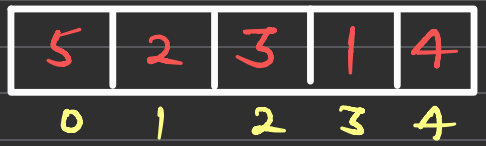
Stable Sorting

2A 2B

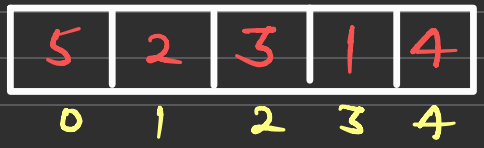
1B 1A

1B 1A 2A 2B 3 4 5

Inplace Sorting



Bubble Sort



- Inplace Algorithm
- Stable Algorithm (Not proofed)
- It mainly compare two adj element & if they are in correct pos. then swap it.

5 2 = 2 5

N = 5

Pass # 0

5 2 3 1 4

2 5 3 1 4

2 3 5 1 4

2 3 1 5 4

2 3 1 4 5

Pass # 1

2 3 1 4 | 5

2 3 1 4 5

2 1 3 4 5

2 1 3 4 5

Pass # 2

2 1 3 | 4 5

1 2 3 4 5

1 2 3 4 5

Pass # 3

1 2 | 3 4 5

1 2 3 4 5

N = 5

1 Rem.

4 elements correct position

Size = N

No. of passes = (N-1)

No. of comparision = (N - pass - 1)

N = 5

N = 4

(0 1 2 3)

Total Pass.	Comparision	N (5)
Pass 0 =	4	5 - 0 - 1 = 4
Pass 1 =	3	5 - 1 - 1 = 3
Pass 2 =	2	5 - 2 - 1 = 2
Pass 3 =	1	5 - 3 - 1 = 1

```
void bubbleSort(arr, n)
{
    for (pass = 0 ; pass < N-1 ; pass++)
    {
        for (j = 0 ; j <= N-pass-1 ; j++)
        {
            if (a[j] > a[j+1])
            {
                swap (a[j], a[j+1])
            }
        }
    }
}
```

N

pass = (N-1)

(N - pass - 1)

N x (N-pass-1)

N^2 - N x pass - N

N^2

Alternative Sorting

Difficulty: **Basic** Accuracy: **50.2%** Submissions: **47K+** Points: **1**

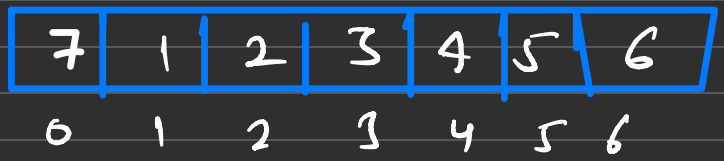
Given an array **arr** of **distinct** integers. Rearrange the array in such a way that the first element is the largest and the second element is the smallest, the third element is the second largest and the fourth element is the second smallest, and so on.

Examples:

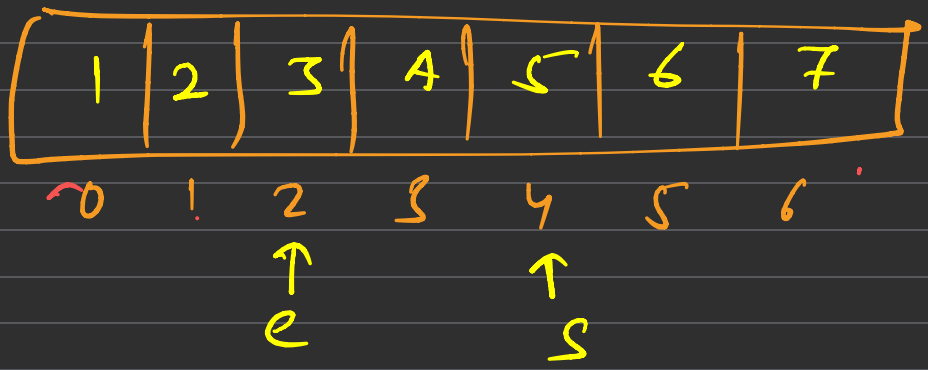
Input: arr[] = [7, 1, 2, 3, 4, 5, 6]
Output: [7, 1, 6, 2, 5, 3, 4]
Explanation: The first element is first maximum and second element is first minimum and so on.

e s e s

FL FS SL SS TL TS



i|p



7 1 6 2 5 3 4 4

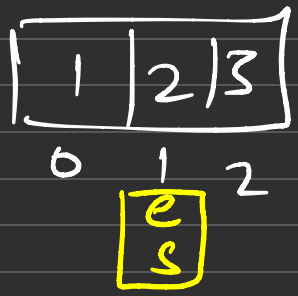
sort (arr)

i/p 3 1 2

o/p = 3 1 2

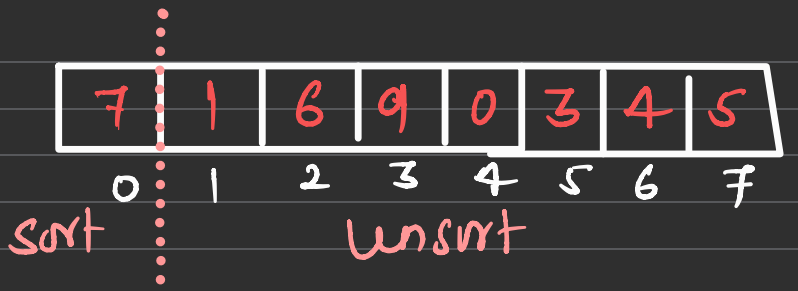
s = 0, e = N-1

```
while (s <= e && s != e)
{
    if (s == e)
    {
    }
    else
    {
        cout << a[e]
        cout << a[s]
        e--
        s++
    }
```

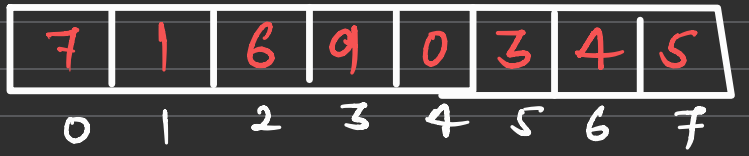


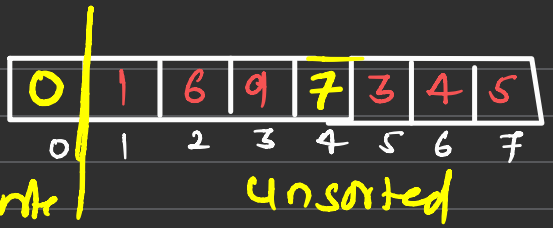
3 1

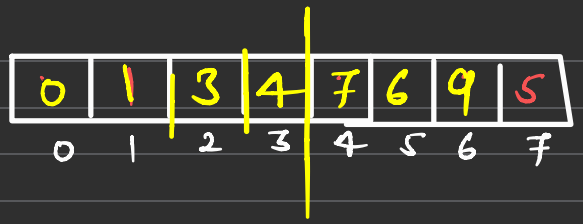
Selection Sort

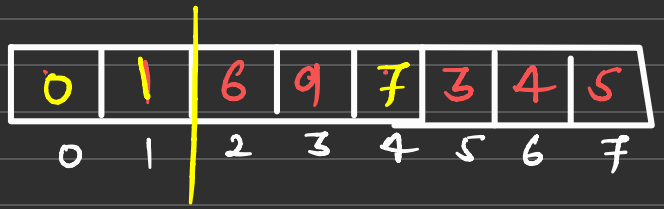


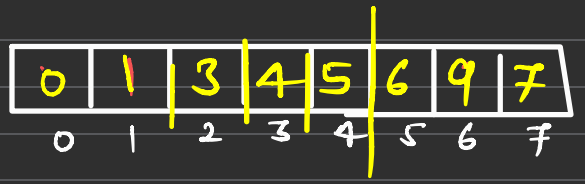
Bring Smallest No. from Unsorted section

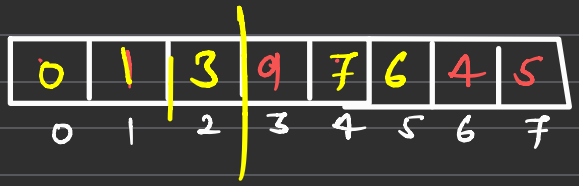


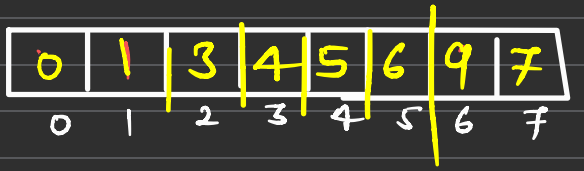






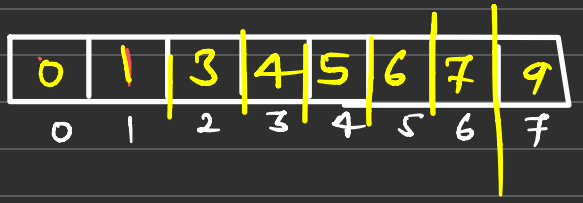


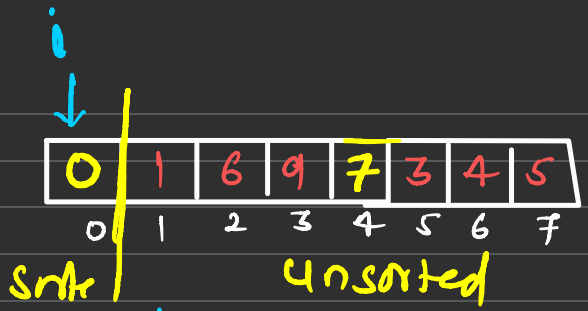


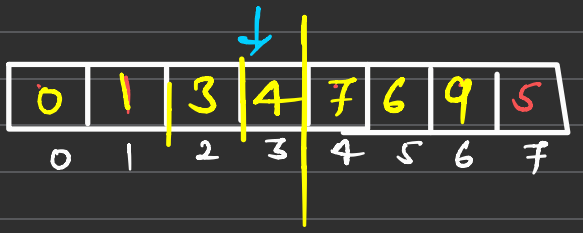


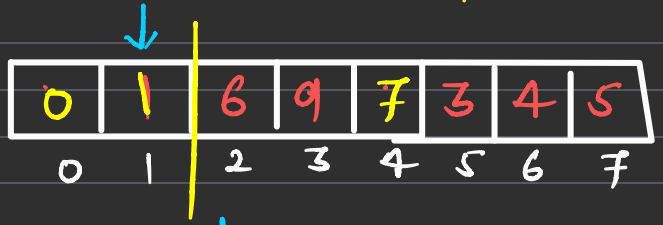
N — (N-1) Correct position

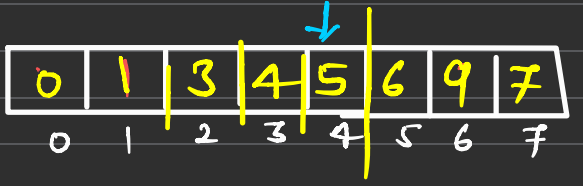
∴ 1 Rem. No.

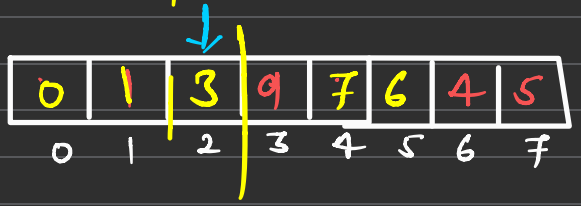


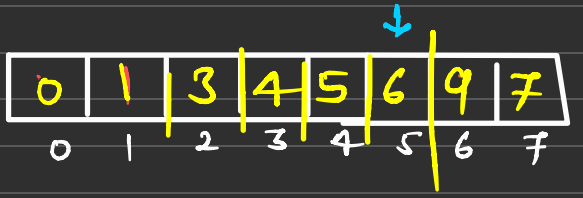






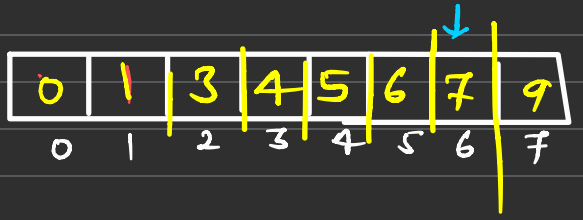






N — (N-1) Correct position

= 1 Rem. No.



arr

7	1	6	9	0	3	4	5
0	1	2	3	4	5	6	7

N = 8

ans = arr[0]
index = 0

```
for(i=0 ; i<N ; i++)
{
    if(a[i] < ans)
    {
        ans = a[i]
        index = i
    }
}
```

index = 4
num = 0

```
for (i=0 ; i< N; i++)        (N)
{
    ans = a[i]
    index = i
    for (j = i+1 ; j< N; j++)    (N)
    {
        if (a[j] < ans)
        {
            ans = a[j]
            index = j
        }
    }
    swap (a[i], a[index])
}
```

min. no. of swaps

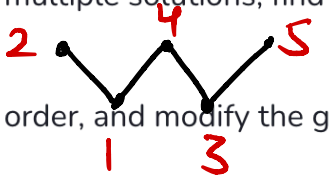
$T.C = O(N^2)$

$S.C = O(1)$

Wave Array

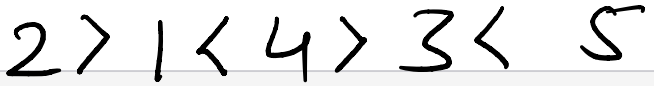
Difficulty: **Medium** Accuracy: **63.69%** Submissions: **278K+** Points: **4** Average Time: **20m**

Given an **sorted** array **arr[]** of integers. Sort the array into a **wave-like** array(In Place). In other words, **arrange the elements** into a sequence such that arr[1] >= arr[2] <= arr[3] >= arr[4] <= arr[5] ..... and so on. If there are multiple solutions, find the **lexicographically smallest** one.



Note: The given array is sorted in ascending order, and modify the given array in-place without returning a new array.

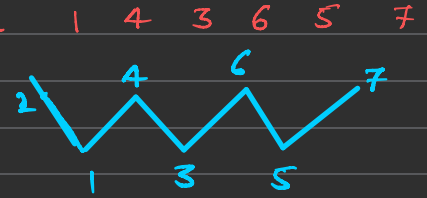
Examples:



Input: arr[] = [1, 2, 3, 4, 5]
Output: [2, 1, 4, 3, 5]
Explanation: Array elements after sorting it in the waveform are 2, 1, 4, 3, 5.

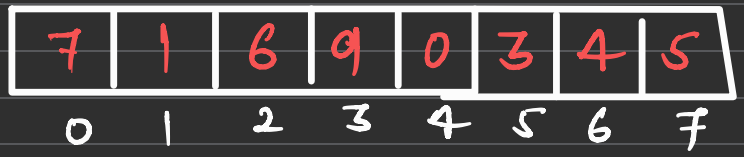
1	2	3	4	5	6	7
0	1	2	3	4	5	6

N = 7

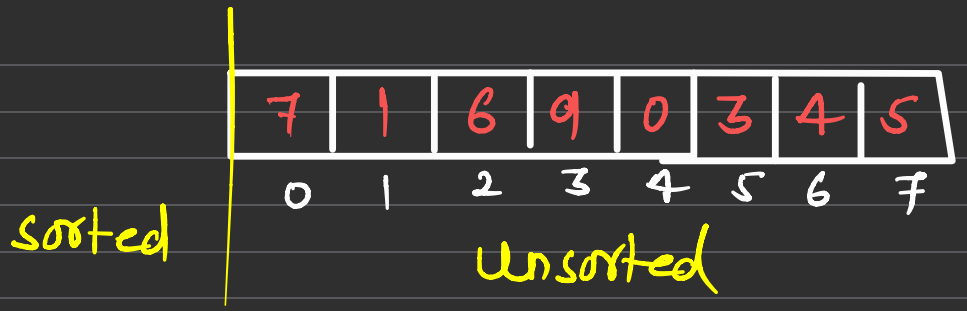



```
for ( i=0 ; i<N-1 ; i=i+2)
{
    swap ( a[i], a[i+1])
}
```

Insertion Sort

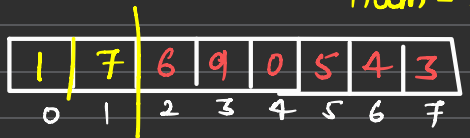


- Online sorting
- Mainly used when your data is almost sorted

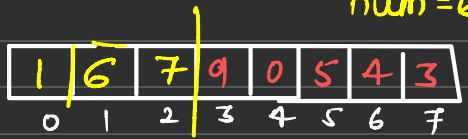


num = 1

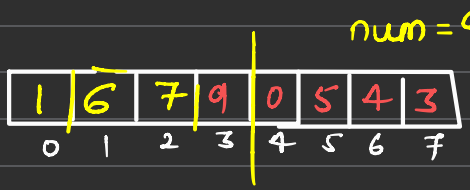
num = 7



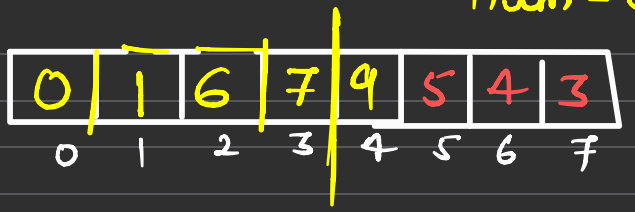
num = 6



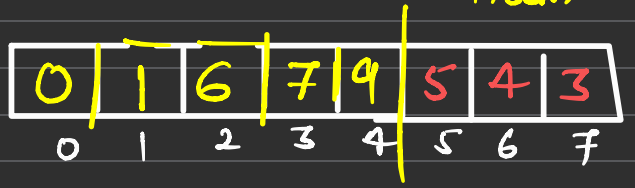
num = 9



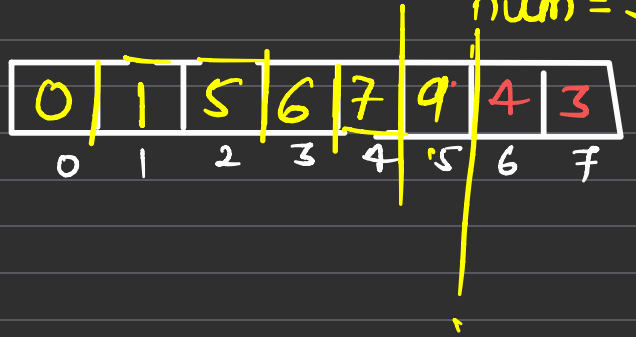
num = 0



num = 9



num = 5



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

3 ans = a[i]

index = i-1

(N-1) while (index ≥ 0 && a[index] > ans)

3 a[index+1] = a[index]

index--;

y

a[index+1] = ans

f

6	7	8	9	10	11
0	1	2	3	4	5

T:C = O(N²)

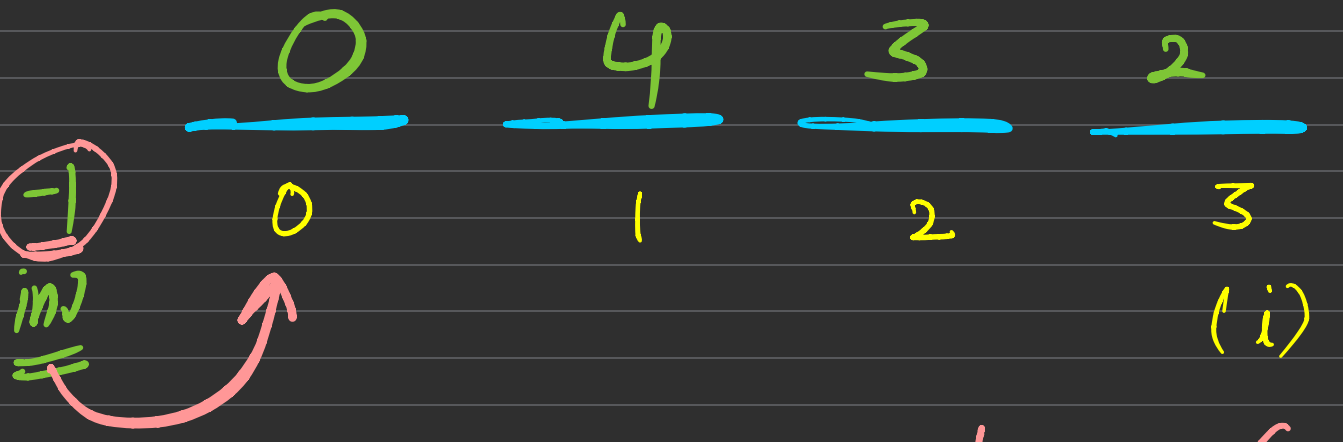
S:C = O(1)

a(index) = ans

a(-1) = ans

ans = 0

index = 2 → 0 → -1



ans = 4