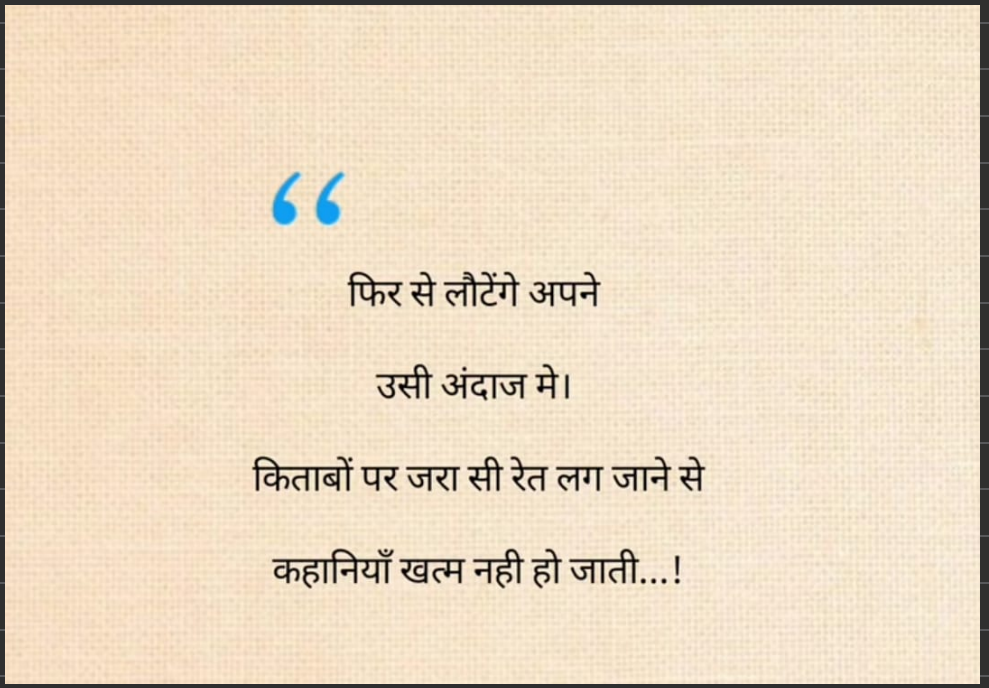
Sorting Algorithms



Made with Goodnotes

What is sorting

Arranging the numbers in either increasing or decreasing order.

Ascending → 1 1 2 3 4 4

descending → 5 5 4 3 2 1 1

increasing → 1 2 3 4 5 6

decreasing → 6 5 4 3 2 1

Made with Goodnotes

Stable Sorting

Relative position of duplicate numbers are maintained

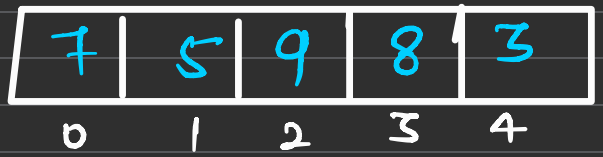
$1 \quad 2_a \quad 4 \quad 3 \quad 2_b \quad 6 \rightarrow 1 \quad 2_a \quad 2_b \quad 3 \quad 4$

In-place Sorting

No need of extra space to sort the elements.

Made with Goodnotes

Bubble Sort



- mainly compares two adj elements
  if they are incorrectly placed → swap

$x > y$ (incorrect)

Made with Goodnotes

N = 5

pass 1

7 5 9 8 3
5 7 9 8 3
5 7 9 8 3
5 7 8 9 3
5 7 8 3 9

(A)

pass 2

5 7 8 3 | 9
5 7 8 3 9
5 7 8 3 9
5 7 3 8 9

pass 3

5 7 3 | 8 9
5 7 3 8 9
5 3 7 8 9

pass 4

5 3 | 7 8 9
3 5 7 8 9

, E / E
H / H
B / B
V, / V
R / R
, P / P

Made with Goodnotes

Size of array = N (5)

No. of passes = N-1 (4)

size	No. of passes	Comparision
5	pass 1	4
	pass 2	3
	pass 3	2
	pass 4	1

(N - pass)

Made with Goodnotes

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

$O(N-1)$

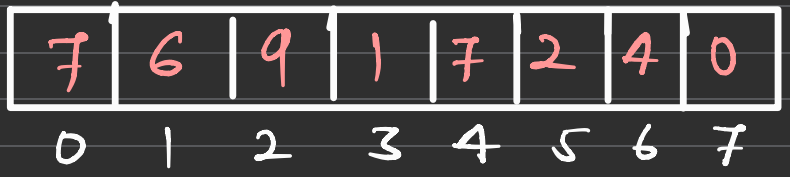
$O(N-1)$

$O(N-1) \times O(N-1)$

$O(N^2 - N - N + 1)$

$O(N^2)$

Made with Goodnotes



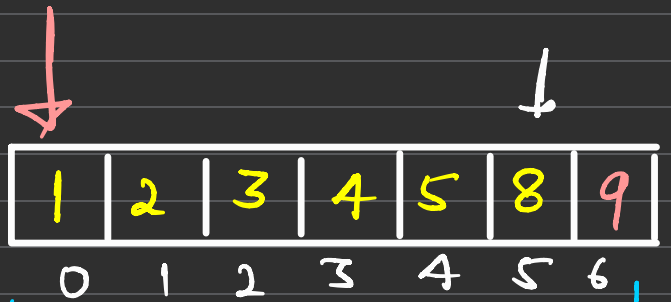
N = 8

Find the index of smallest No. ?

```
int small = a[0]
int index = 0

for (i = 1 ; i < N ; i++)
{
    if (a[i] < small)
    {
        small = a[i]
        index = i
    }
}
```

Made with Goodnotes

Selection Sort



smallNo = ~~9~~ 8

minIndex = ~~5~~ 6

(N-1) 1 > Correct position

Made with Goodnotes

```
void selectionsort (arr, N)
{
O(N)  for (i=0 ; i<N ; i++)
      {
        smallNo = a[i]
        minIndex = i
O(N)    for (j = i+1 ; j<n ; j++)
        {
          if (a[i] < smallNo)
          {
            smallNo = a[i]
            minIndex = i
          }
        }
        swap (arr[i], a[minIndex])
      }
}
```

O(N X N)

O(N²)

Made with Goodnotes

N = 7

9 3 8 4 5 1 2

Count the inversion ?

inversion : pair such that they are not in correct order , a(i) > a(i+k)

K = 0, 1, 2, 3.... N-1)

Made with Goodnotes

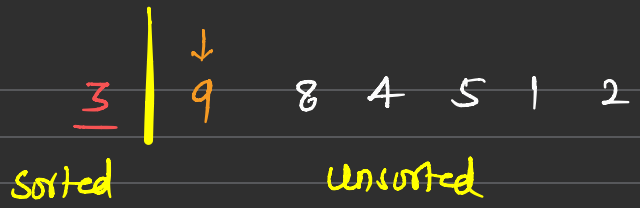
Made with Goodnotes

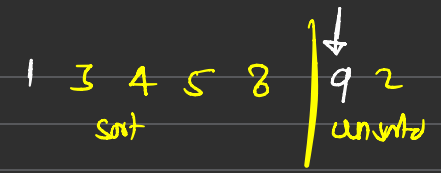
Insertion Sort

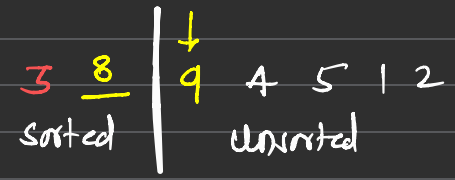
* Online Sorting

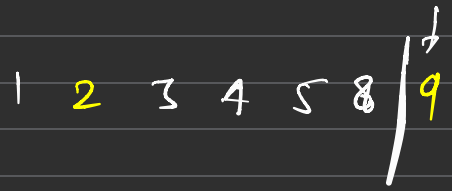
It does not wait for all the data to store → It keeps on sorting as data comes

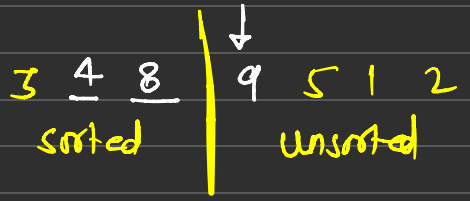
Made with Goodnotes

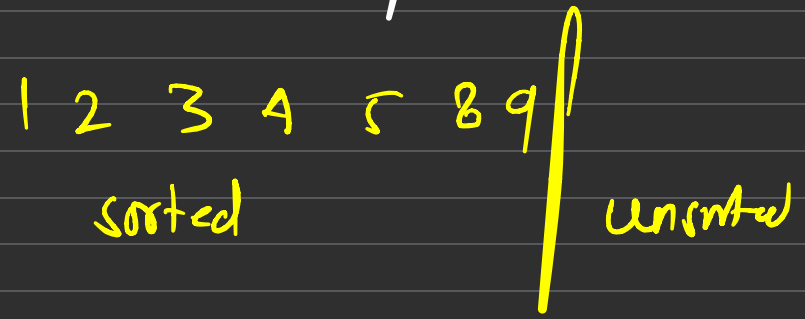


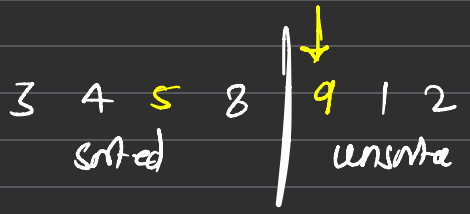












Made with Goodnotes

```
void insertionSort (arr[], N)
{
    for(i=1 ; i<N; i++)  ⟶ O(N)
    {
        num = a[i]
        j = i-1
        while ( j≥0 && a[j] > num)  ⟶ O(N)
        {
            a[j+1] = a[j]
            j--
        }
        arr[j+1] = num
    }
}
```

$O(N \times N)$

$O(N^2)$

Made with Goodnotes

Segregate 0s and 1s

Difficulty: **Easy** Accuracy: **54.25%** Submissions: **138K+** Points: **2** Average Time: **15m**

Given an array **arr** consisting of only **0**'s and **1**'s in random order. Modify the array **in-place** to segregate 0s onto the left side and 1s onto the right side of the array.

Examples :

Input: arr[] = [0, 0, 1, 1, 0]
Output: [0, 0, 0, 1, 1]
Explanation: After segregation, all the 0's are on the left and 1's are on the right. Modified array will be [0, 0, 0, 1, 1].

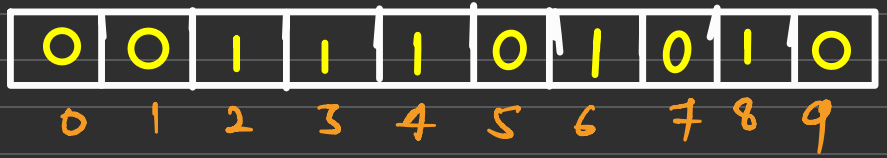
Input: arr[] = [1, 1, 1, 1]
Output: [1, 1, 1, 1]
Explanation: There are no 0s in the given array, so the modified array is [1, 1, 1, 1]

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(1)

Constraints:

$1 \leq arr.size() \leq 10^6$

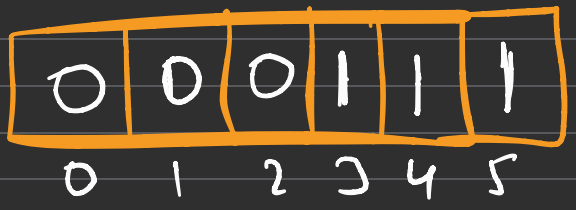
Made with Goodnotes



N = 10

- Sort the array → $O(N^2)$
- Traverse array to get cnt of zero → $O(2N)$
  & again traverse to put zero

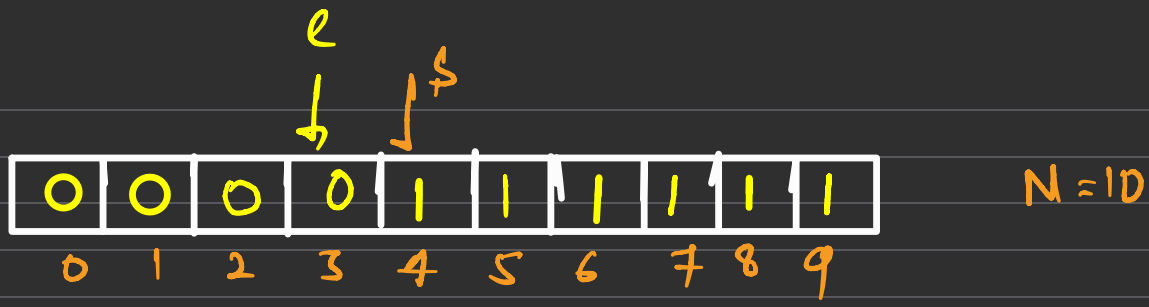
Made with Goodnotes

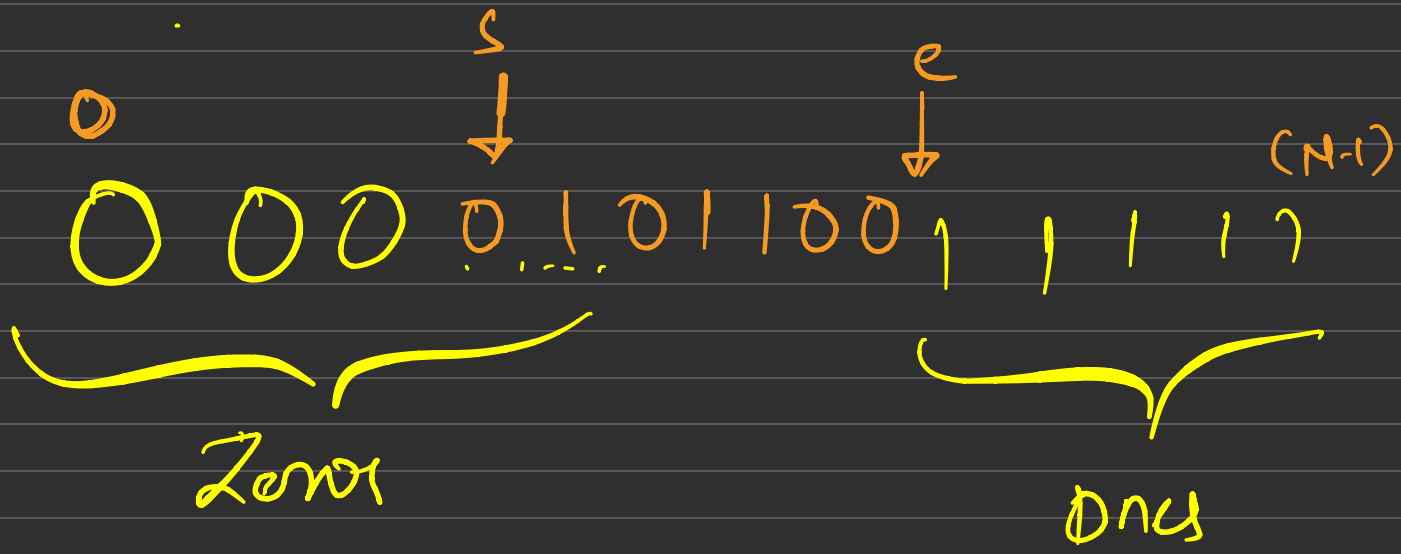



```
int cnt=0

for(i=0 ; i<N ; i++)          ——— O(N)
{   if(a(i)==0)
        cnt++
}

for(i=0 ; i<n ; i++)          ——— O(N)
{       if(cnt>0)
        {   a(i)=0
            cnt--
        }
        else
            a(i)=1
}
```

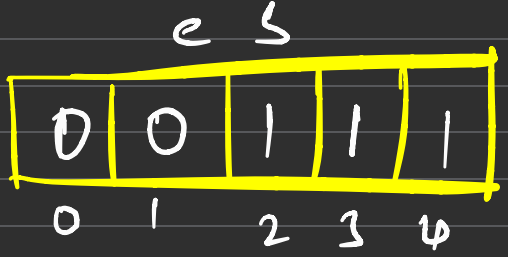
cnt = ~~3~~
~~2~~
~~1~~
0

Made with Goodnotes





Made with Goodnotes




```
s=0 , e = n-1

while (s<=e)  ____________ O(N)
{
    if (a[s]==0)
    {
        s++
    }
    else
    {
        swap (a[s], a[e])
        e--
    }
}
```

Made with Goodnotes

Alternative Sorting

Difficulty: **Basic** Accuracy: **50.2%** Submissions: **48K+** 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.

Input: arr[] = [1, 6, 9, 4, 3, 7, 8, 2]
Output: [9, 1, 8, 2, 7, 3, 6, 4]
Explanation: The first element is first maximum and second element is first minimum and so on.

Made with Goodnotes

Made with Goodnotes