Sorting

- It is a process of arranging items systematically.

* Bubble Sort *

- It is the simplest Algorithm that works by repeatedly swapping the adjacent element if they are in wrong order.

Example:

First Pass: 3, 1, 5, 4, 2

1, 3, 5, 4, 2

1, 3, 4, 5, 2

1, 3, 4, 2~~0~~, $\boxed{5}$

with first pass through the entire array, the largest element (here 5) came to the end

Second Pass: 1, 3, 4, 2, 5.

1, 3, 2, $\boxed{4, 5}$

with 2nd Pass 2nd largest element comes at second from the last index.

Third Pass: 1, 3, 2, 4, 5

$\boxed{1, 2, 3, 4, 5}$

sorted !!

don't need to compare again & again since it is already sorted.

Bubble sort is also known as Sinking sort or Exchange sort.

let's understand more deeply how Actually bubble sort works!! ..

counter
$i = 0$
1st pass.

$$\begin{array}{ccccc} i & j & & & \\ 3, & 1, & 5, & 4, & 2 \\ 0 & 1 & 2 & 3 & 4 \end{array}$$

swap ↓

$$\begin{array}{ccccc} & & j & j & \\ 1, & 3, & 5, & 4, & 2 \end{array}$$

↓ swap

1, 3, 4, 5, 2. $\circled{j}$ → internal loop it runs [n-1] times.

↓ swap

1, 3, 4, ~~5~~2, $\boxed{5.}$

$i = 1$
2nd pass.

$$\begin{array}{ccccc} & i & & & \\ 1, & 3, & 4, & 2, & 5. \\ & j & j & j & \end{array}$$

↓ swap.

$\underbrace{1, 3, 2~~4~~}$, $\boxed{4, 5}$ → j will only check this elements because after this already sorted

$i = 2$
3rd pass.

$$\begin{array}{ccccc} 1, & 3, & 2, & 4, & 5 \\ & j & j & & \end{array}$$

↓ swap

1, 2, 3, 4, 5

• Time Complexity

• Space Complexity : O(1) //constant.
⇒ since here no extra space is required.
i.e. like copying the array etc. is not required.
Also known as inplace sorting algorithm

• Time Complexity :
1) Best case → Array is sorted.
↳ O(N)

i = 0 1, 2, 3, 4, 5
1st pass j j j j

→ Only once it ran we don't need to check it again.

Note :- when j never swaps for value of i, it means array is sorted. Hence you can end the program.

★ Best Case Comparisons = N − 1 ⇒ $\textcircled{N}$
Since in time complexity constants are ignored, we don't want exact time, we just want relationship i.e. mathematical function

2) worst case ↳ sorting descending order to ascending
i = 0 5, 4, 3, 2, 1
1st pass j

4, 5, 3, 2, 1
j

4, 3, 5, 2, 1 ⇒ (N−1) swaps.
j

4, 3, 2, 5, 1
j

4, 3, 2, 1, 5

Page No.
Date :

$i = 1$ $\quad 4, \underset{j}{3}, \underset{\downarrow}{2}, 1, 5$

2nd pass

$3, \underset{\downarrow}{4}, \underset{j}{2}, 1, 5$

$3, 2, \underset{\downarrow}{4}, \underset{j}{1}, 5$ $\quad \Rightarrow (N-2)$ swaps.

$3, 2, 1, 4, 5.$

$i = 2$ $\quad 3, \underset{j}{2}, \underset{\downarrow}{1}, 4, 5$

3rd pass

$2, \underset{\downarrow}{3}, \underset{j}{1}, 4, 5$ $\quad \Rightarrow (N-3)$ swaps.

$2, 1, 3, 4, 5.$

$i = 3$ $\quad 2, \underset{j}{1}, \underset{\downarrow}{3}, 4, 5$ $\quad \Rightarrow (N-4)$ swaps.

4th pass

$1, 2, 3, 4, 5.$

total comparisons $= N-1 + N-2 + N-3 + N-4$

$$= 4N - (1+2+3+4)$$

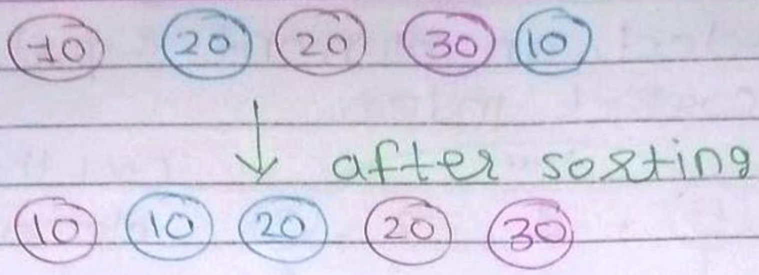
$$= 4N - \left[\frac{N \times (N+1)}{2}\right]$$

$$= 4N - \frac{N^2+N}{2}$$

$$= O\left(\frac{7N - N^2}{2}\right)$$

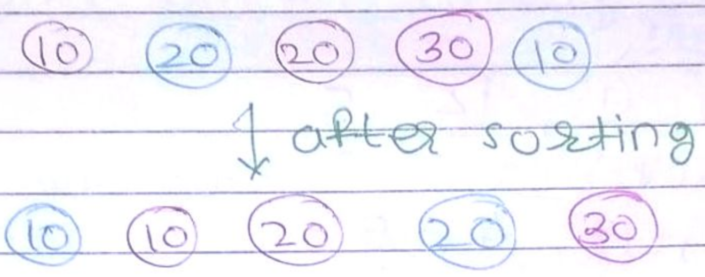
total comparisons $= O(N^2)$ $\quad$ In time complexity constant & less dominating terms are ignored.

Stable : Sorting Algo



⇒ Here after sorting the order is maintained
⇒ order is same when value is same.

Unstable : Sorting Algo



Ex:-

```
static void bubbleSort(int[] arr){
   boolean swapped;
  for(i=0; i<arr.length; i++){
     swapped = false;
   for (j=1 ; j<arr.length-i ; i++){
      if (arr[j]<arr[j-1]){
            int temp = arr[j];
            arr[j]  = arr[j-1];
            arr[j-1] = temp;
            swapped = true;
      }
   }
   if ( ! swapped){
        break;
   }.
```