BUBBLE SORT

BUBBLE SORT is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Agar hume kabhi bhi Bolay ki BUBBLING kro Toh it means BUBBLE-SORT ALGORITHM use kro!

☆ Har element apne se piche walay se compare hoga

☆ Agar chota hua toh agey jayega warna nahi jayega!

5	9	8	2	1

ITERATION 1

1.1 5 9 8 2 1

1.2 5 ~~9~~ 8 ~~8~~ 9 2 1

1.3 5 8 ~~9~~ 2 ~~2~~ 9 1

1.4 5 8 2 ~~9~~ 1 ~~1~~ 9

Array: 5 8 2 1 9

ITERATION 2

2.1 5 8 2 1 9

2.2 5 ~~8~~ 2 ~~2~~ 8 1 9

2.3 5 2 ~~8~~ 1 ~~1~~ 8 9

Array: 5 2 1 8 9

ITERATION 3

3.1 ~~5~~ 2 ~~2~~ 5 1 8 9

3.2 2 ~~5~~ 1 ~~1~~ 5 8 9

Array: 2 1 5 8 9

ITERATION 4

4.1 1 ~~2~~ ~~1~~ 2 5 8 9

Array: 1 2 5 8 9

1	2	5	8	9

Array Element 5 hai toh uski 4 iteration (Journey) Hogi, Aur 4 total elements sort hongy → WHY

Becoz Agar hum 4 elements ko sort krdey 5 elements me se. Then, 5th element toh ~~pe~~ khud hi apni correct place pe ajayega!

```
public static void main (String [] args) {
  Scanner s = new Scanner (System.in);
  int n = s.nextInt();
  int [] arr = new int [n];
  for ( int i = 0; i<n ; i++) {
     arr[i] = s.nextInt();
  }
  bubbleSort (arr);
  print (arr);
}
```

arguments me
↓ ARRAY jayega!

```
public static void  bubbleSort (int [] arr) {
  int n = arr.length;
 for (int itr = 1; itr<n ; itr++) {
    for (int j = 0; j<n-itr; j++) {
       if (isSmaller (arr, j+1, j) == true) {
          swap (arr, j+1, j);
       }
    }
  }
}
```

→ yeh loop no. of iteration/ journey ke liye laga hai!

→ Journey ① se $\boxed{n-1}$ tak hogi

→ Yeh loop comparison ke liye hai ∴ ~~itr~~ $\boxed{n-1-itr}$ tak comparison hoga

Example : Elements = 5

itr = 1 ∴ n-1-itr

$5-1-1 = 3 \rightarrow (0-3)$ ↓ i.e 4 comparisons

```
//used for swapping (ith) and (jth) elements of away
public static void swap (int [] arr, int i, int j) {
  System.out.println ("Swapping" + arr[i] + "and" + arr[j]);
  int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
}
// Returing true if (ith) element is smaller than (jth) element
public static bollean isSmaller (int [] arr, int i, int j) {
System.out.println ("Comparing" + arr[i] + "and" + arr[j]);
 if (arr[i] < arr[j]) {
     return true;
 }else {
     return false;
  }
}
public static void print (int [] arr) {
  for (int i=0; i<arr.length; i++) {
     System.out.println (arr[i]);
  }
}
```

★ Time Complexity

(n-1) comparisons krne me hume k time laga ∴ Time Taken = K(n-1)

ITERATION 1 : $T(n) = K(n-1) + T(\cancel{n-1})$ → Time taken for sorting (n-1) elements

ITERATION 2 : $T(\cancel{n-1}) = K(n-2) + T(\cancel{n-2})$

ITERATION 3 : $T(\cancel{n-2}) = K(n-3) + T(\cancel{n-3})$

+

LAST ITERATION : $T(\cancel{2}) = K(1)$ → Last me (2) elements ke bich me (1) comparison hoga!

$$\therefore T(n) = K(n-1) + K(n-2) + K(n-3) + \ldots\ldots + K(1)$$

$$= K[\underbrace{(n-1)+(n-2)+\ldots\ldots+(1)}_{A.P}]$$

$$= K\left[\frac{(n)(n-1)}{2}\right]$$

$$\therefore \boxed{T(n) \propto n^2}$$

WORST CASE : $O(n^2)$
↳ when array is in reverse Order

Best CASE : $O(n)$
↳ Array is already Sorted

Average CASE : $O(n^2)$

Inner loop does $O(n)$ work on each iteration

Outer loop does $O(n)$ iteration.

★ Space Complexity:

$O(1)$