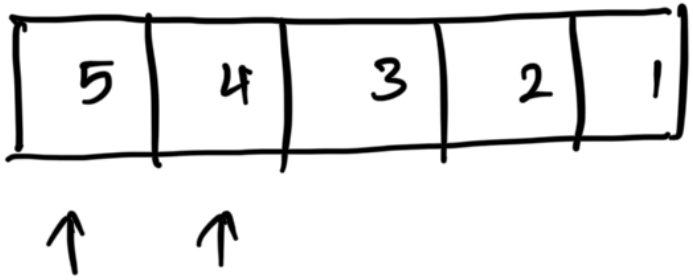
Bubble Sort

Q. What is Bubble sort? → (1)
Q. What is Passthrough? → (2)

A → (1) Bubble sort is one of the sorting algorithm to sort array's element in ascending order.

How it sorts?

- Two pointers are used, start from left of the array.



- The two pointer's value is compared, if the second pointer's value is out of order, that is second pointer's value is less then first pointer's value, then

$\therefore$ [5] < [4] = false

we swap the value.

- move the pointers forward, and repeat the above steps until the end of the array.

(2)

∴ Each time we follow the above steps, its know as one passthrough.

After each passthrough, the maximum is moved to the end.

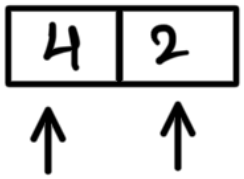
- if No swapping is done in a passthrough then it means array is sorted and we can break, else follow the above steps.

eg:

4	2	7	1	3

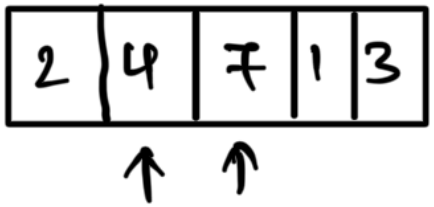
→ array

So we will start from left of array.

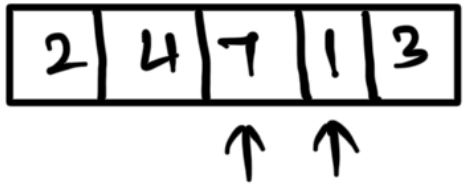


(two pointer)

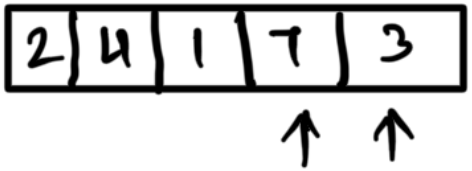
Here when compare we find that its false as '4' is greater then '2', so we swap. and move the pointer's forward



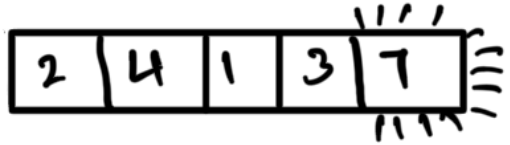
Here is true, No need of swapping. Hence moving pointers further.



Here its false, Hence swapping, and moving pointers furthers.



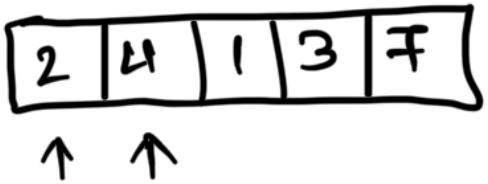
again follow the same.



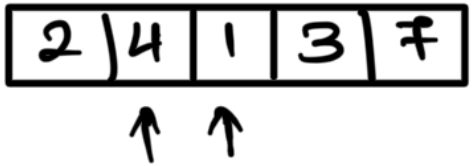
we have travers till the end of the array.

we have completed our first passthrough. After each passthrough, the maximum element is "bubbled" at the end of the array, and we can say for sure that element is in right position.

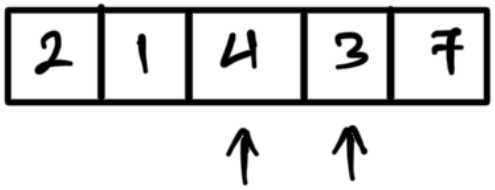
We are again going to follow the steps since we had <u>atleast one swap</u>.



'2' is less than '4', so no need of swap.

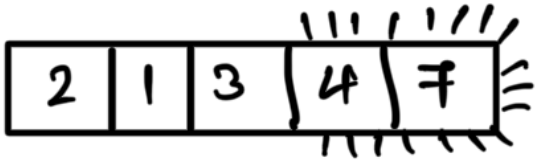


'1' is less than '4', so we swap.



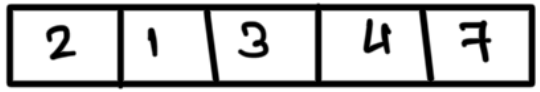
'3' is less than '4' so swapping

is the array, so swapping

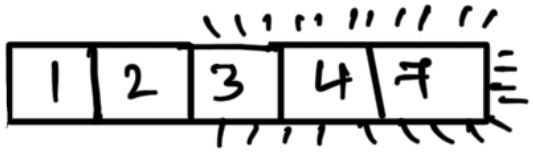


We know that, last element is already sorted, so no need to compare that, and by that we complete our 2nd passthrough.

At the end of 2nd passthrough, we can say for sure last 2 elements are in correct position. Since this passthrough had swapping, we will continue to loop through array.

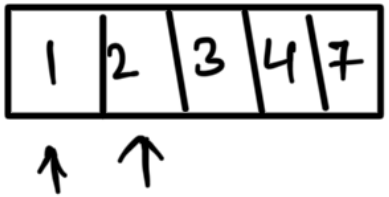


∴ 1 < 2, so



∴ 2 < 3, swapping not needed,

Since we had atleast one swap, will continue,



∴ 1 < 2, No need to swap.

All the other element are in correct order, no need to process further, pass through didn't have any swap, means array is sort Hence we can stop.

Pseudo code:

```
func  bubbleSort ( arr )
 {
   unsorted_till_index = len(arr) - 1 ;
   sorted = false;

   while ( NOT sorted )
    {
      sorted = true;   // assuming its sorted

     for ( i = 0 → unsorted_till_index - 1 )
      {
        if ( arr[i] > arr[i+1] )
```

If no element is swapped array is sorted

```
        {
            sorted = false;
            swap ( i , i+1 )
        }
    }
    unsorted_till_index -= 1 ;
```

// maximum element is bubbled at last

```
}

main ()
{
```

```
arr = [2, 3, 4, 5, 2, 1]
bubbleSort (arr)
print (arr);
}
```

Efficiency :

Two kinds of steps.

(1) Comparison : Compare two elements to find if elements are in correct order.

(11) Swap : Swapping two elements to

Sort the element.

comparison is done till unsorted index

eg: An array with 5 elements., it will compare,

- in 1st pass through, until last element
  total of 4 comparison happens.
- in 2nd pass through, comparison happens utill 2nd last element, as we know last element is sorted,
  total of 3 comparison happens.
- Before array is sorted, at last