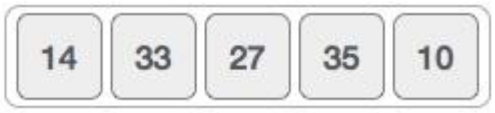
BUBBLE SORT

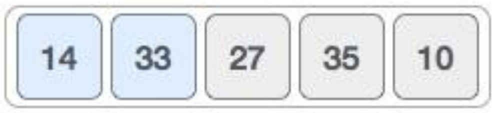
Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where **n** is the number of items.

How Bubble Sort Works?

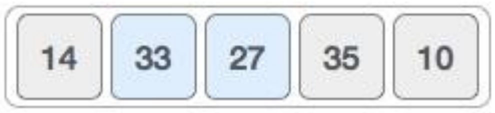
We take an unsorted array for our example. Bubble sort takes $O(n^2)$ time so we're keeping it short and precise.



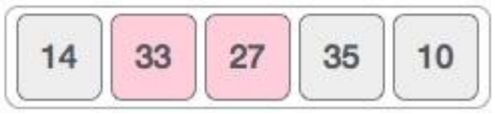
Bubble sort starts with very first two elements, comparing them to check which one is greater.



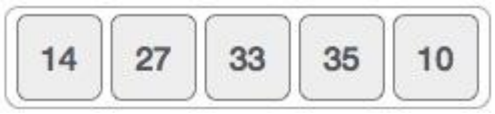
In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.



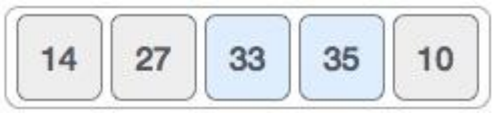
We find that 27 is smaller than 33 and these two values must be swapped.



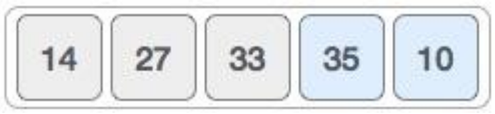
The new array should look like this −



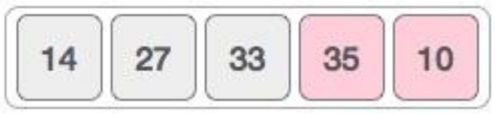
Next we compare 33 and 35. We find that both are in already sorted positions.



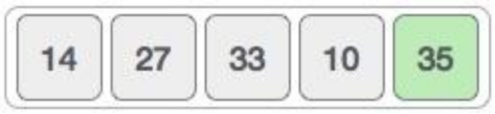
Then we move to the next two values, 35 and 10.



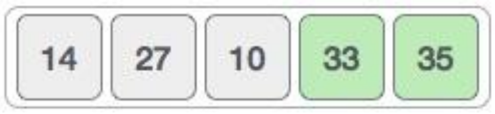
We know then that 10 is smaller 35. Hence they are not sorted.



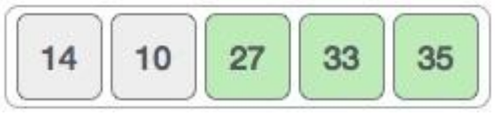
We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this −



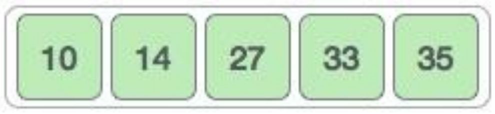
To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this −



Notice that after each iteration, at least one value moves at the end.



And when there's no swap required, bubble sorts learns that an array is completely sorted.



Now we should look into some practical aspects of bubble sort.

Algorithm

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

```
begin BubbleSort(list)

   for all elements of list
      if list[i] > list[i+1]
         swap(list[i], list[i+1])
      end if
   end for

   return list

end BubbleSort
```

Pseudocode

We observe in algorithm that Bubble Sort compares each pair of array element unless the whole array is completely sorted in an ascending order. This may cause a few complexity issues like what if the array needs no more swapping as all the elements are already ascending.

To ease-out the issue, we use one flag variable **swapped** which will help us see if any swap has happened or not. If no swap has occurred, i.e. the array requires no more processing to be sorted, it will come out of the loop.

Pseudocode of BubbleSort algorithm can be written as follows −

```
procedure bubbleSort( list : array of items )

   loop = list.count;

   for i = 0 to loop-1 do:
      swapped = false

      for j = 0 to loop-1 do:

         /* compare the adjacent elements */
         if list[j] > list[j+1] then
            /* swap them */
            swap( list[j], list[j+1] )
            swapped = true
         end if

      end for

      /*if no number was swapped that means
      array is sorted now, break the loop.*/

      if(not swapped) then
         break
      end if

   end for

end procedure return list
```

Implementation

One more issue we did not address in our original algorithm and its improvised pseudocode, is that, after every iteration the highest values settles down at the end of the array. Hence, the next iteration need not include already sorted elements. For this purpose, in our implementation, we restrict the inner loop to avoid already sorted values.

<u>**BUBBLE SORT MCQs**</u>

1. What is an external sorting algorithm?
a) Algorithm that uses tape or disk during the sort
b) Algorithm that uses main memory during the sort
c) Algorithm that involves swapping

d) Algorithm that are considered ‘in place’
View Answer
Answer: a
Explanation: As the name suggests, external sorting algorithm uses external memory like tape or disk.

2. What is an internal sorting algorithm?
a) Algorithm that uses tape or disk during the sort
b) Algorithm that uses main memory during the sort
c) Algorithm that involves swapping
d) Algorithm that are considered ‘in place’
View Answer
Answer: b
Explanation: As the name suggests, internal sorting algorithm uses internal main memory.

3. What is the worst case complexity of bubble sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)
View Answer
Answer: d
Explanation: Bubble sort works by starting from the first element and swapping the elements if required in each iteration.

advertisement

4. Select the appropriate code that performs bubble sort.
a)

```
for(int j=arr.length-1; j>=0; j--)
{
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                }
        }
}
```

b)

```
for(int j=arr.length-1; j>=0; j--)
{
        for(int k=0; k<j; k++)
        {
                if(arr[k] < arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                }
        }
}
```
c)

```
for(int j=arr.length; j>=0; j--)
{
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                }
        }
}
```
d)

```
for(int j=arr.length; j>=0; j--)
{
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+2])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                }
        }
}
```
View Answer
Answer: a
Explanation: The outer loop keeps count of number of iterations, and the inner loop checks to see if swapping is necessary.

5. What is the average case complexity of bubble sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)
View Answer
Answer: d
Explanation: Bubble sort works by starting from the first element and swapping the elements if required in each iteration even in the average case.

6. Which of the following is not an advantage of optimised bubble sort over other sorting techniques in case of sorted elements?
a) It is faster
b) Consumes less memory
c) Detects whether the input is already sorted
d) Consumes less time
View Answer

Answer: c
Explanation: Optimised Bubble sort is one of the simplest sorting techniques and perhaps the only advantage it has over other techniques is that it can detect whether the input is already sorted. It is faster than other in case of sorted array and consumes less time to describe whether the input array is sorted or not. It consumes same memory than other sorting techniques. Hence it is not an advantage.

7. The given array is arr = {1, 2, 4, 3}. Bubble sort is used to sort the array elements. How many iterations will be done to sort the array?
a) 4
b) 2
c) 1
d) 0
View Answer
Answer: a
Explanation: Even though the first two elements are already sorted, bubble sort needs 4 iterations to sort the given array.

8. How can you improve the best case efficiency in bubble sort? (The input is already sorted)
a)

```
        boolean swapped = false;
        for(int j=arr.length-1; j>=0 && swapped; j--)
        {
                swapped = true;
                for(int k=0; k<j; k++)
                {
                        if(arr[k] > arr[k+1])
                        {
                                int temp = arr[k];
                                arr[k] = arr[k+1];
                                arr[k+1] = temp;
                                swapped = false;
                        }
                }
        }
```

b)

```
        boolean swapped = true;
        for(int j=arr.length-1; j>=0 && swapped; j--)
        {
                swapped = false;
                for(int k=0; k<j; k++)
                {
                        if(arr[k] > arr[k+1])
                        {
                                int temp = arr[k];
                                arr[k] = arr[k+1];
                                arr[k+1] = temp;
                        }
                }
        }
```

c)

```
        boolean swapped = true;
        for(int j=arr.length-1; j>=0 && swapped; j--)
        {
                swapped = false;
                for(int k=0; k<j; k++)
                {
                        if(arr[k] > arr[k+1])
                        {
                                int temp = arr[k];
                                arr[k] = arr[k+1];
                                arr[k+1] = temp;
                                swapped = true;
                        }
                }
        }
```

d)

```
        boolean swapped = true;
        for(int j=arr.length-1; j>=0 && swapped; j--)
        {
                for(int k=0; k<j; k++)
                {
                        if(arr[k] > arr[k+1])
                        {
                                int temp = arr[k];
                                arr[k] = arr[k+1];
                                arr[k+1] = temp;
                                swapped = true;
                        }
                }
        }
```

View Answer
Answer: c
Explanation: A boolean variable ‘swapped’ determines whether any swapping has happened in a particular iteration, if no swapping has occurred, then the given array is sorted and no more iterations are required.

9. What is the best case efficiency of bubble sort in the improvised version?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)
View Answer
Answer: c
Explanation: Some iterations can be skipped if the list is sorted, hence efficiency improves to O(n).

10. The given array is arr = {1,2,4,3}. Bubble sort is used to sort the array elements. How many iterations will be done to sort the array with improvised version?
a) 4
b) 2
c) 1

d) 0
View Answer
Answer: b
Explanation: Only 2 elements in the given array are not sorted, hence only 2 iterations are required to sort them.