

[Home](#)[Data Structure](#)[C](#)[C++](#)[C#](#)[Java](#)[SQL](#)[HTML](#)[CSS](#)[JavaScript](#)[Ajax](#)[↑ SCROLL TO TOP](#)

Bubble sort Algorithm

In this article, we will discuss the Bubble sort Algorithm. The working procedure of bubble sort is simplest. This article will be very helpful and interesting to students as they might face bubble sort as a question in their examinations. So, it is important to discuss the topic.

Bubble sort works on the repeatedly swapping of adjacent elements until they are not in the intended order. It is called bubble sort because the movement of array elements is just like the movement of air bubbles in the water. Bubbles in water rise up to the surface; similarly, the array elements in bubble sort move to the end in each iteration.

Although it is simple to use, it is primarily used as an educational tool because the performance of bubble sort is poor in the real world. It is not suitable for large data sets. The average and worst-case complexity of Bubble sort is $O(n^2)$, where n is a number of items.

Bubble sort is majorly used where -

- complexity does not matter
- simple and short code is preferred

Algorithm

In the algorithm given below, suppose **arr** is an array of **n** elements. The assumed **swap** function in the algorithm will swap the values of given array elements.

```
begin BubbleSort(arr)
  for all array elements
    if arr[i] > arr[i+1]
      swap(arr[i], arr[i+1])
    end if
  end for
  return arr
end BubbleSort
```

Working of Bubble sort Algorithm

Now, let's see the working of Bubble sort Algorithm.

↑ SCROLL TO TOP

To understand the working of bubble sort algorithm, let's take an unsorted array. We are taking a short and accurate array, as we know the complexity of bubble sort is **$O(n^2)$** .

Let the elements of array are -

13	32	26	35	10
----	----	----	----	----

First Pass

Sorting will start from the initial two elements. Let compare them to check which is greater.

13	32	26	35	10
----	----	----	----	----

Here, 32 is greater than 13 ($32 > 13$), so it is already sorted. Now, compare 32 with 26.

13	32	26	35	10
----	----	----	----	----

Here, 26 is smaller than 36. So, swapping is required. After swapping new array will look like -

13	26	32	35	10
----	----	----	----	----

Now, compare 32 and 35.

13	26	32	35	10
----	----	----	----	----

Here, 35 is greater than 32. So, there is no swapping required as they are already sorted.

Now, the comparison will be in between 35 and 10.

13	26	32	35	10
----	----	----	----	----

Here, 10 is smaller than 35 that are not sorted. So, swapping is required. Now, we reach at the end of the array. After first pass, the array will be -

13	26	32	10	35
----	----	----	----	----

↑ SCROLL TO TOP

Now, move to the second iteration.

Second Pass

The same process will be followed for second iteration.

13	26	32	10	35
----	----	----	----	----

13	26	32	10	35
----	----	----	----	----

13	26	32	10	35
----	----	----	----	----

Here, 10 is smaller than 32. So, swapping is required. After swapping, the array will be -

13	26	10	32	35
----	----	----	----	----

13	26	10	32	35
----	----	----	----	----

Now, move to the third iteration.

Third Pass

The same process will be followed for third iteration.

13	26	10	32	35
----	----	----	----	----

13	26	10	32	35
----	----	----	----	----

Here, 10 is smaller than 26. So, swapping is required. After swapping, the array will be -

13	10	26	32	35
----	----	----	----	----

13	10	26	32	35
----	----	----	----	----

13	10	26	32	35
----	----	----	----	----

Now, move to the fourth iteration.

↑ SCROLL TO TOP

Fourth pass

Similarly, after the fourth iteration, the array will be -

10	13	26	32	35
----	----	----	----	----

Hence, there is no swapping required, so the array is completely sorted.

Bubble sort complexity

Now, let's see the time complexity of bubble sort in the best case, average case, and worst case. We will also see the space complexity of bubble sort.

1. Time Complexity

Case	Time Complexity
Best Case	$O(n)$
Average Case	$O(n^2)$
Worst Case	$O(n^2)$

- **Best Case Complexity** - It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of bubble sort is **$O(n)$** .
- **Average Case Complexity** - It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of bubble sort is **$O(n^2)$** .
- **Worst Case Complexity** - It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of bubble sort is **$O(n^2)$** .

2. Space Complexity

Space Complexity	$O(1)$
	YES

↑ SCROLL TO TOP

- The space complexity of bubble sort is $O(1)$. It is because, in bubble sort, an extra variable is required for swapping.
- The space complexity of optimized bubble sort is $O(2)$. It is because two extra variables are required in optimized bubble sort.

Now, let's discuss the optimized bubble sort algorithm.

Optimized Bubble sort Algorithm

In the bubble sort algorithm, comparisons are made even when the array is already sorted. Because of that, the execution time increases.

To solve it, we can use an extra variable **swapped**. It is set to **true** if swapping requires; otherwise, it is set to **false**.

It will be helpful, as suppose after an iteration, if there is no swapping required, the value of variable **swapped** will be **false**. It means that the elements are already sorted, and no further iterations are required.

This method will reduce the execution time and also optimizes the bubble sort.

Algorithm for optimized bubble sort

```
bubbleSort(array)
n = length(array)
repeat
  swapped = false
  for i = 1 to n - 1
    if array[i - 1] > array[i], then
      swap(array[i - 1], array[i])
      swapped = true
    end if
  end for
  n = n - 1
until not swapped
end bubbleSort
```

Now, let's see the programs of Bubble sort in different programming languages.

Program: Write a program to implement bubble sort in C language.

```
#include<stdio.h>

void print(int a[], int n) //function to print array elements
{
    int i;
    for(i = 0; i < n; i++)
    {
        printf("%d ",a[i]);
    }
}

void bubble(int a[], int n) // function to implement bubble sort
{
    int i, j, temp;
    for(i = 0; i < n; i++)
    {
        for(j = i+1; j < n; j++)
        {
            if(a[j] < a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
}

void main ()
{
    int i, j,temp;
    int a[5] = { 10, 35, 32, 13, 26};
    int n = sizeof(a)/sizeof(a[0]);
    printf("Before sorting array elements are - \n");
```

↑ SCROLL TO TOP

```
bubble(a, n);  
printf("\nAfter sorting array elements are - \n");  
print(a, n);  
}
```

Output

```
Before sorting array elements are -  
10 35 32 13 26  
After sorting array elements are -  
10 13 26 32 35
```

Program: Write a program to implement bubble sort in C++ language.

```
#include<iostream>  
  
using namespace std;  
  
void print(int a[], int n) //function to print array elements  
{  
    int i;  
    for(i = 0; i < n; i++)  
    {  
        cout<<a[i]<<" ";  
    }  
}  
  
void bubble(int a[], int n) // function to implement bubble sort  
{  
    int i, j, temp;  
    for(i = 0; i < n; i++)  
    {  
        for(j = i+1; j < n; j++)  
        {  
            if(a[j] < a[i])  
            {  
                temp = a[i];  
                a[i] = a[j];  
                a[j] = temp;  
            }  
        }  
    }  
}
```

↑ SCROLL TO TOP


```
    }

}

int main()
{
    int i, j, temp;
    int a[5] = {45, 1, 32, 13, 26};
    int n = sizeof(a)/sizeof(a[0]);
    cout<<"Before sorting array elements are - \n";
    print(a, n);
    bubble(a, n);
    cout<<"\nAfter sorting array elements are - \n";
    print(a, n);
    return 0;
}
```

Output

```
Before sorting array elements are -
45 1 32 13 26
After sorting array elements are -
1 13 26 32 45
```

Program: Write a program to implement bubble sort in C# language.

```
using System;
public class Bubble
{
    static void print (int[]a) //function to print array elements
    {
        int n = a.Length;
        int i;
        for (i = 0; i < n; i++)
        {
            Console.Write (" " + a[i]);
        }
    }
}
```

```
static void bubble (int[]a) // function to implement bubble sort
{
    int n = a.Length;
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (a[j] < a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
}

public static void Main ()
{
    int[] a = { 45, 1, 32, 13, 26 };
    Console.WriteLine ("Before sorting array elements are - \n");
    print (a);
    bubble (a);
    Console.WriteLine ();
    Console.WriteLine ("After sorting array elements are - \n");
    print (a);
}
}
```

Output

```
Before sorting array elements are -
45 1 32 13 26

After sorting array elements are -
1 13 26 32 45
```

[↑ SCROLL TO TOP](#)

Program to implement bubble sort in Java.

```
public class Bubble {  
    static void print (int a[]) //function to print array elements  
    {  
        int n = a.length;  
        int i;  
        for (i = 0; i < n; i++)  
        {  
            System.out.print(a[i] + " ");  
        }  
    }  
  
    static void bubbleSort (int a[]) // function to implement bubble sort  
    {  
        int n = a.length;  
        int i, j, temp;  
        for (i = 0; i < n; i++)  
        {  
            for (j = i + 1; j < n; j++)  
            {  
                if (a[j] < a[i])  
                {  
                    temp = a[i];  
                    a[i] = a[j];  
                    a[j] = temp;  
                }  
            }  
        }  
    }  
  
    public static void main(String[] args) {  
        int a[] = {35, 10, 31, 11, 26};  
        Bubble b1 = new Bubble();  
        System.out.println("Before sorting array elements are - ");  
        b1.print(a);  
        b1.bubbleSort(a);  
        System.out.println();  
        System.out.println("After sorting array elements are - ");  
    }  
}
```

[↑ SCROLL TO TOP](#)

```
}  
}
```

Output

```
D:\JTP>javac Bubble.java  
D:\JTP>java Bubble  
Before sorting array elements are -  
35 10 31 11 26  
After sorting array elements are -  
10 11 26 31 35
```

Program: Write a program to implement bubble sort in JavaScript.

```
<html>  
<head>  
</head>  
<body>  
<script>  
    var a = [35, 10, 31, 11, 26];  
    function print() //function to print array elements  
    {  
        for(i = 0; i < 5; i++)  
        {  
            document.writeln(a[i]);  
        }  
    }  
    document.write("Before sorting array elements are - " + "<br>");  
    print();  
  
    for(i = 0; i < 5; i++)  
    {  
        for (j = 0; j < 5; j++)  
        {  
            if(a[i] < a[j])  
            {  
                temp = a[i];  
                a[i] = a[j];  
                a[j] = temp;  
            }  
        }  
    }  
}
```

↑ SCROLL TO TOP

```
    }  
    }  
}  
document.write("<br> After sorting array elements are - " + "<br>");  
print();  
</script>  
</body>  
</html>
```

Output



Program: Write a program to implement bubble sort in PHP.

```
<?php  
$a = array(2, 45, 88, 11, 5);  
function printArray($a)  
{  
    for($i = 0; $i < 5; $i++)  
    {  
        print_r($a[$i]);  
        echo " ";  
    }  
}  
echo "Before sorting array elements are - <br>";  
printArray($a);  
for($i = 0; $i < 5; $i++)  
{  
    for ($j = 0; $j < 5; $j++)  
    {  
        if($a[$i] < $a[$j])
```

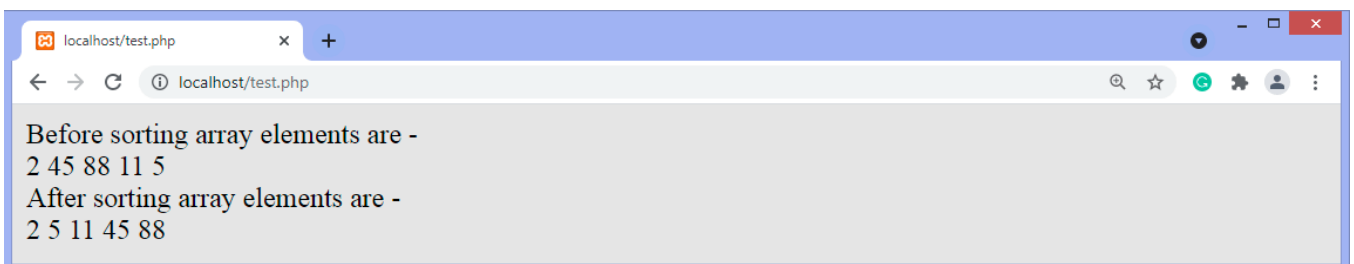
↑ SCROLL TO TOP

```

        $temp = $a[$i];
        $a[$i]=$a[$j];
        $a[$j] = $temp;
    }
}
}
echo "<br> After sorting array elements are - <br>";
printArray($a);
?>

```

Output



Program: Write a program to implement bubble sort in python.

```

a = [35, 10, 31, 11, 26]
print("Before sorting array elements are - ")
for i in a:
    print(i, end = " ")
for i in range(0,len(a)):
    for j in range(i+1,len(a)):
        if a[j]<a[i]:
            temp = a[j]
            a[j]=a[i]
            a[i]=temp
print("\nAfter sorting array elements are - ")
for i in a:
    print(i, end = " ")

```

Output

↑ SCROLL TO TOP

```
Before sorting array elements are -  
35 10 31 11 26  
After sorting array elements are -  
10 11 26 31 35
```

So, that's all about the article. Hope the article will be helpful and informative to you.

This article was not only limited to the algorithm. We have also discussed the algorithm's complexity, working, optimized form, and implementation in different programming languages.

[← Prev](#)[Next →](#)

 [For Videos Join Our Youtube Channel: Join Now](#)


Feedback


- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share





Learn Latest Tutorials


 [Splunk tutorial](#)
Splunk

 [SPSS tutorial](#)
SPSS


 [Swagger tutorial](#)
Swagger

 [T-SQL tutorial](#)
Transact-SQL


 [Tumblr tutorial](#)
Tumblr

 [React tutorial](#)
ReactJS


 [Regex tutorial](#)
Regex

 [Reinforcement learning tutorial](#)
Reinforcement Learning


[↑ SCROLL TO TOP](#)




R Programming tutorial
R Programming




RxJS tutorial
RxJS




React Native tutorial
React Native




Python Design Patterns
Python Design Patterns



Python Pillow tutorial
Python Pillow




Python Turtle tutorial
Python Turtle




Keras tutorial
Keras


Preparation




Aptitude
Aptitude




Logical Reasoning
Reasoning



Verbal Ability
Verbal Ability




Interview Questions
Interview Questions




Company Interview Questions
Company Questions


Trending Technologies




Artificial Intelligence Tutorial
Artificial Intelligence



AWS Tutorial
AWS




Selenium tutorial
Selenium




Cloud Computing tutorial
Cloud Computing




Hadoop tutorial
Hadoop



ReactJS Tutorial
ReactJS




Data Science Tutorial
Data Science




Angular 7 Tutorial
Angular 7

↑ SCROLL TO TOP



Blockchain
Tutorial

Blockchain




Git Tutorial

Git



Machine
Learning Tutorial

Machine Learning



DevOps
Tutorial

DevOps

B.Tech / MCA



DBMS tutorial

DBMS



Data Structures
tutorial

Data Structures



DAA tutorial

DAA



Operating
System tutorial

Operating System



Computer
Network tutorial

Computer Network



Compiler
Design tutorial

Compiler Design



Computer
Organization and
Architecture

Computer
Organization



Discrete
Mathematics
Tutorial

Discrete
Mathematics



Ethical Hacking
Tutorial

Ethical Hacking



Computer
Graphics Tutorial

Computer Graphics



Software
Engineering
Tutorial

Software
Engineering



html tutorial

Web Technology



Cyber Security
tutorial

Cyber Security



Automata
Tutorial

Automata



C Language
tutorial

C Programming



C++ tutorial

C++



Java tutorial

Java



.Net
Framework
tutorial

.Net



Python tutorial

Python



List of
Programs

Programs



Control
Systems tutorial

Control System



Data Mining
Tutorial

Data Mining



Data
Warehouse
Tutorial

Data Warehouse

↑ SCROLL TO TOP

