Bubble Sort

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

Example:

First Pass:

(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.

(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4

(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2

(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

(**1 4** 2 5 8) –> (**1 4** 2 5 8)

(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2

(1 2 **4 5** 8) –> (1 2 **4 5** 8)

(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

Third Pass:

(**1 2** 4 5 8) –> (**1 2** 4 5 8)

(1 **2 4** 5 8) –> (1 **2 4** 5 8)

(1 2 **4 5** 8) –> (1 2 **4 5** 8)

(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

Following is the implementations of Bubble Sort.

C++

```
// C++ program for implementation of Bubble sort
#include <bits/stdc++.h>
using namespace std;

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)

    // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
```

```
        if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// Driver code
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    cout<<"Sorted array: \n";
    printArray(arr, n);
    return 0;
}

// This code is contributed by rathbhupendra
```

C

```
// C program for implementation of Bubble sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program to test above functions
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Java

```
// Java program for implementation of Bubble Sort
class BubbleSort
{
    void bubbleSort(int arr[])
```

```
    {
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {
                    // swap arr[j+1] and arr[i]
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }

    /* Prints the array */
    void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Driver method to test above
    public static void main(String args[])
    {
        BubbleSort ob = new BubbleSort();
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
        ob.bubbleSort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
/* This code is contributed by Rajat Mishra */
```

Python

```
# Python program for implementation of Bubble Sort

def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]

# Driver code to test above
arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)

print ("Sorted array is:")
for i in range(len(arr)):
    print ("%d" %arr[i]),
```

C#

```
// C# program for implementation
// of Bubble Sort
using System;

class GFG
{
    static void bubbleSort(int []arr)
    {
        int n = arr.Length;
        for (int i = 0; i < n - 1; i++)
            for (int j = 0; j < n - i - 1; j++)
                if (arr[j] > arr[j + 1])
```

```
                {
                    // swap temp and arr[i]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
    }

    /* Prints the array */
    static void printArray(int []arr)
    {
        int n = arr.Length;
        for (int i = 0; i < n; ++i)
            Console.Write(arr[i] + " ");
        Console.WriteLine();
    }

    // Driver method
    public static void Main()
    {
        int []arr = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(arr);
        Console.WriteLine("Sorted array");
        printArray(arr);
    }

}

// This code is contributed by Sam007
```

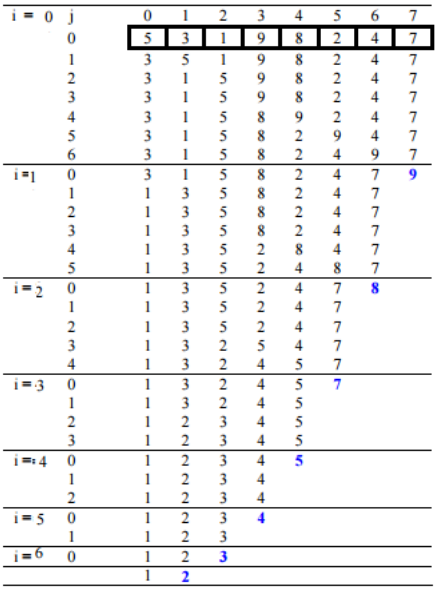
PHP

```
<?php
// PHP program for implementation
// of Bubble Sort

function bubbleSort(&$arr)
{
    $n = sizeof($arr);

    // Traverse through all array elements
    for($i = 0; $i < $n; $i++)
    {
        // Last i elements are already in place
        for ($j = 0; $j < $n - $i - 1; $j++)
        {
            // traverse the array from 0 to n-i-1
            // Swap if the element found is greater
            // than the next element
            if ($arr[$j] > $arr[$j+1])
            {
                $t = $arr[$j];
                $arr[$j] = $arr[$j+1];
                $arr[$j+1] = $t;
            }
        }
    }
}

// Driver code to test above
$arr = array(64, 34, 25, 12, 22, 11, 90);

$len = sizeof($arr);
bubbleSort($arr);

echo "Sorted array : \n";

for ($i = 0; $i < $len; $i++)
    echo $arr[$i]." ";

// This code is contributed by ChitraNayal.
?>
```

Output:

```
Sorted array:
11 12 22 25 34 64 90
```

<!—**Illustration :**



i = 0	j	0	1	2	3	4	5	6	7
	0	5	3	1	9	8	2	4	7
	1	3	5	1	9	8	2	4	7
	2	3	1	5	9	8	2	4	7
	3	3	1	5	9	8	2	4	7
	4	3	1	5	8	9	2	4	7
	5	3	1	5	8	2	9	4	7
	6	3	1	5	8	2	4	9	7
i =1	0	3	1	5	8	2	4	7	9
	1	1	3	5	8	2	4	7	
	2	1	3	5	8	2	4	7	
	3	1	3	5	8	2	4	7	
	4	1	3	5	2	8	4	7	
	5	1	3	5	2	4	8	7	
i =2	0	1	3	5	2	4	7	8	
	1	1	3	5	2	4	7		
	2	1	3	5	2	4	7		
	3	1	3	2	5	4	7		
	4	1	3	2	4	5	7		
i =3	0	1	3	2	4	5	7		
	1	1	3	2	4	5			
	2	1	2	3	4	5			
	3	1	2	3	4	5			
i =4	0	1	2	3	4	5			
	1	1	2	3	4				
	2	1	2	3	4				
i = 5	0	1	2	3	4				
	1	1	2	3					
i =6	0	1	2	3					
		1	2						

—>

Optimized Implementation:

The above function always runs O(n^2) time even if the array is sorted. It can be optimized by stopping the algorithm if inner loop didn't cause any swap.

CPP

```cpp
// Optimized implementation of Bubble sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// An optimized version of Bubble Sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   bool swapped;
   for (i = 0; i < n-1; i++)
   {
     swapped = false;
     for (j = 0; j < n-i-1; j++)
     {
        if (arr[j] > arr[j+1])
        {
           swap(&arr[j], &arr[j+1]);
           swapped = true;
        }
     }

     // IF no two elements were swapped by inner loop, then break
     if (swapped == false)
        break;
   }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
```

```
        printf("%d ", arr[i]);
    printf("n");
}

// Driver program to test above functions
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Java

```
// Optimized java implementation
// of Bubble sort
import java.io.*;

class GFG
{
    // An optimized version of Bubble Sort
    static void bubbleSort(int arr[], int n)
    {
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++)
        {
            swapped = false;
            for (j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    // swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // IF no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }

    // Function to print an array
    static void printArray(int arr[], int size)
    {
        int i;
        for (i = 0; i < size; i++)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Driver program
    public static void main(String args[])
    {
        int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
        int n = arr.length;
        bubbleSort(arr, n);
        System.out.println("Sorted array: ");
        printArray(arr, n);
    }
}


// This code is contributed
// by Nikita Tiwari.
```

Python3

```
# Python3 Optimized implementation
# of Bubble sort

# An optimized version of Bubble Sort
def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):
        swapped = False

        # Last i elements are already
        #  in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to
            # n-i-1. Swap if the element
            # found is greater than the
            # next element
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True

        # IF no two elements were swapped
        # by inner loop, then break
        if swapped == False:
            break

# Driver code to test above
arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)

print ("Sorted array :")
for i in range(len(arr)):
    print ("%d" %arr[i],end=" ")

# This code is contributed by Shreyanshi Arun
```

C#

```
// Optimized C# implementation
// of Bubble sort
using System;

class GFG
{
    // An optimized version of Bubble Sort
    static void bubbleSort(int []arr, int n)
    {
        int i, j, temp;
        bool swapped;
        for (i = 0; i < n - 1; i++)
        {
            swapped = false;
            for (j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    // swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // IF no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }

    // Function to print an array
    static void printArray(int []arr, int size)
    {
        int i;
        for (i = 0; i < size; i++)
```

```
            Console.Write(arr[i] + " ");
        Console.WriteLine();
    }

    // Driver method
    public static void Main()
    {
        int []arr = {64, 34, 25, 12, 22, 11, 90};
        int n = arr.Length;
        bubbleSort(arr,n);
        Console.WriteLine("Sorted array");
        printArray(arr,n);
    }

}
// This code is contributed by Sam007
```

PHP

```
<?php
// PHP Optimized implementation
// of Bubble sort

// An optimized version of Bubble Sort
function bubbleSort(&$arr)
{
    $n = sizeof($arr);

    // Traverse through all array elements
    for($i = 0; $i < $n; $i++)
    {
        $swapped = False;

        // Last i elements are already
        // in place
        for ($j = 0; $j < $n - $i - 1; $j++)
        {

            // traverse the array from 0 to
            // n-i-1. Swap if the element
            // found is greater than the
            // next element
            if ($arr[$j] > $arr[$j+1])
            {
                $t = $arr[$j];
                $arr[$j] = $arr[$j+1];
                $arr[$j+1] = $t;
                $swapped = True;
            }
        }

        // IF no two elements were swapped
        // by inner loop, then break
        if ($swapped == False)
            break;
    }
}

// Driver code to test above
$arr = array(64, 34, 25, 12, 22, 11, 90);
$len = sizeof($arr);
bubbleSort($arr);

echo "Sorted array : \n";

for($i = 0; $i < $len; $i++)
    echo $arr[$i]." ";

// This code is contributed by ChitraNayal.
?>
```

Output:

```
Sorted array:
11 12 22 25 34 64 90
```

Worst and Average Case Time Complexity: O(n*n). Worst case occurs when array is reverse sorted.

Best Case Time Complexity: O(n). Best case occurs when array is already sorted.

Auxiliary Space: O(1)

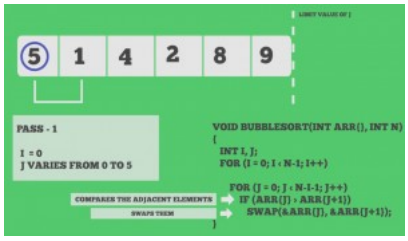
Boundary Cases: Bubble sort takes minimum time (Order of n) when elements are already sorted.

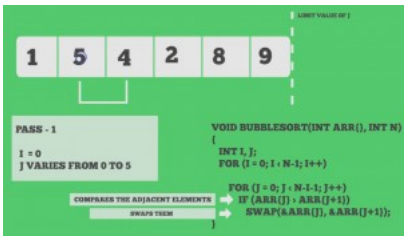
Sorting In Place: Yes

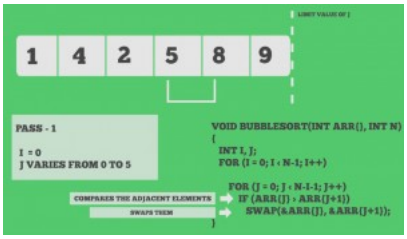
Stable: Yes

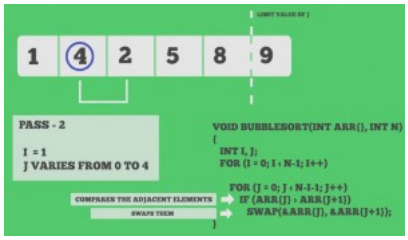
Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.
In computer graphics it is popular for its capability to detect a very small error (like swap of just two elements) in almost-sorted arrays and fix it with just linear complexity (2n). For example, it is used in a polygon filling algorithm, where bounding lines are sorted by their x coordinate at a specific scan line (a line parallel to x axis) and with incrementing y their order changes (two elements are swapped) only at intersections of two lines (Source: Wikipedia)

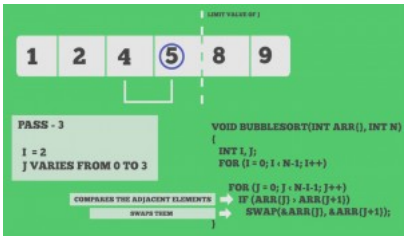
Snapshots:

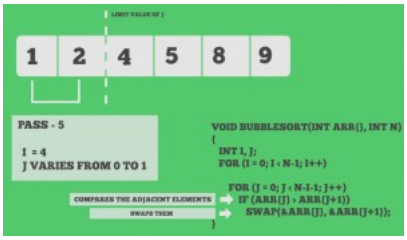












Quiz on Bubble Sort

Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz:

- Selection Sort
- Insertion Sort
- Merge Sort
- Heap Sort

- QuickSort
- Radix Sort
- Counting Sort
- Bucket Sort
- ShellSort

Recursive Bubble Sort

Coding practice for sorting.

Reference:

- Wikipedia – Bubble Sort
- Image Source

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Comparison among Bubble Sort, Selection Sort and Insertion Sort
Recursive Bubble Sort
C Program for Bubble Sort
C++ Program for Bubble Sort
Bubble sort using two Stacks
Sorting Strings using Bubble Sort
C++ Program for Recursive Bubble Sort
Java Program for Bubble Sort
Python Program for Bubble Sort
Java Program for Recursive Bubble Sort
Sorting Algorithms Visualization : Bubble Sort
C Program for Bubble Sort on Linked List
Bubble Sort On Doubly Linked List
Bubble Sort for Linked List by Swapping nodes
Why Quick Sort preferred for Arrays and Merge Sort for Linked Lists?

Improved By : Ita_c, SumitBM, rathbhupendra

Article Tags : Sorting redBus

Practice Tags : redBus Sorting

39

To-do Done

1.7

Based on **153** vote(s)

Feedback/ Suggest Improvement Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments