

LEARN DATA STRUCTURE
organizing data

Data Structures & Algorithms

DSA - Home

DSA - Overview

DSA - Environment Setup

Algorithm

DSA - Algorithms Basics

DSA - Asymptotic Analysis

DSA - Greedy Algorithms

DSA - Divide and Conquer

DSA - Dynamic Programming

Data Structures

DSA - Data Structure Basics

DSA - Array Data Structure

Linked Lists

DSA - Linked List Basics

DSA - Doubly Linked List

DSA - Circular Linked List

Stack & Queue

DSA - Stack

DSA - Expression Parsing

DSA - Queue

Searching Techniques

DSA - Linear Search

DSA - Binary Search

DSA - Interpolation Search

DSA - Hash Table

Sorting Techniques

DSA - Sorting Algorithms

DSA - Bubble Sort

DSA - Insertion Sort

DSA - Selection Sort

DSA - Merge Sort

DSA - Shell Sort

DSA - Quick Sort

Graph Data Structure

DSA - Graph Data Structure

DSA - Depth First Traversal

DSA - Breadth First Traversal

Tree Data Structure

DSA - Tree Data Structure

DSA - Tree Traversal

DSA - Binary Search Tree

DSA - AVL Tree

DSA - Spanning Tree

DSA - Heap

Recursion

DSA - Recursion Basics

DSA - Tower of Hanoi

DSA - Fibonacci Series

DSA Useful Resources

DSA - Questions and Answers

DSA - Quick Guide

DSA - Useful Resources

DSA - Discussion

Selected Reading

UPSC IAS Exams Notes

Developer's Best Practices

Questions and Answers

Effective Resume Writing


HR Interview Questions

Computer Glossary

Who is Who

Data Structure - Bubble Sort Algorithm

Advertisements



Previous Page

Next Page

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.

14

33

27

35

10

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

14

33

27

35

10

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

14

33

27

35

10

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

14

33

27

35

10

The new array should look like this –

14

27

33

35

10

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

14

27

33

35

10

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

14

27

33

35

10

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

14

27

33

35

10

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

14

27

33

10

35

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 –

14

27

10

33

35

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

14

10

27

33

35

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

10

14

27

33

35

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.


To know about bubble sort implementation in C programming language, please click here [↗](#) .


Previous Page

Print

Next Page

Advertisements








Tutorials Point (India) Pvt. Ltd.

YouTube

65K







Write for us | FAQ's | Helping | Contact

© Copyright 2018. All Rights Reserved.

Enter email for newsletter

go