

## TUTORIAL

# Bubble Sort

## Chapter

### 1. Bubble Sort

#### Topics

1.2 Algorithm - Sort in ascending order

1.4 Optimising Above Implementation

1.7 Video Solution

Bubble Sort is the simplest comparison based sorting algorithm. It is in-place and needs no extra memory.

#### Idea:

This algorithm repeatedly goes through the array, compares adjacent elements and swaps them if they are in the wrong order. This is not an efficient sorting algorithm, but due to its simplicity, it is often introduced to the students for understanding the foundations of sorting.

## Algorithm - Sort in ascending order

- Repeatedly traverse the array, and in each pass compare the adjacent elements. If the order of adjacent elements is wrong (*i.e.*  $arr[j] > arr[j+1]$ ), then swap them.
- Suppose the length of the array is **n**. Now to sort the array we need to do at max **n-1** passes(traversals) on it.

- After the first pass, the largest element will move to the last index (*i.e. index  $n-1$* ).
- Similarly, after the second pass, the second largest element will move to the second last index (*i.e. index  $n-2$* ).
- After the  $i$ th pass, the  $i$ th largest element will move to the *index  $n-i$* . That means after the  $i$ th pass, the last  $i$  elements in the array will be at their correct positions. Therefore in the next pass, we need to check the adjacent elements only till the  **$n-i-1$**  index.
- Finally, after doing all the passes the given array will be sorted.

Let's visualise the algorithm with the following example:

```
arr[ ] = {6, 3, 8, 9, 5}
```

### First Pass :

{**6**, **3**, 8, 9, 5}

compare  $arr[0]$  with  $arr[1]$  :  $arr[0] > arr[1]$  so we'll swap them, and the array will become {3, 6, 8, 9, 5}

{3, **6**, **8**, 9, 5}

compare  $arr[1]$  with  $arr[2]$  :  $arr[1] < arr[2]$  so no swapping will happen, and the array will remain same {3, 6, 8, 9, 5}

{3, 6, **8**, **9**, 5}

compare  $arr[2]$  with  $arr[3]$  :  $arr[2] < arr[3]$  so no swapping will happen, and the array will remain same {3, 6, 8, 9, 5}

{3, 6, 8, **9**, **5**}

compare  $arr[3]$  with  $arr[4]$  :  $arr[3] > arr[4]$  so we'll swap them, and the array will become {3, 6, 8, 5, **9**}

***After the first pass, the largest element comes to the last index.***

### Second Pass :

{**3**, **6**, 8, 5, 9}

compare  $arr[0]$  with  $arr[1]$  :  $arr[0] < arr[1]$  so no swapping will happen,  
and the array will remain same {3, 6, 8, 5, 9}

{3, **6**, **8**, 5, 9}

compare  $arr[1]$  with  $arr[2]$  :  $arr[1] < arr[2]$  so no swapping will happen,  
and the array will remain same {3, 6, 8, 5, 9}

{3, 6, **8**, **5**, 9}

compare  $arr[2]$  with  $arr[3]$  :  $arr[2] > arr[3]$  so we'll swap them, and the  
array will become {3, 6, 5, **8**, **9**}

***After the second pass, the second largest element has come to its correct position.***

**Third Pass :**

{**3**, **6**, 5, 8, 9}

compare  $arr[0]$  with  $arr[1]$  :  $arr[0] < arr[1]$  so no swapping will happen,  
and the array will remain same {3, 6, 5, 8, 9}

{3, **6**, **5**, 8, 9}

compare  $arr[1]$  with  $arr[2]$  :  $arr[1] > arr[2]$  so we'll swap them, and the  
array will become {3, 5, **6**, **8**, **9**}

***After the third pass, the third largest element has come to the third last position. Also we can notice that the last 3 elements have occupied their correct positions after the third pass.***

**Fourth Pass :**

{**3**, **5**, 6, 8, 9}

compare  $arr[0]$  with  $arr[1]$  :  $arr[0] < arr[1]$  so no swapping will happen,  
and the array will remain same {3, **5**, **6**, **8**, **9**}

*No need to check for the remaining elements on the right, as they are already at their correct positions.*

**After doing all the passes, we can clearly observe that the given array is sorted in ascending order.**

### Pseudo Code

```
for i := 0 to n-2 do
  for j := 0 to n-i-2 do
    if arr[j] > A[j+1]
      swap(A[j], A[j+1])
    endif
  end
end
end
```

```
1  function bubbleSort(arr,n){
2
3    for (let i = 0; i < n-1; i++){
4      // last i elements are already at the correct
      position
5      for (let j = 0; j < n-i-1; j++){
6        if (arr[j] > arr[j+1]){
7          // swap arr[j], arr[j+1]
8          let temp = arr[j]
9          arr[j] = arr[j+1]
10         arr[j+1] = temp
11       }
12     }
13   }
14 }
15
16 function printArray(arr){
17   console.log(arr.join(' '))
18 }
19
20 function main(){
21   let arr = [6, 3, 8, 9, 5]
22   let n = arr.length
```

JavaScript

```
23
24     console.log("Given Array: ")
25     printArray(arr)
26
27     bubbleSort(arr, n)
28
29     console.log("Sorted Array: ")
30     printArray(arr, n)
31 }
32
33 main()
```

```
1 #include <stdio.h>
2
3 void swap(int *a, int *b)
4 {
5     int temp = *a;
6     *a = *b;
7     *b = temp;
8 }
9
10 void bubbleSort(int arr[], int n)
11 {
12     for (int i = 0; i < n-1; i++)
13     {
14         // last i elements are already at the correct
15         position
16         for (int j = 0; j < n-i-1; j++)
17         {
18             if (arr[j] > arr[j+1])
19                 swap(&arr[j], &arr[j+1]);
20         }
21     }
22
23 void printArray(int arr[], int n)
24 {
25     for (int i = 0; i < n; i++)
```

```

26     printf("%d ", arr[i]);
27     printf("\n");
28 }
29
30 int main()
31 {
32     int arr[] = {6, 3, 8, 9, 5};
33     int n = sizeof(arr)/sizeof(arr[0]);
34
35     printf("Given Array: ");
36     printArray(arr, n);
37
38     bubbleSort(arr, n);
39
40     printf("Sorted Array: ");
41     printArray(arr, n);
42
43     return 0;
44 }
45

```

```

1  public class Main
2  {
3      static void bubbleSort(int arr[], int n)
4      {
5          for (int i = 0; i < n-1; i++)
6          {
7              // last i elements are already at the correct
position
8              for (int j = 0; j < n-i-1; j++)
9              {
10                 if (arr[j] > arr[j+1])
11                 {
12                     // swap arr[j], arr[j+1]
13                     int temp = arr[j];
14                     arr[j] = arr[j+1];
15                     arr[j+1] = temp;
16                 }

```

Java

```

17         }
18     }
19 }
20
21 static void printArray(int arr[], int n)
22 {
23     for (int i = 0; i < n; i++)
24         System.out.print(arr[i] + " ");
25     System.out.println();
26 }
27
28 public static void main(String args[])
29 {
30     int arr[] = {6, 3, 8, 9, 5};
31     int n = arr.length;
32
33     System.out.print("Given Array: ");
34     printArray(arr, n);
35
36     bubbleSort(arr, n);
37
38     System.out.print("Sorted Array: ");
39     printArray(arr, n);
40 }
41 }
42

```

```

1
2 def bubbleSort(arr,n):
3     for i in range(n-1):
4         # last i elements are already at the correct
position
5         for j in range(n-i-1):
6             if(arr[j] > arr[j+1]):
7                 # Swapping elements
8                 arr[j],arr[j+1] = arr[j+1],arr[j]
9
10

```

Python 3

```

11 def printArray(arr):
12     print(' '.join(str(x) for x in arr))
13
14 if __name__ == '__main__':
15     arr = [6, 3, 8, 9, 5]
16     n = len(arr)
17     print('Given Array:')
18     printArray(arr)
19
20     bubbleSort(arr,n)
21
22     print('Sorted Array:')
23     printArray(arr)

```

```

1  #include <iostream>
2  using namespace std;
3
4  void swap(int *a, int *b)
5  {
6      int temp = *a;
7      *a = *b;
8      *b = temp;
9  }
10
11 void bubbleSort(int arr[], int n)
12 {
13     for (int i = 0; i < n-1; i++)
14     {
15         // last i elements are already at the correct
16         position
17         for (int j = 0; j < n-i-1; j++)
18         {
19             if (arr[j] > arr[j+1])
20                 swap(&arr[j], &arr[j+1]);
21         }
22     }
23 }

```

C++



```

24 void printArray(int arr[], int n)
25 {
26     for (int i = 0; i < n; i++)
27         cout << arr[i] << " ";
28     cout << "\n";
29 }
30
31 int main()
32 {
33     int arr[] = {6, 3, 8, 9, 5};
34     int n = sizeof(arr)/sizeof(arr[0]);
35     cout<<"Given Array: ";
36     printArray(arr, n);
37
38     bubbleSort(arr, n);
39
40     cout<<"Sorted Array: ";
41     printArray(arr, n);
42
43     return 0;
44 }
45

```

## Optimising Above Implementation

---

In the above implementation, we have to do  $n-1$  passes on the array even if the array was already sorted in some  $i$ th pass. It can be optimised by stopping the algorithm if no swapping happened at all in the current pass.

### Pseudo Code

```

for i := 0 to n-2 do
    swapped = false;
    for j := 0 to n-i-2 do
        if arr[j] > A[j+1]
            swap(A[j], A[j+1])
            swapped = true;
        endif
    endfor
endfor

```

```
    end
    if swapped == false
        break;
    end
end
```

## Javascript

```
1
2 function bubbleSort(arr,n){
3     for (let i = 0; i < n-1; i++){
4         let swapped = false
5         // last i elements are already at the correct
position
6         for (let j = 0; j < n-i-1; j++){
7             if (arr[j] > arr[j+1]){
8                 // swap arr[j], arr[j+1]
9                 let temp = arr[j]
10                arr[j] = arr[j+1]
11                arr[j+1] = temp
12                swapped = true
13            }
14        }
15        // If no swapping happened in the current pass, then
break
16        if(!swapped){
17            break;
18        }
19    }
20 }
21
22 function printArray(arr){
23     console.log(arr.join(' '))
24 }
25
26 function main(){
27     let arr = [6, 3, 8, 9, 5]
28     let n = arr.length
29
30     console.log("Given Array: ")
31     printArray(arr)
```

```
32
33     bubbleSort(arr, n)
34
35     console.log("Sorted Array: ")
36     printArray(arr, n)
37 }
38
39 main()
```

```
1  #include <stdio.h>
2
3  void swap(int *a, int *b)
4  {
5      int temp = *a;
6      *a = *b;
7      *b = temp;
8  }
9
10 void bubbleSort(int arr[], int n)
11 {
12     for (int i = 0; i < n-1; i++)
13     {
14         int swapped = 0;
15         // last i elements are already at the correct
position
16         for (int j = 0; j < n-i-1; j++)
17         {
18             if (arr[j] > arr[j+1])
19             {
20                 swap(&arr[j], &arr[j+1]);
21                 swapped = 1;
22             }
23         }
24         // If no swapping happened in the current pass,
then break
25         if (swapped == 0)
26             break;
27     }
28 }
```

```

29
30 void printArray(int arr[], int n)
31 {
32     for (int i = 0; i < n; i++)
33         printf("%d ", arr[i]);
34     printf("\n");
35 }
36
37 int main()
38 {
39     int arr[] = {6, 3, 8, 9, 5};
40     int n = sizeof(arr)/sizeof(arr[0]);
41
42     printf("Given Array: ");
43     printArray(arr, n);
44
45     bubbleSort(arr, n);
46
47     printf("Sorted Array: ");
48     printArray(arr, n);
49
50     return 0;
51 }
52

```

```

1 public class Main
2 {
3     static void bubbleSort(int arr[], int n)
4     {
5         for (int i = 0; i < n-1; i++)
6         {
7             Boolean swapped = false;
8             // last i elements are already at the correct
position
9             for (int j = 0; j < n-i-1; j++)
10            {
11                if (arr[j] > arr[j+1])
12                {

```

Java

```
13         // swap arr[j], arr[j+1]
14         int temp = arr[j];
15         arr[j] = arr[j+1];
16         arr[j+1] = temp;
17
18         swapped = true;
19     }
20 }
21 // If no swapping happened in the current
pass, then break
22     if (swapped == false)
23         break;
24 }
25 }
26
27 static void printArray(int arr[], int n)
28 {
29     for (int i = 0; i < n; i++)
30         System.out.print(arr[i] + " ");
31     System.out.println();
32 }
33
34 public static void main(String args[])
35 {
36     int arr[] = {6, 3, 8, 9, 5};
37     int n = arr.length;
38
39     System.out.print("Given Array: ");
40     printArray(arr, n);
41
42     bubbleSort(arr, n);
43
44     System.out.print("Sorted Array: ");
45     printArray(arr, n);
46 }
47 }
48
```

Python 3

```
1 def bubbleSort(arr,n):
2     for i in range(n-1):
3         swapped = False
4         # last i elements are already at the correct
position
5         for j in range(n-i-1):
6             if(arr[j] > arr[j+1]):
7                 # Swapping elements
8                 arr[j],arr[j+1] = arr[j+1],arr[j]
9                 swapped = True
10        # If no swapping happened in the current pass,
then break
11        if not swapped:
12            break
13
14
15 def printArray(arr):
16     print(' '.join(str(x) for x in arr))
17
18 if __name__ == '__main__':
19     arr = [6, 3, 8, 9, 5]
20     n = len(arr)
21     print('Given Array:')
22     printArray(arr)
23
24     bubbleSort(arr,n)
25
26     print('Sorted Array:')
27     printArray(arr)
```

C++

```
1 #include <iostream>
2 using namespace std;
3
4 void swap(int *a, int *b)
5 {
6     int temp = *a;
7     *a = *b;
8     *b = temp;
9 }
```

```
10
11 void bubbleSort(int arr[], int n)
12 {
13     for (int i = 0; i < n-1; i++)
14     {
15         bool swapped = false;
16         // last i elements are already at the correct
position
17         for (int j = 0; j < n-i-1; j++)
18         {
19             if (arr[j] > arr[j+1])
20             {
21                 swap(&arr[j], &arr[j+1]);
22                 swapped = true;
23             }
24         }
25         // If no swapping happened in the current pass,
then break
26         if (swapped == false)
27             break;
28     }
29 }
30
31 void printArray(int arr[], int n)
32 {
33     for (int i = 0; i < n; i++)
34         cout << arr[i] << " ";
35     cout << "\n";
36 }
37
38 int main()
39 {
40     int arr[] = {6, 3, 8, 9, 5};
41     int n = sizeof(arr)/sizeof(arr[0]);
42
43     cout<<"Given Array: ";
44     printArray(arr, n);
45
```

```
46     bubbleSort(arr, n);
47
48     cout<<"Sorted Array: ";
49     printArray(arr, n);
50
51     return 0;
52 }
53
```

### Properties of Bubble Sort:

**Worst and Average Case Time Complexity:**  $O(n^2)$  ; *Worst case occurs when the array is sorted in opposite direction*

**Best Case Time Complexity:**  $O(n)$  ; *Best case occurs when the given array is already sorted*

**Space Complexity:**  $O(1)$

**In-Place Sorting Algorithm:** Yes

**Stable Sorting Algorithm:** Yes

## Video Solution

---

```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/K2YfGoP9Kw4" title="YouTube
video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
```



