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**)

```
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]
```

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.

```
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
```

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.

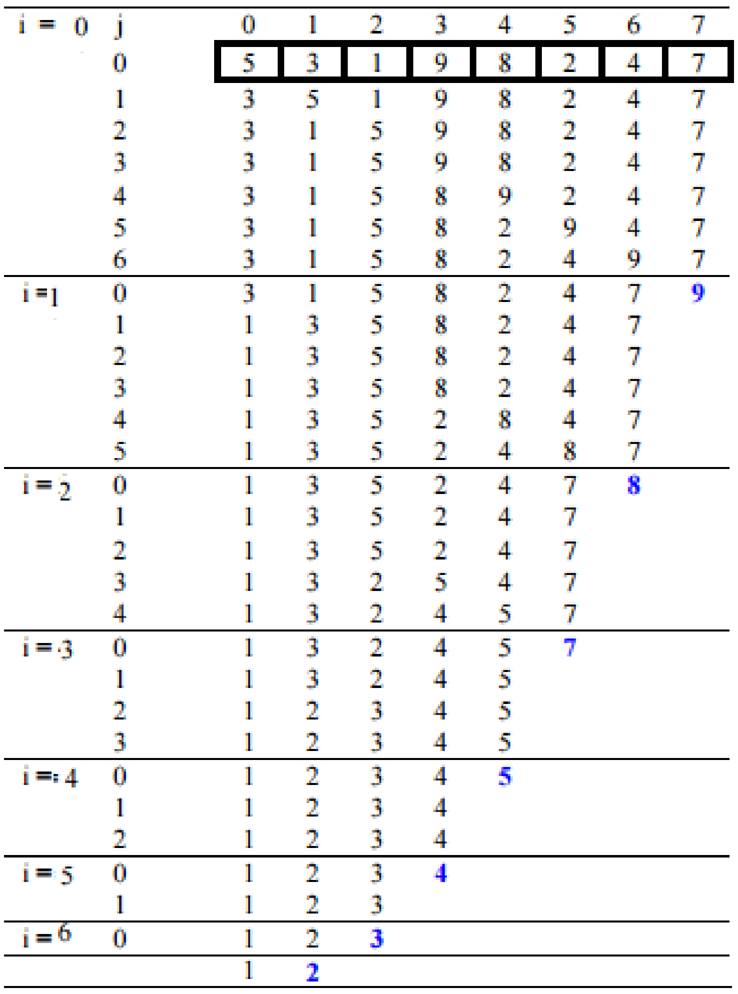
Sorting In Place: Yes

Stable: Yes

Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.

Mera

```
def bubbleSort(arr):
    n = len(arr)
    count = 0
    for badaloop in range(n-1):
        for chotaloop in range(0,n-badaloop-1):
            if arr[chotaloop]>arr[chotaloop+1]:
                arr[chotaloop] ,arr[chotaloop + 1] =
arr[chotaloop+1],arr[chotaloop]
                count = count+1
    return arr,count
```



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						