Bubble Sort

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

If the given array has to be sorted in ascending order, then bubble sort will start by comparing the first element of the array with the second element, if the first element is greater than the second element, it will **swap** both the elements, and then move on to compare the second and the third element, and so on.

If we have total n elements, then we need to repeat this process for n-1 times. We call the process a pass.

It is known as **bubble sort**, because with every complete iteration the largest element in the given array, bubbles up toward s the last place or the highest index, just like a water bubble rises up to the water surface.

Sorting takes place by stepping through all the elements one-by-one and comparing it with the adjacent element and swapping them if required.

Implementing Bubble Sort Algorithm

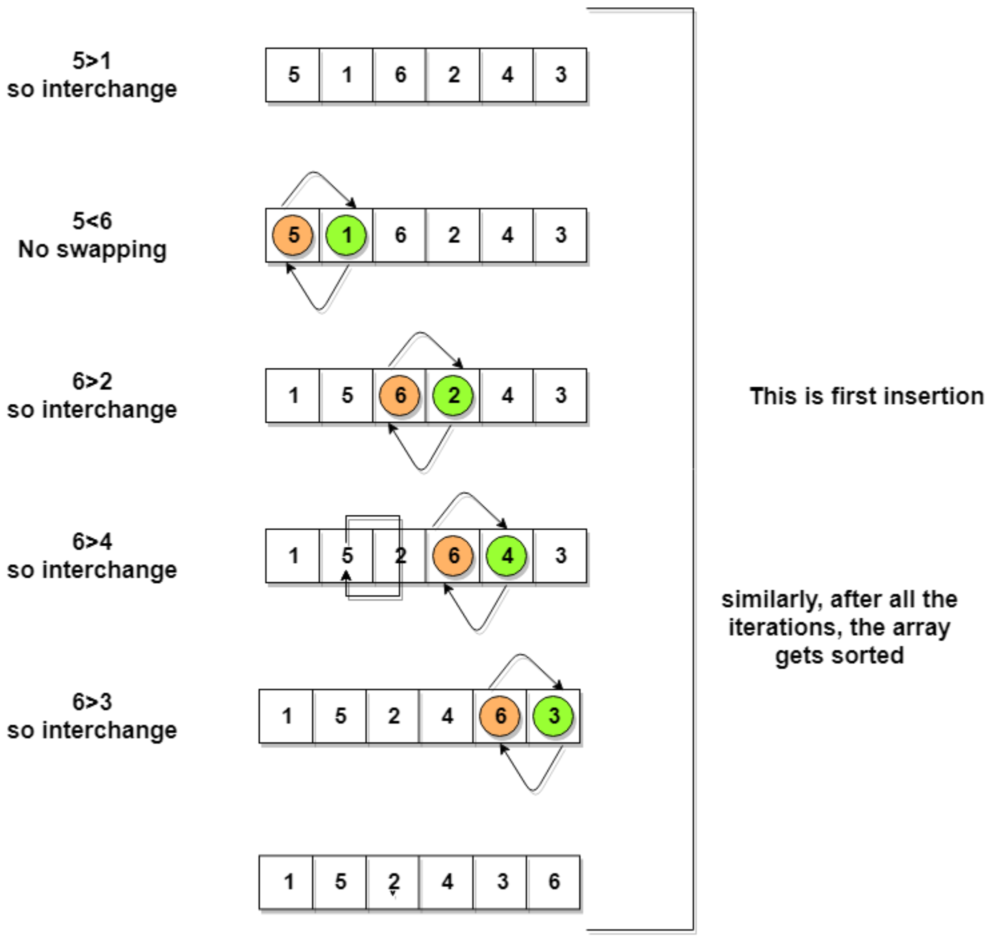
Following are the steps involved in bubble sort (for sorting a given array in ascending order):

1. Starting with the first element (index = 0), compare the current element with the next element of the array.
2. If the current element is greater than the next element of the array, swap them.
3. If the current element is less than the next element, move to the next element. **Repeat Step 1**.

Let's consider an array with values `{5, 1, 6, 2, 4, 3}`

Below, we have a pictorial representation of how bubble sort will sort the given array.

https://www.studytonight.com/data-structures/bubble-sort
https://www.geeksforgeeks.org/bubble-sort/



So as we can see in the representation above, after the first iteration, 6 is placed at the last index, which is the correct position for it.

Similarly, after the second iteration, 5 will be at the second last index, and so on.

Repeat the process as another pass. The last pass, which means the last time to compare elements, is size-1.

To illustrate the result after every pass:

Pass 1: 1, 5, 2, 4, 3, <u>6</u>

Pass 2: 1, 2, 4, 3, <u>5</u>, <u>6</u>

Pass 3: 1, 2, 3, <u>4</u>, <u>5</u>, <u>6</u>

Pass 4: 1, 2, <u>3</u>, <u>4</u>, <u>5</u>, <u>6</u>

Pass 5: <u>1</u>, <u>2</u>, <u>3</u>, <u>4</u>, <u>5</u>, <u>6</u>

https://www.studytonight.com/data-structures/bubble-sort
https://www.geeksforgeeks.org/bubble-sort/

The underlined data are the sorted data after each pass. Only the unsorted data will be compared in the succeeding passes.

Note: Array is already sorted in pass 3. But remember that the program cannot see the values, so it continues to compare until the last pass.

Another 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**)

Here is the sample code for the bubble sort algorithm:

```
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]);
}
```

https://www.studytonight.com/data-structures/bubble-sort
https://www.geeksforgeeks.org/bubble-sort/

The following are the links for the bubble sort tutorials:

1. https://youtu.be/nmhjrI-aW5o

Here is the complete output based from the sample from the video:

Given: 5 1 4 2 8 9

Pass 1: 1 4 2 5 8 9

Pass 2: 1 2 4 5 8 9

Pass 3: 1 2 4 5 8 9

Pass 4: 1 2 4 5 8 9

Pass 5: 1 2 4 5 8 9

2. https://www.youtube.com/watch?v=18OO361--1E&feature=emb_rel_pause

Here is the complete output based from the sample from the video:

Given: 30 20 40 10

Pass 1: 20 30 10 40

Pass 2: 20 10 30 40

Pass 1: 10 20 30 40

https://www.studytonight.com/data-structures/bubble-sort
https://www.geeksforgeeks.org/bubble-sort/