**BUBBLE SORT**

**MATH COMPUTATION**

Bubble sort is a simple, comparison-based sorting algorithm that repeatedly compares and swaps adjacent elements if they are in the wrong order, gradually moving the largest elements to the end of the list. When no exchanges are required, the file is sorted.

Assuming the list is an array of **n** elements and using a swap function to exchange values, bubble sort follows these steps:

1. Compare each pair of adjacent elements.

2. Swap them if the first is greater than the second.

3. Continue this process until the end of the array is reached.

4. Repeat the process from the start if the array is not sorted.

5. The result is a sorted array.

**Example**

Let us take an unsorted array: array= [25,15,30,40,5]

**0 1 2 3 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **25** | **15** | **30** | **40** | **5** |

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

**0 1 2 3 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **25** | **15** | **30** | **40** | **5** |

we will swap the first two elements because **25** is greater than **15**, the array will be:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **30** | **40** | **5** |

we will compare again the element at index 1 and 2, which are **25** and **30**, in this case no need to swap because **30** is greater than **25**, so it is already in sorted locations.

The array will be:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **30** | **40** | **5** |

Next, we compare 30 and 40. We find that both are in already sorted positions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **30** | **40** | **5** |

We move to the next two values, 40 and 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **30** | **40** | **5** |

We know then that 5 is smaller 40. Hence, they are not sorted. We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **30** | **5** | **40** |

we are now showing how an array should look like after each iteration. After the second iteration, it should look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **30** | **5** | **40** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **25** | **5** | **30** | **40** |

We will repeat step 1 to 4 and notice after each iteration, at least one value moves at the end.

Third iteration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **15** | **5** | **25** | **30** | **40** |

Four iteration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **15** | **25** | **30** | **40** |

Bubble sort iterates through an array, swapping adjacent elements if they are in the wrong order. When a full pass is made without any swaps, the algorithm determines that the array is completely sorted.

The sorted array is:

**0 1 2 4 5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **15** | **25** | **30** | **40** |