Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort algorithm. In each step, elements written in **bold** are being compared.

**First Pass:**  
( **5** **1** 4 2 8 ) {\displaystyle \to } ( **1** **5** 4 2 8 ), Here, algorithm compares the first two elements, and swaps them since 5 > 1  
( 1 **5** **4** 2 8 ) {\displaystyle \to } ( 1 **4** **5** 2 8 ), It then compares the second and third items and swaps them since 5 > 4  
( 1 4 **5** **2** 8 ) {\displaystyle \to } ( 1 4 **2** **5** 8 ), Swap since 5 > 2  
( 1 4 2 **5** **8** ) {\displaystyle \to } ( 1 4 2 **5** **8** ), Now, since these elements are already in order (8 > 5), algorithm does not swap them.  
The algorithm has reached the end of the list of numbers and the largest number, 8, has bubbled to the top. It now starts again.  
  
**Second Pass:**  
( **1** **4** 2 5 8 ) {\displaystyle \to } ( **1** **4** 2 5 8 ), no swap needed  
( 1 **4** **2** 5 8 ) {\displaystyle \to } ( 1 **2** **4** 5 8 ), Swap since 4 > 2  
( 1 2 **4** **5** 8 ) {\displaystyle \to } ( 1 2 **4** **5** 8 ), no swap needed  
( 1 2 4 **5** **8** ) {\displaystyle \to } ( 1 2 4 **5** **8** ), no swap needed  
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 ) {\displaystyle \to } ( **1** **2** 4 5 8 )  
( 1 **2** **4** 5 8 ) {\displaystyle \to } ( 1 **2** **4** 5 8 )  
( 1 2 **4** **5** 8 ) {\displaystyle \to } ( 1 2 **4** **5** 8 )  
( 1 2 4 **5** **8** ) {\displaystyle \to } ( 1 2 4 **5** **8** )  
Finally, the array is sorted, and the algorithm can terminate.

**Pseudocode implementation**[[edit](https://en.wikibooks.org/w/index.php?title=A-level_Computing/AQA/Paper_1/Fundamentals_of_algorithms/Sorting_algorithms&action=edit&section=3" \o "Edit section: Pseudocode implementation)]

The algorithm can be expressed as:

**procedure** bubbleSort( A **:** list of sortable items )

**do**

swapped = false

**for each** i **in** 1 **to** length(A) - 1 **inclusive do:**

**if** A[i-1] > A[i] **then**

swap( A[i-1], A[i] )

swapped = true

**end if**

**end for**

**while** swapped

**end procedure**