**Bubble Sort**

The algorithm maintains **two subarrays** in a given array.

1. The subarray which is already **sorted**.
2. Remaining subarray which is **unsorted**.

* The **bubble sort** compares adjacent items and exchanges them if they are out of order.
* In the worst case, this algorithm makes n2 passes over the data.
* During the first pass, you compare the first two items in the array.

If they are out of order, you swap them.

( **5 1** 4 2 8 ) –> ( **1 5** 4 2 8 )

Here, algorithm compares the first two elements, and swaps since 5 > 1.

* You then compare the items in the next pair—that is, in positions 2 and 3 of the array.

If they are out of order, you swap them.

( 1 **5 4** 2 8 ) –> ( 1 **4 5** 2 8 )

Swap since 5 > 4

* Compare positions 3 and 4 of the array.

( 1 4 **5 2** 8 ) –> ( 1 4 **2 5** 8 )

Swap since 5 > 2

* Compare positions 4 and 5 of the array.

( 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.

* The array now is broken up into two subarrays: one part unsorted, and one part sorted.

Blue = sorted and Gray = unsorted

( 1 4 2 5 8 )

* You iterate over the array again, swapping elements until you reach the end of the unsorted array (you do not need to compare to 8 because it is already in its sorted position).
* Figure 11-2a illustrates the first pass of a bubble sort of an array of five integers.
  + You compare the items in the first pair—29 and 10—and exchange them because they are out of order.
  + Next you con- sider the second pair—29 and 14—and exchange these items because they are out of order.
  + The items in the third pair—29 and 37—are in order, and so you do not exchange them.
  + Finally, you exchange the items in the last pair—37 and 13.
* Although the array is not sorted after the first pass, the largest item has “bubbled” to its proper position at the end of the array.
* During the second pass of the bubble sort, you return to the beginning of the array and consider pairs of items in exactly the same manner as the first pass.
* You do not, however, include the last—and largest—item of the array.
* That is, the second pass considers the first n – 1 items of the array.
* The second pass considers the first n – 2 items of the array. After the second pass, the second-largest item in the array will be in its proper place in the next-to-last position of the array, as Figure 11-2b illustrates.
* Now, ignoring the last two items, which are in order, you continue with subsequent passes until the array is sorted.
* Although a bubble sort requires at most n – 1 passes to sort the array, fewer passes might be possible to sort a particular array.
* Thus, you could terminate the process if no exchanges occur during any pass.

Table

Description automatically generated