Bubble Sort

Khalid Hourani

February 23, 2017

The **Bubble Sort** is a simple sort on an array with following structure:

1. Iterate through the array.
2. If an element and the element that follows are out of order, swap them.
3. After iterating through the array, if a swap has been performed, return to step 1. Otherwise terminate.

Note that, after i iterations through the array, the rightmost i elements are in the correct position. Thus, on the i^{th} iteration, we need only compare the first $n - i + 1$ elements of the array.

So, for example, the array [8, 6, 7, 5, 3, 0, 9] is sorted as follows:

1.

$$\begin{aligned}
&[8,6,7,5,3,0,9] \xrightarrow{\text{iterate}} [\underline{8,6},7,5,3,0,9] \xrightarrow{\text{swap}} [\underline{6,8},7,5,3,0,9]\\
&\xrightarrow{\text{iterate}} [6,\underline{8,7},5,3,0,9] \xrightarrow{\text{swap}} [6,\underline{7,8},5,3,0,9] \xrightarrow{\text{iterate}} [6,7,\underline{8,5},3,0,9]\\
&\xrightarrow{\text{swap}} [6,7,\underline{5,8},3,0,9] \xrightarrow{\text{iterate}} [6,7,5,\underline{8,3},0,9] \xrightarrow{\text{swap}} [6,7,5,\underline{3,8},0,9]\\
&\xrightarrow{\text{iterate}} [6,7,5,3,\underline{8,0},9] \xrightarrow{\text{swap}} [6,7,5,3,\underline{0,8},9] \xrightarrow{\text{iterate}} [6,7,5,3,0,\underline{8,9}]
\end{aligned}$$

2.

$$\begin{aligned}
&[6,7,5,3,0,8,9] \xrightarrow{\text{iterate}} [\underline{6,7},5,4,0,8,9] \xrightarrow{\text{iterate}} [6,\underline{7,5},3,0,8,9]\\
&\xrightarrow{\text{swap}} [6,\underline{5,7},3,0,8,9] \xrightarrow{\text{iterate}} [6,5,\underline{7,3},0,8,9] \xrightarrow{\text{swap}} [6,5,\underline{3,7},0,8,9]\\
&\xrightarrow{\text{iterate}} [6,5,3,\underline{7,0},8,9] \xrightarrow{\text{swap}} [6,5,3,\underline{0,7},8,9] \xrightarrow{\text{iterate}} [6,5,3,0,\underline{7,8},9]
\end{aligned}$$

3.

$$\begin{aligned}
&[6,5,4,0,7,8,9] \xrightarrow{\text{iterate}} [\underline{6,5},3,0,7,8,9] \xrightarrow{\text{swap}} [\underline{5,6},3,0,7,8,9]\\
&\xrightarrow{\text{iterate}} [5,\underline{6,3},0,7,8,9] \xrightarrow{\text{swap}} [5,\underline{3,6},0,7,8,9] \xrightarrow{\text{iterate}} [5,3,\underline{6,0},7,8,9]\\
&\xrightarrow{\text{swap}} [5,3,\underline{0,6},7,8,9] \xrightarrow{\text{iterate}} [5,3,0,\underline{6,7},8,9]
\end{aligned}$$

4.

$$\begin{aligned}
&[5,3,0,6,7,8,9] \xrightarrow{\text{iterate}} [\underline{5,3},0,6,7,8,9] \xrightarrow{\text{swap}} [\underline{3,5},0,6,7,8,9]\\
&\xrightarrow{\text{iterate}} [3,\underline{5,0},6,7,8,9] \xrightarrow{\text{swap}} [3,\underline{0,5},6,7,8,9] \xrightarrow{\text{iterate}} [3,0,\underline{5,6},7,8,9]
\end{aligned}$$

5.

$$\begin{aligned}
&[3,0,5,6,7,8,9] \xrightarrow{\text{iterate}} [\underline{3,0},5,6,7,8,9] \xrightarrow{\text{swap}} [\underline{0,3},5,6,7,8,9]\\
&\xrightarrow{\text{iterate}} [0,\underline{3,5},6,7,8,9]
\end{aligned}$$

Thus, after a rather lengthy process, iterating through the array of 7 elements a total of 6 times, we get the sorted list: [0, 3, 5, 6, 7, 8, 9].

Now, suppose we perform the sort on a list of n elements. The best case scenario is when the list is already sorted, in which case one pass through the list must be made with n comparisons. Thus, the Bubble Sort is $\Omega(n)$. The worst case, however, is when the list is in reverse order. In this case, the Bubble Sort will require $n - i$ comparisons for the i^{th} pass of n passes through the array, or

$$\sum_{i=0}^{n} n - i = \sum_{i=1}^{n} i = \frac{n^2 + n}{2}$$

total comparisons. Thus, the Bubble Sort is $O(n^2)$.

The following C++ code demonstrates the Bubble Sort by taking an integer array and its length as input and sorting the array:

```
void bubble_sort(int* list, int n)
{
        bool flag = true; //flag to check if swap has been made
        int i = 0; //counter to keep track of number of passes through the array
        while (flag)
        {
                i++;
                flag = false; //reset flag at start of each pass through array
                for (int j = 0; j < n - i; j++)
                {
                        if (list[j] > list[j + 1]) //compare consecutive elements
                        {
                                swap(list[j], list[j + 1]);
                                flag = true; //swap and set flag if elements are out of order
                        }
                }
        }
}
```