Sorting Algorithm – Bubble Sort

Bubble sort builds a sorted list by continuously examining through the unsorted list, comparing adjacent values and exchange them if they are not in order. The algorithm is named by the way the largest (or smallest) value "bubbles" to the top of the unsorted list after each pass. When a pass requires no swapping, it indicates the list is sorted and the algorithm ends. Therefore, unlike insertion sort and selection sort, bubble sort can implicitly detect if the list is sorted within each pass and does not continue if the list is in order.

Suppose the following list is to be sorted in ascending order. The algorithm starts by comparing 25 to 9.

25	9	11	2	31	23

Since they are not in order, they are swapped, with the result

9	25	11	2	31	23

Next 25 is compared to 11.

9	25	11	2	31	23

Since they are not in order, they are exchanged. Then, 25 is compared to 2.

9	11	25	2	31	23

Again, they are swapped since they are not in order. 25 is then compared to 31.

9	11	2	25	31	23

Since they are in order, they remain in the same position. Lastly, 31 is compared to 23,

9	11	2	25	31	23

and since 31 is larger than 23, their positions need to be switched.

At the end of the first pass, with all the comparing and exchanging, the largest value 31 is at the top of the list (like a bubble in a liquid rising to the top). The top one item is in the correct position (a sorted list of size 1).

9	11	2	25	23	31

The algorithm continues the same process on the remaining list (excluding the last element). As it proceeds, the size of the sorted list grows and each pass deals with a shorter subsequence of the original list until all values are in the correct positions.

Second pass:

9	11	2	25	23	31
9	11	2	25	23	31
9	2	11	25	23	31
9	2	11	25	23	31
9	2	11	23	25	31

`

Third pass:

9	2	11	23	25	31
2	9	11	23	25	31
2	9	11	23	25	31
2	9	11	23	25	31

Fourth pass:

2	9	11	23	25	31
2	9	11	23	25	31
2	9	11	23	25	31

During the fourth pass, no swapping takes place, which implies the list is already sorted. The algorithm ends, and no further pass is needed.

Implementation

The implementation of bubble sort continuously compares adjacent elements and swaps them, if necessary, in a list of size starting `L.length`, then sublists of sizes `list.length-1`, `list.length-2`, etc,. The outter loop loops through the upper bound of the sublists and inner loop compare adjacent elements and swap them if necessary. A boolean is used to keep track if swapping occurs in a pass. No swapping implies the list is sorted and the outter loops terminates.

```
bubbleSort(List L)
   for upperbound equals L.length-1 down to 1 and sorted is false
      set sorted to true
      for j from 0 to upperbound - 1
         if element at index j is greater than element at index j+1
            set sorted to false
            swap element at index j and element at index j+1
```