Bubble Sort and Modified Bubble Sort

Have an array you need to put in order? Keeping business records and want to sort them by ID number or last name of client? Then you'll need a sorting algorithm. To understand the more complex and efficient sorting algorithms, it's important to first understand the simpler, but slower algorithms. In this article, you'll learn about bubble sort, including a modified bubble sort that's slightly more efficient; insertion sort; and selection sort. Any of these sorting algorithms are good enough for most small tasks, though if you were going to process a large amount of data, you would want to choose one of the sorting algorithms listed on the advanced sorting page.

Bubble sort

The simplest sorting algorithm is bubble sort. The bubble sort works by iterating down an array to be sorted from the first element to the last, comparing each pair of elements and switching their positions if necessary. This process is repeated as many times as necessary, until the array is sorted. Since the worst case scenario is that the array is in reverse order, and that the first element in sorted array is the last element in the starting array, the most exchanges that will be necessary is equal to the length of the array. Here is a simple example:

Given an array 23154 a bubble sort would lead to the following sequence of partially sorted arrays: 21354, 21345, 12345. First the 1 and 3 would be compared and switched, then the 4 and 5. On the next pass, the 1 and 2 would switch, and the array would be in order.

The basic code for bubble sort looks like this, for sorting an integer array:

```
        for(int x=0; x<n; x++)

        {

                for(int y=0; y<n-1; y++)

                {

                        if(array[y]>array[y+1])

                        {

                                int temp = array[y+1];

                                array[y+1] = array[y];

                                array[y] = temp;
```

```
                    }

                }

        }
```

Notice that this will always loop n times from 0 to n, so the order of this algorithm is O(n^2). This is both the best and worst case scenario because the code contains no way of determining if the array is already in order.

A better version of bubble sort, known as modified bubble sort, includes a flag that is set if an exchange is made after an entire pass over the array. If no exchange is made, then it should be clear that the array is already in order because no two elements need to be switched. In that case, the sort should end. The new best case order for this algorithm is O(n), as if the array is already sorted, then no exchanges are made. You can figure out the code yourself! It only requires a few changes to the original bubble sort.

Part 2: Selection Sort and Insertion Sort