Chapter 29: Bubble Sort

Parameter	Description
Stable	Yes
In place	Yes
Best case complexity	O(n)
Average case complexity	O(n^2)
Worst case complexity	O(n^2)
Space complexity	O(1)

Section 29.1: Bubble Sort

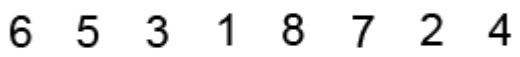
The `BubbleSort` compares each successive pair of elements in an unordered list and inverts the elements if they are not in order.

The following example illustrates the bubble sort on the list `{6,5,3,1,8,7,2,4}` (pairs that were compared in each step are encapsulated in '**'):

```
{6,5,3,1,8,7,2,4}
{**5,6**,3,1,8,7,2,4} -- 5 < 6 -> swap
{5,**3,6**,1,8,7,2,4} -- 3 < 6 -> swap
{5,3,**1,6**,8,7,2,4} -- 1 < 6 -> swap
{5,3,1,**6,8**,7,2,4} -- 8 > 6 -> no swap
{5,3,1,6,**7,8**,2,4} -- 7 < 8 -> swap
{5,3,1,6,7,**2,8**,4} -- 2 < 8 -> swap
{5,3,1,6,7,2,**4,8**} -- 4 < 8 -> swap
```

After one iteration through the list, we have `{5,3,1,6,7,2,4,8}`. Note that the greatest unsorted value in the array (8 in this case) will always reach its final position. Thus, to be sure the list is sorted we must iterate n-1 times for lists of length n.

Graphic:



Section 29.2: Implementation in C & C++

An example implementation of `BubbleSort` in `C++`:

```
void bubbleSort(vector<int>numbers)
{
    for(int i = numbers.size() - 1; i >= 0; i--) {
        for(int j = 1; j <= i; j++) {
            if(numbers[j-1] > numbers[j]) {
                swap(numbers[j-1],numbers(j));
```

```
            }
        }
    }
}
```

C Implementation

```
void bubble_sort(long list[], long n)
{
  long c, d, t;

  for (c = 0 ; c < ( n - 1 ); c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (list[d] > list[d+1])
      {
        /* Swapping */

        t         = list[d];
        list[d]   = list[d+1];
        list[d+1] = t;
      }
    }
  }
}
```

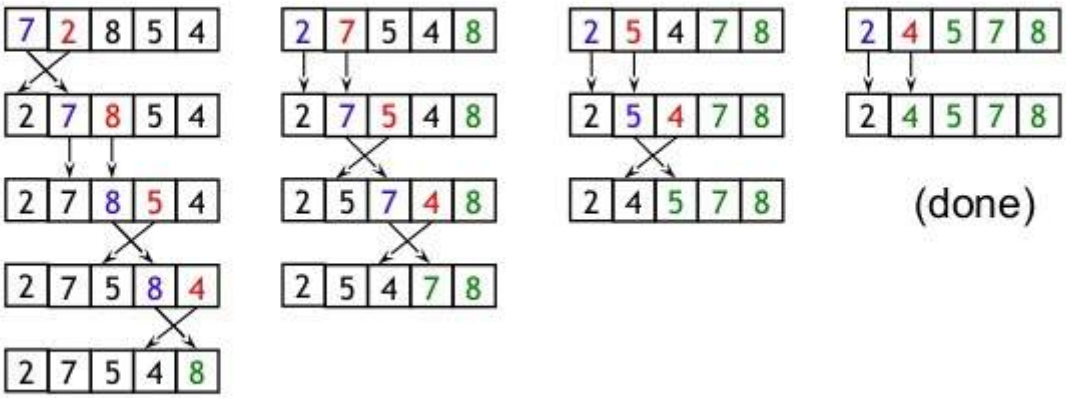
Bubble Sort with pointer

```
void pointer_bubble_sort(long * list, long n)
{
  long c, d, t;

  for (c = 0 ; c < ( n - 1 ); c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if ( * (list + d ) > *(list+d+1))
      {
        /* Swapping */

        t         = * (list + d );
        * (list + d )   = * (list + d + 1 );
        * (list + d + 1) = t;
      }
    }
  }
}
```

Section 29.3: Implementation in C#

Bubble sort is also known as **Sinking Sort**. It is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order.

Bubble sort example



Implementation of Bubble Sort

I used C# language to implement bubble sort algorithm

```
public class BubbleSort
{
    public static void SortBubble(int[] input)
    {
        for (var i = input.Length - 1; i >= 0; i--)
        {
            for (var j = input.Length - 1 - 1; j >= 0; j--)
            {
                if (input[j] <= input[j + 1]) continue;
                var temp = input[j + 1];
                input[j + 1] = input[j];
                input[j] = temp;
            }
        }
    }

    public static int[] Main(int[] input)
    {
        SortBubble(input);
        return input;
    }
}
```

Section 29.4: Python Implementation

```
#!/usr/bin/python

input_list = [10,1,2,11]

for i in range(len(input_list)):
  for j in range(i):
    if int(input_list[j]) > int(input_list[j+1]):
      input_list[j],input_list[j+1] = input_list[j+1],input_list[j]

print input_list
```

Section 29.5: Implementation in Java

```
public class MyBubbleSort {

    public static void bubble_srt(int array[]) {//main logic
        int n = array.length;
        int k;
        for (int m = n; m >= 0; m--) {
            for (int i = 0; i < n - 1; i++) {
                k = i + 1;
                if (array[i] > array[k]) {
                    swapNumbers(i, k, array);
                }
            }
            printNumbers(array);
        }
    }

    private static void swapNumbers(int i, int j, int[] array) {

        int temp;
        temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    private static void printNumbers(int[] input) {

        for (int i = 0; i < input.length; i++) {
            System.out.print(input[i] + ", ");
        }
        System.out.println("\n");
    }

    public static void main(String[] args) {
        int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };
        bubble_srt(input);

    }
}
```

Section 29.6: Implementation in Javascript

```
        function bubbleSort(a)
          {
                var swapped;
                do {
                    swapped = false;
                    for (var i=0; i < a.length-1; i++) {
                        if (a[i] > a[i+1]) {
                            var temp = a[i];
                            a[i] = a[i+1];
                            a[i+1] = temp;
                            swapped = true;
                        }
                    }
                } while (swapped);
            }

   var a = [3, 203, 34, 746, 200, 984, 198, 764, 9];
```

```
bubbleSort(a);
console.log(a); //logs [ 3, 9, 34, 198, 200, 203, 746, 764, 984 ]
```