Sorting And Searching Algorithm

What is Sorting Algorithm?

Sorting is a process of ordering or placing a list of elements from a collection in some kind of order. It can be done in ascending and descending order. It arranges the data in a sequence which makes searching easier.

Sorting Algorithm is used to reorganize a given array or list elements using a comparison operator.

Main Application of Sorting Algorithm

- Finding the median of a set of keys
- Organizing items by price on a retail website
- Determining the order of sites on a search engine.
- In library, keep the books organized
- Arranging numbers in ascending and descending order

Sorting Process

- As humans, we can do this by lining up the items
- Computers not like humans can visually compare things two items at once
- It uses two steps to execute over and over until data are sorted

a. Compare the two items
b. Swap the two items or copy one item but still; the details are handled in different ways.

Main Types of Sorting Algorithm

1. **Bubble Sort**
2. **Section Sort**
3. **Insertion Sort**

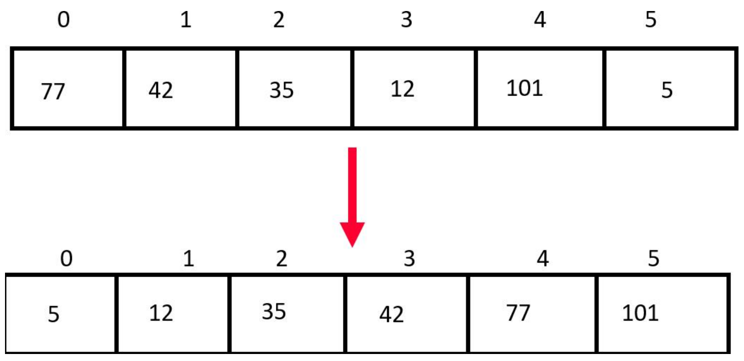
Bubble Sort Algorithm

The simplest in sorting process. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared, and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst-case complexity are of O(n2) where n is the number of items.

How bubble sort Works?

We take an unsorted array for our example. Bubble sort takes O(n2) time so we're keeping it short and precise.

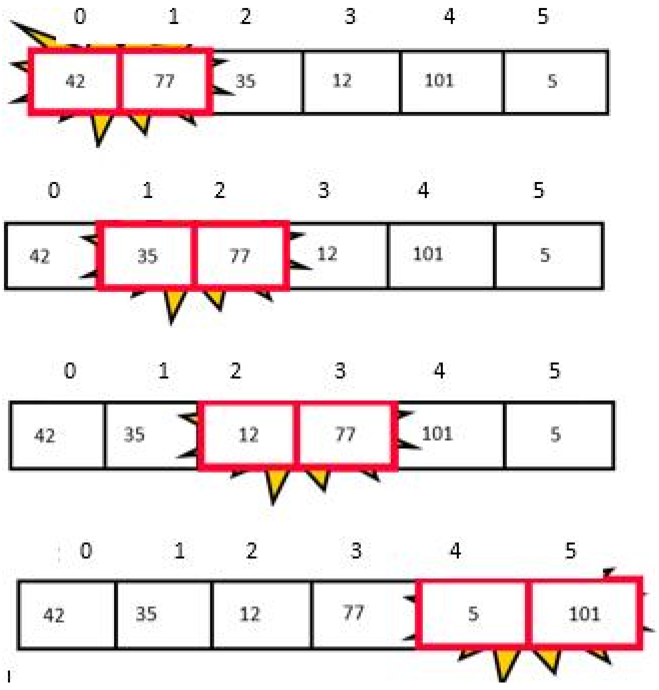
Sorting takes an unordered collection and makes it an ordered one.

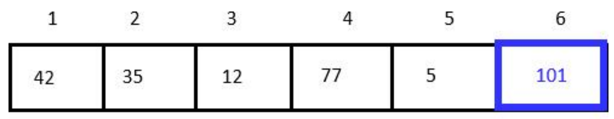


Bubbling Up" the Largest Element

- Traverse a collection of elements
- Move from the front to the end
- “Bubble” the **largest value** to the end using **pair-wise comparisons**

and **swapping**





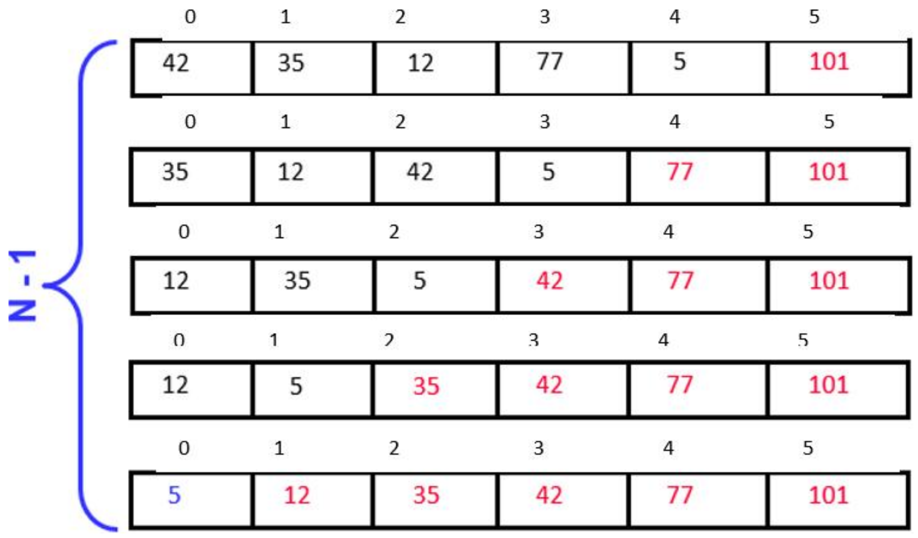
Largest value correctly placed

- Notice that only the largest value is correctly placed
- All other values are still out of order
- So, we need to repeat this process

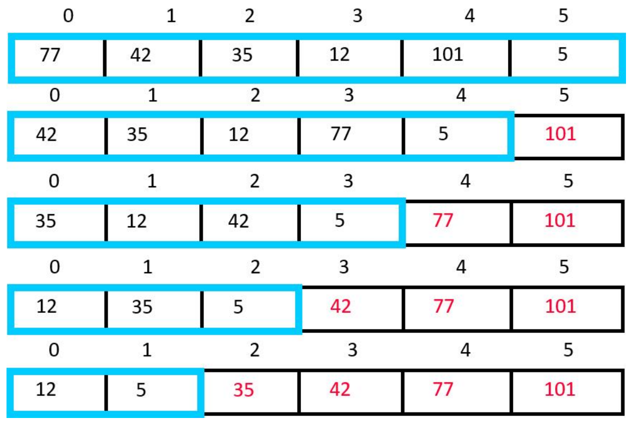
Repeat “Bubble Up” How Many Times?

- If we have N elements…
- And if each time we bubble an element, we place it in its correct location…
- Then we repeat the “bubble up” process N – 1 times.
- This guarantees we’ll correctly place all N elements.

Bubbling all the Elements



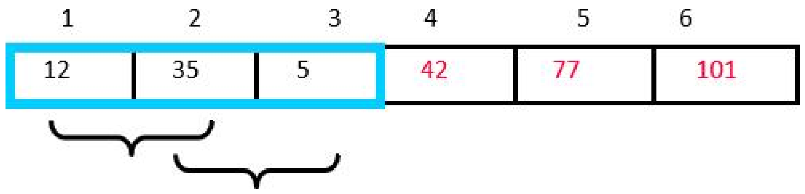
Reducing the number of comparisons



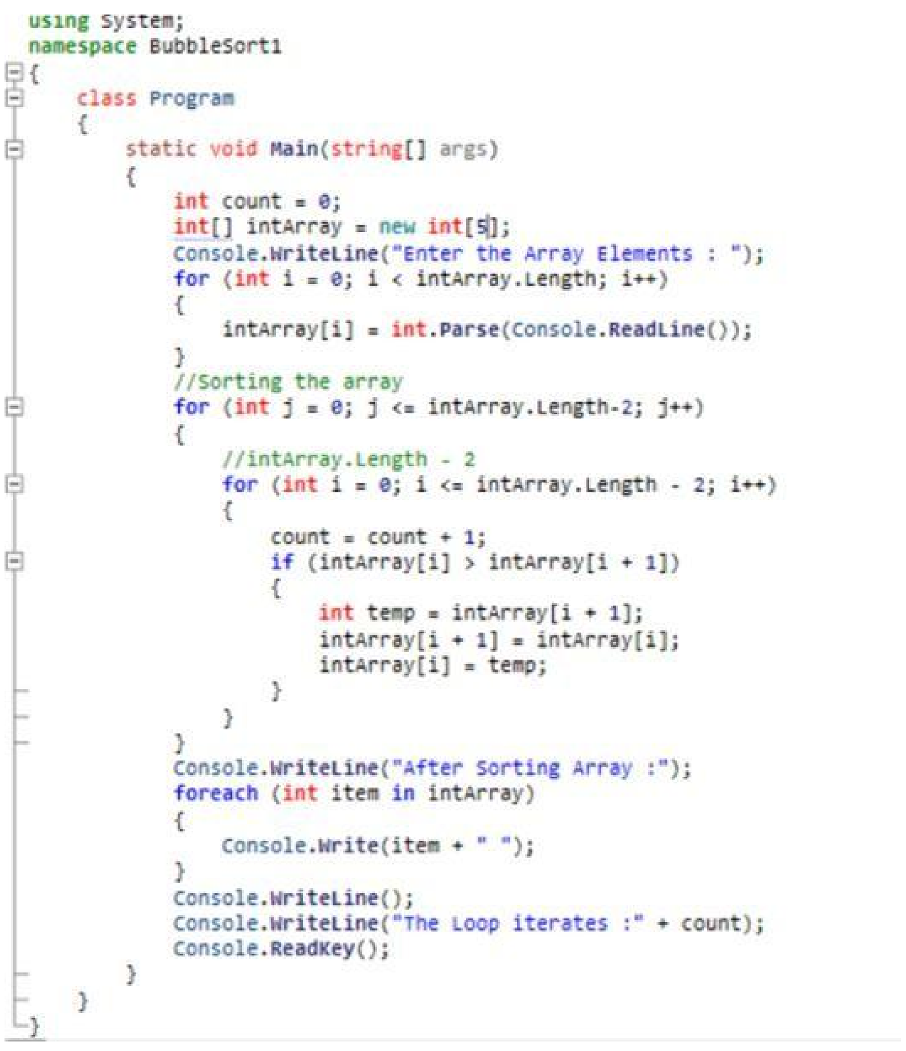
Reducing the number of comparisons

- On the Nth “bubble up”, we only need to do MAX-N comparisons.
- For example:
    - This is the 4th “bubble up”
    - MAX is 6

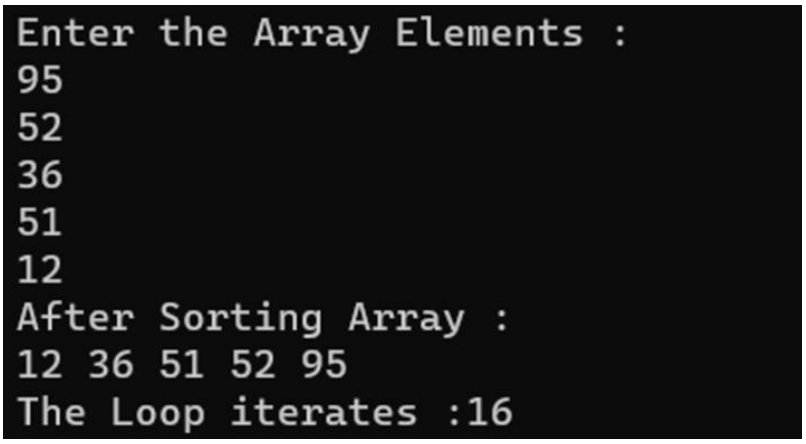
Thus we have 2 comparisons to do



Implementing the bubble sort algorithm in C#



```
using System;
namespace BubbleSort1
{
    class Program
    {
        static void Main(string[] args)
        {
            int count = 0;
            int[] intArray = new int[5];
            Console.WriteLine("Enter the Array Elements : ");
            for (int i = 0; i < intArray.Length; i++)
            {
                intArray[i] = int.Parse(Console.ReadLine());
            }
            //Sorting the array
            for (int j = 0; j <= intArray.Length-2; j++)
            {
                //intArray.Length - 2
                for (int i = 0; i <= intArray.Length - 2; i++)
                {
                    count = count + 1;
                    if (intArray[i] > intArray[i + 1])
                    {
                        int temp = intArray[i + 1];
                        intArray[i + 1] = intArray[i];
                        intArray[i] = temp;
                    }
                }
            }
            Console.WriteLine("After Sorting Array :");
            foreach (int item in intArray)
            {
                Console.Write(item + " ");
            }
            Console.WriteLine();
            Console.WriteLine("The Loop iterates :" + count);
            Console.ReadKey();
        }
    }
}
```



```
Enter the Array Elements :
95
52
36
51
12
After Sorting Array :
12 36 51 52 95
The Loop iterates :16
```

Summary

- “Bubble Up” algorithm will move largest value to its correct location (to the right)

- Repeat “Bubble Up” until all elements are correctly placed: Maximum of N-1 times. Can finish early if no swapping occurs
- We reduce the number of elements we compare each time one is correctly placed