Selection Sort Algorithm

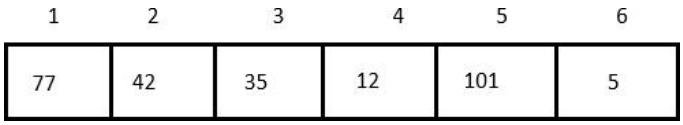
Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

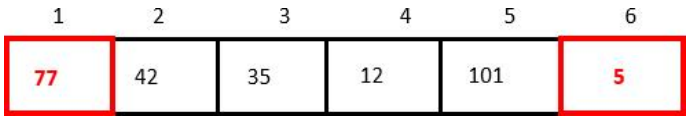
This algorithm is not suitable for large data sets as its average and worst case complexities are of O(n2), where n is the number of items.

How Selection Sort Works?

a. Find the minimum value in the list.
b. Swap it with the value in the **first position.**
c. Repeat the steps for the remainder of the list starting the **next position after the swapping to the first position.**



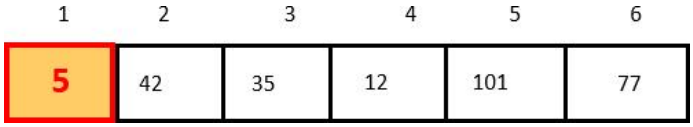
Consider the following depicted array as an example.



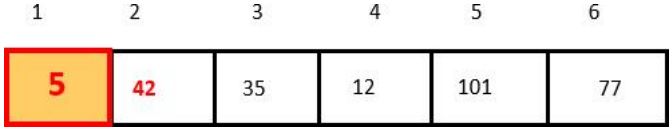
For the first position in the sorted list, the whole list is scanned sequentially. The first position where 77 is stored presently, we search the whole list and find that 5 is the lowest value.

0 1 2 3 4 5

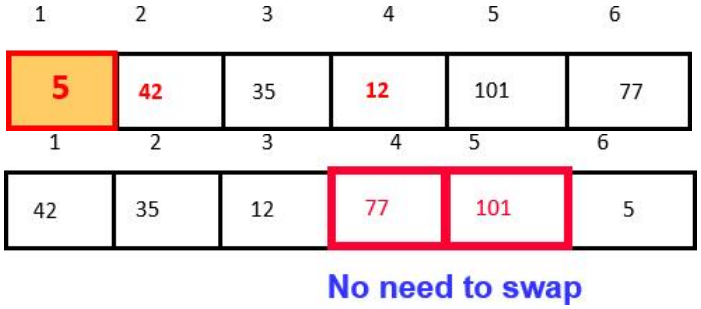
So we replace 77 with 5. After one iteration 5, which happens to be the minimum value in the list, appears in the first position of the sorted list.



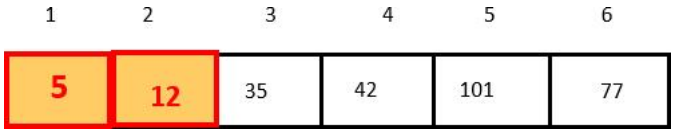
For the second position, where 42 is residing, we start scanning the rest of the list in a linear manner.



We find that 12 is the second lowest value in the list and it should appear at the second place. We swap these values.

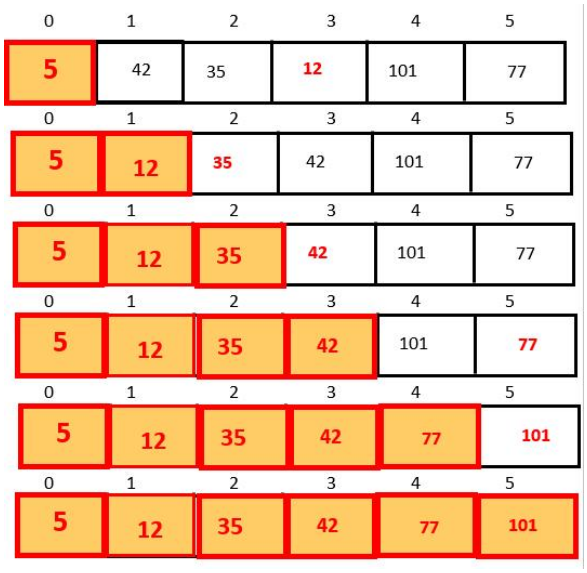


After two iterations, two least values are positioned at the beginning in a sorted manner.



The same process is applied to the rest of the items in the array. Following

is a pictorial depiction of the entire sorting process –



Implementing the Insertion sort algorithm in C#

```
using System;

class SelectionSort
{
    static void Sort (int []arr)
    {
        int n = arr.Length;

        // One by one move boundary of unsorted subarray
        for (int i = 0; i < n - 1; i++)
        {
            // Find the minimum element in unsorted array
            int min_idx = i;
            for (int j = i + 1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            // Swap the found minimum element with the first
            // element
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }

    // Prints the array
    static void printArray(int []arr)
    {
        int n = arr.Length;
        for (int i=0; i<n; ++i)
            Console.Write(arr[i]+" ");
        Console.WriteLine();
    }

    // Driver code
    public static void Main()
    {
        int []arr = {64,25,12,22,11};
        Sort(arr);
        Console.WriteLine("Sorted array");
        printArray(arr);
        Console.ReadKey();
    }
}
```

```
Sorted array
11 12 22 25 64
```