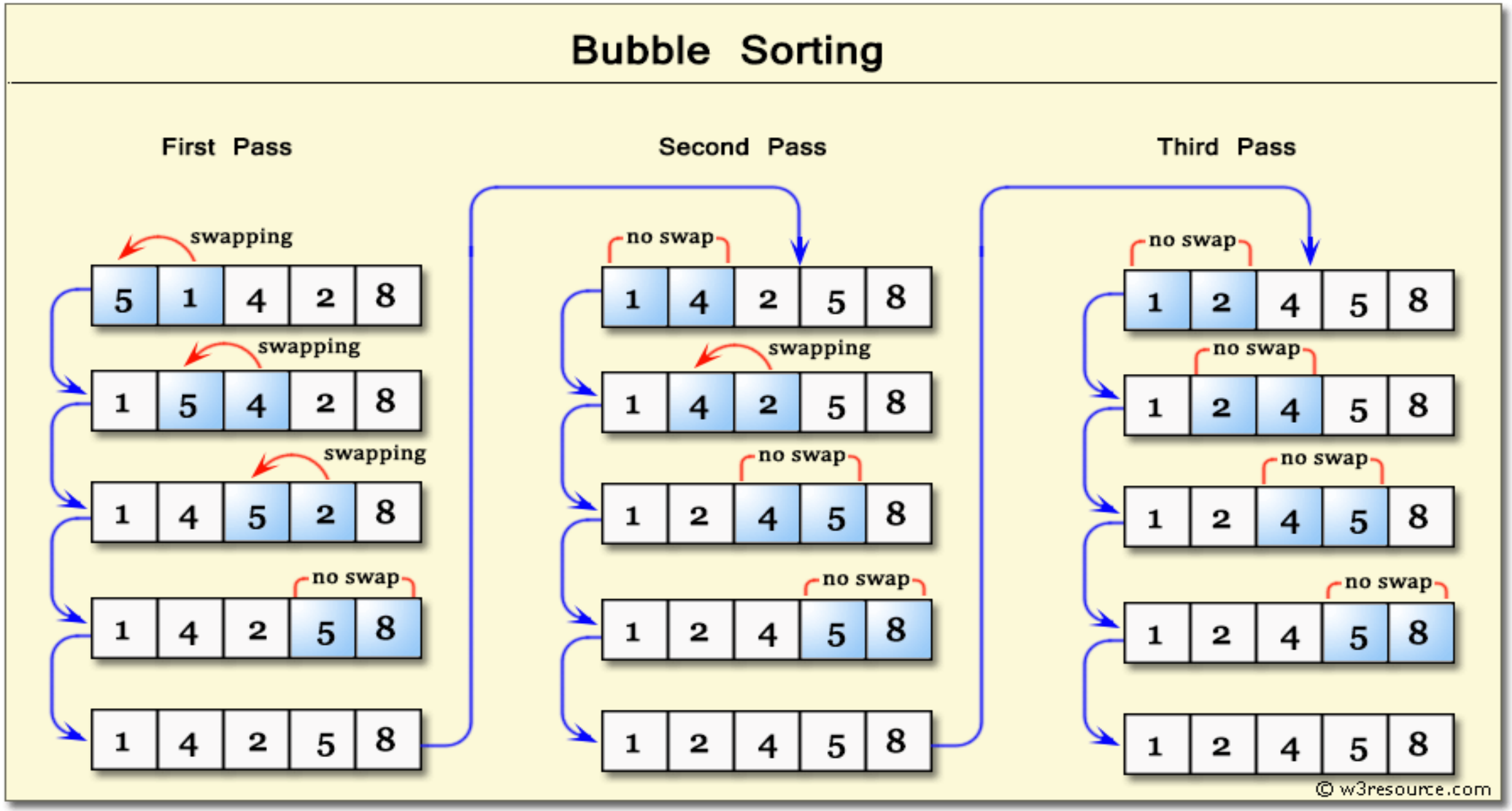
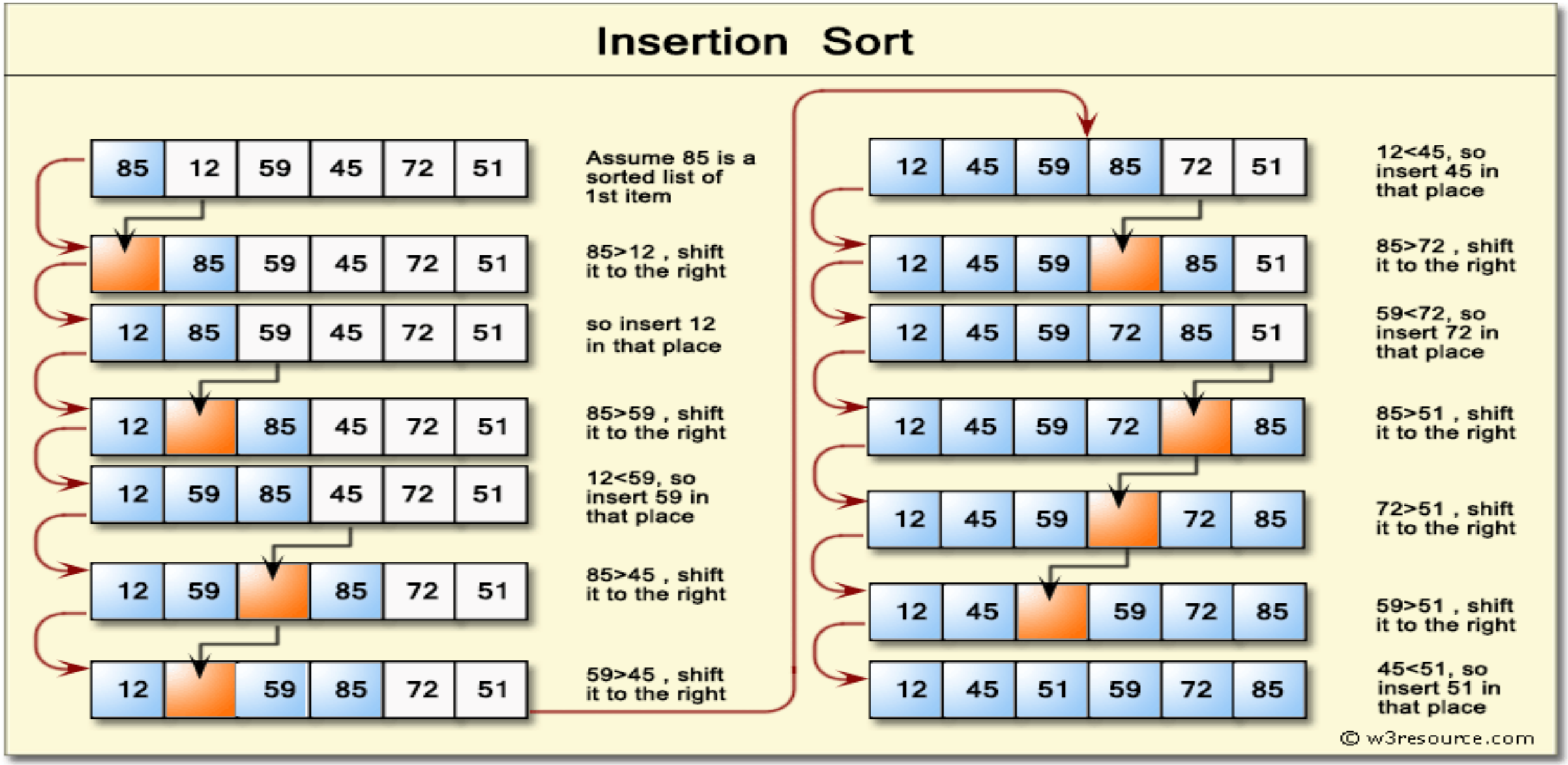
Selection Sort

```
{
    0 references
    static void Main(string[] args)
    {
        int[] arr = { 5, 7, 1, 19, 25, 4, 80, 2 };
        int temp;
        for (int i = 0; i < arr.Length - 1; i++)
        {
            for (int j = i + 1; j < arr.Length; j++)
            {
                //massivin birinci ədədi ilə ikinci ədədi müqayise edirik və
                //əgər birinci ədəd ikinci edədən böyükdürsə birinci ədəd ikinci
                //ədədin yerinə ikinci ədəd birinci ədədin yerinə yazırıq və
                //dövr davam etdikcə massivin bütün elementləri bu üsulla
                //yoxlanaraq sıra ilə düzülür
                if (arr[i] > arr[j])
                {
                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
    }
```


Bubble Sorting
First Pass
Second Pass
Third Pass
swapping
swapping
swapping
no swap
no swap
no swap
no swap
no swap
no swap
5 1 4 2 8
1 5 4 2 8
1 4 5 2 8
1 4 2 5 8
1 4 2 5 8
1 4 2 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
1 2 4 5 8
© w3resource.com

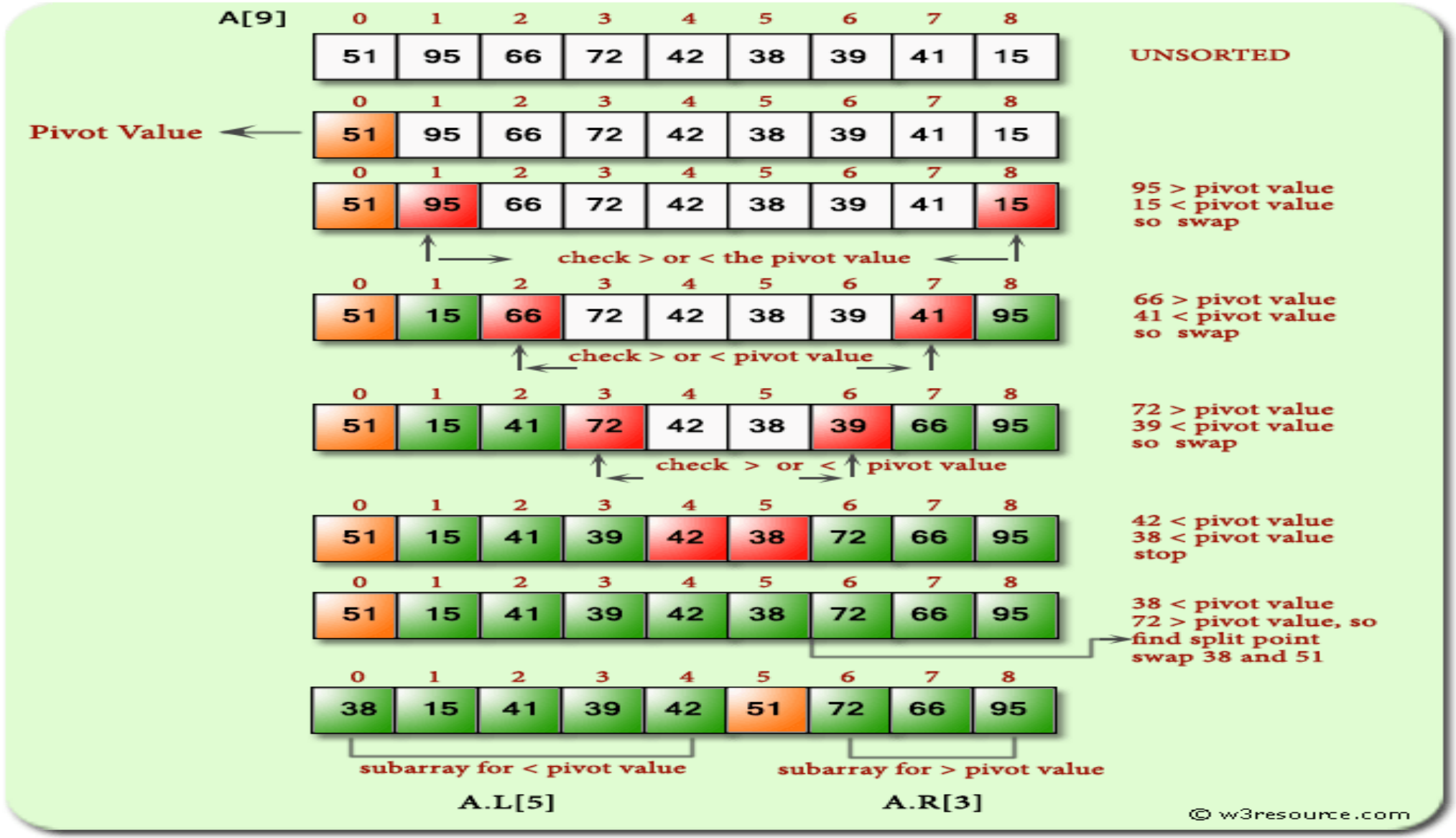
```
{
    0 references
    static void Main(string[] args)
    {
        int[] arr = { 3, 0, 2, 5, -1, 4, 1 };
        int temp;
        for (int p = 0; p <= arr.Length - 2; p++)
        {
            //massivin birinci ədədi ilə ikinci ədədin müqayise edirik və
            //əgər birinci ədəd ikinci edədən böyükdürsə birinci ədəd ikinci
            //ədədin yerinə ikinci ədəd birinci ədədin yerinə yazırıq sonra
            //ikinci üçüncü, üçüncü dördüncü v.s ilə yoxlanılır və
            //ikinci dövr(i) bitdikdə birinci dövr(p) işə başlayır və
            //müqayisə olunmayanlar müqayisə edilir və massivin bütün
            //elementləri bu üsulla yoxlanaraq sıra ilə düzülür.
            for (int i = 0; i <= arr.Length - 2; i++)
            {
                if (arr[i] > arr[i + 1])
                {
                    temp = arr[i + 1];
                    arr[i + 1] = arr[i];
                    arr[i] = temp;
                }
            }
        }
```

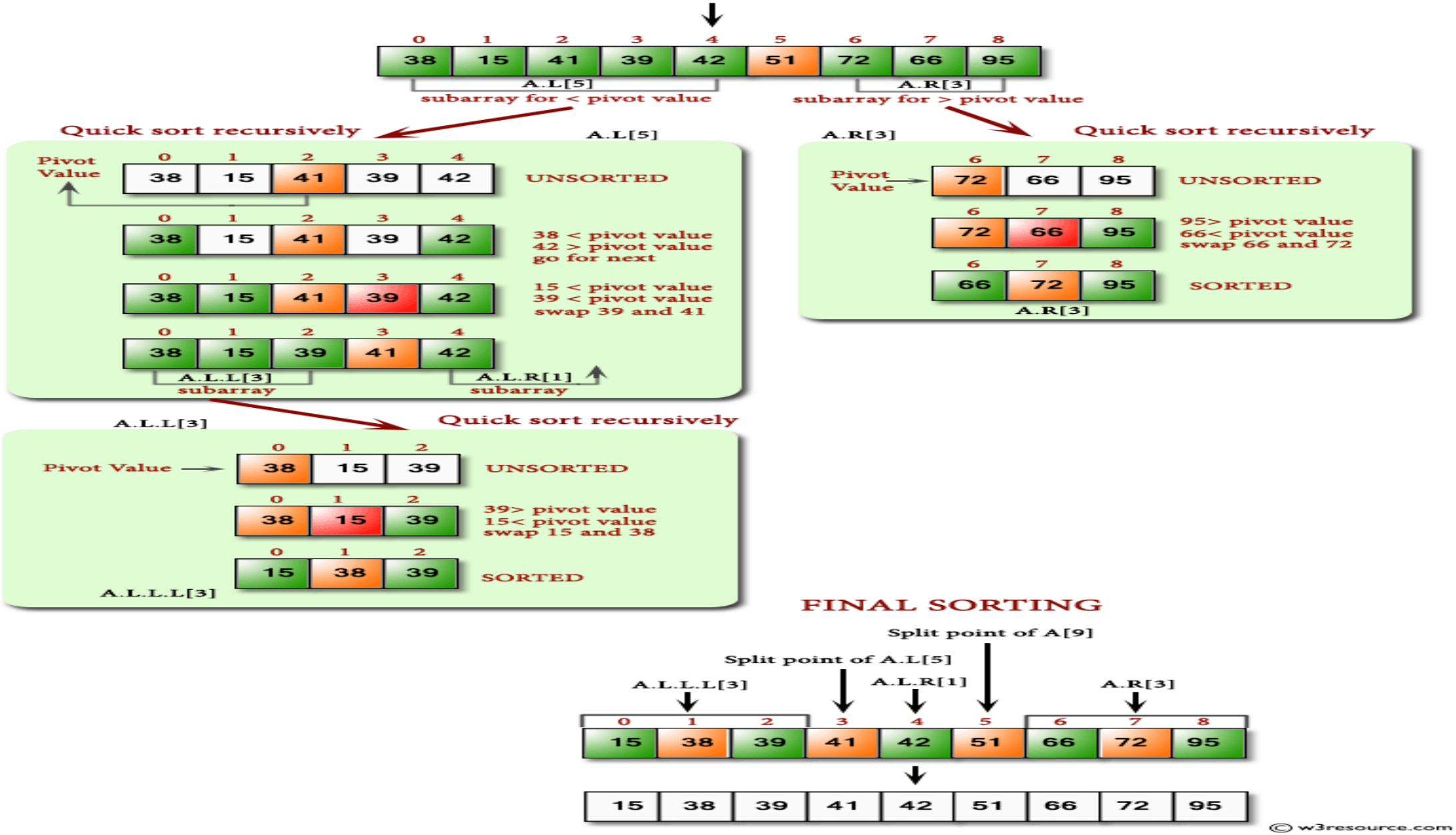
Insertion Sort

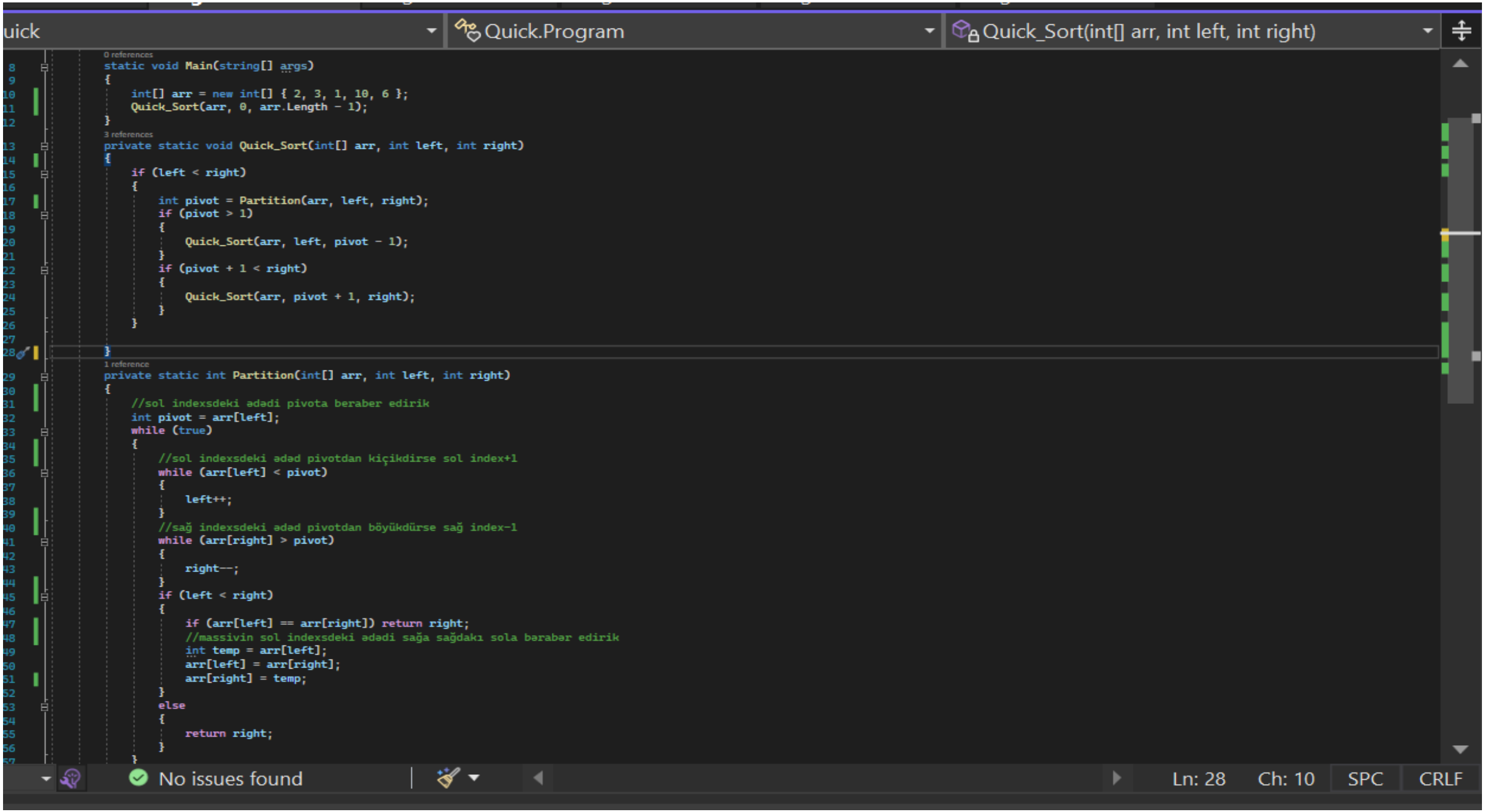


```
{
    0 references
    static void Main(string[] args)
    {
        int[] arr = { 3, 0, 2, 5, -1, 4, 1 };
        int temp;
        for (int i = 0; i < arr.Length - 1; i++)
        {
            //massivin ikinci ədədi ilə birinci ədədin müqayise edirik və
            //əgər birinci ədəd ikinci edədən böyükdürsə birinci ədəd ikinci
            //ədədin yerinə ikinci ədəd birinci ədədin yerinə yazırıq sonra
            //3-cü 2-ci və 1-ci ilə,4-cü 3-cü, 2-ci və 1-ci ilə və v.s ilə yoxlanılır və
            // massivin bütün elementləri bu üsulla yoxlanaraq sıra ilə düzülür.
            for (int j = i + 1; j > 0; j--)
            {
                if (arr[j - 1] > arr[j])
                {
                    temp = arr[j - 1];
                    arr[j - 1] = arr[j];
                    arr[j] = temp;
                }
            }
        }
    }
}
```

Quick Sort


A[9]
0 1 2 3 4 5 6 7 8
51 95 66 72 42 38 39 41 15
UNSORTED
Pivot Value
51 95 66 72 42 38 39 41 15
51 95 66 72 42 38 39 41 15
95 > pivot value
15 < pivot value
so swap
check > or < the pivot value
51 15 66 72 42 38 39 41 95
66 > pivot value
41 < pivot value
so swap
check > or < pivot value
51 15 41 72 42 38 39 66 95
72 > pivot value
39 < pivot value
so swap
check > or < pivot value
51 15 41 39 42 38 72 66 95
42 < pivot value
38 < pivot value
stop
51 15 41 39 42 38 72 66 95
38 < pivot value
72 > pivot value, so
find split point
swap 38 and 51
38 15 41 39 42 51 72 66 95
subarray for < pivot value
subarray for > pivot value
A.L[5]
A.R[3]
© w3resource.com


0 1 2 3 4 5 6 7 8
38 15 41 39 42 51 72 66 95
A.L[5]
subarray for < pivot value
A.R[3]
subarray for > pivot value
Quick sort recursively
A.L[5]
Pivot Value
UNSORTED
38 < pivot value
42 > pivot value
go for next
15 < pivot value
39 < pivot value
swap 39 and 41
A.L.L[3]
subarray
A.L.R[1]
subarray
A.L.L[3]
Quick sort recursively
Pivot Value
UNSORTED
39> pivot value
15< pivot value
swap 15 and 38
SORTED
A.L.L.L[3]
A.R[3]
Quick sort recursively
Pivot Value
UNSORTED
95> pivot value
66< pivot value
swap 66 and 72
SORTED
A.R[3]
FINAL SORTING
Split point of A[9]
Split point of A.L[5]
A.L.L.L[3]
A.L.R[1]
A.R[3]
15 38 39 41 42 51 66 72 95
15 38 39 41 42 51 66 72 95
© w3resource.com


uick
Quick.Program
Quick_Sort(int[] arr, int left, int right)
0 references
static void Main(string[] args)
{
int[] arr = new int[] { 2, 3, 1, 10, 6 };
Quick_Sort(arr, 0, arr.Length - 1);
}
3 references
private static void Quick_Sort(int[] arr, int left, int right)
{
if (left < right)
{
int pivot = Partition(arr, left, right);
if (pivot > 1)
{
Quick_Sort(arr, left, pivot - 1);
}
if (pivot + 1 < right)
{
Quick_Sort(arr, pivot + 1, right);
}
}
}
1 reference
private static int Partition(int[] arr, int left, int right)
{
//sol indexsdeki ədədi pivota beraber edirik
int pivot = arr[left];
while (true)
{
//sol indexsdeki ədəd pivotdan kiçikdirse sol index+1
while (arr[left] < pivot)
{
left++;
}
//sağ indexsdeki ədəd pivotdan böyükdürse sağ index-1
while (arr[right] > pivot)
{
right--;
}
if (left < right)
{
if (arr[left] == arr[right]) return right;
//massivin sol indexsdeki ədədi sağa sağdakı sola bərabər edirik
int temp = arr[left];
arr[left] = arr[right];
arr[right] = temp;
}
else
{
return right;
}
}
No issues found
Ln: 28
Ch: 10
SPC
CRLF

Counting sort

```
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            int[] array = { 2, 5, -4, 11, 0, 8, 22, 67, 51, 6 };
            int[] sortedArray = new int[array.Length];
            // find smallest and largest value
            int minVal = array[0];
            int maxVal = array[0];
            for (int i = 1; i < array.Length; i++)
            {
                if (array[i] < minVal) minVal = array[i];
                else if (array[i] > maxVal) maxVal = array[i];
            }
            // init array of frequencies
            int[] counts = new int[maxVal - minVal + 1];
            // init the frequencies
            for (int i = 0; i < array.Length; i++)
            {
                counts[array[i] - minVal]++;
            }
            // recalculate
            counts[0]--;
            for (int i = 1; i < counts.Length; i++)
            {
                counts[i] = counts[i] + counts[i - 1];
            }
            // Sort the array
            for (int i = array.Length - 1; i >= 0; i--)
            {
                sortedArray[counts[array[i] - minVal]--] = array[i];
            }
        }
    }
}
```

No issues found