Merge Sort

Merge Sort is one of the most popular sorting algorithms that is based on the principle of Divide and Conquer Algorithm.

Merge Sort Pseudocode:

```
procedure mergesort( var a as array )
     if ( n == 1 )
          return a
     var l1 as array = a[0] ... a[n/2]
     var l2 as array = a[n/2+1] ... a[n]
     l1 = mergesort( l1 )
     l2 = mergesort( l2 )
     return merge( l1, l2 )
end procedure
procedure merge( var a as array, var b as array )
     var c as array
     while ( a and b have elements )
          if ( a[0] > b[0] )
               add b[0] to the end of c
               remove b[0] from b
          end if
          else
               add a[0] to the end of c
               remove a[0] from a
          end else
     end while
     while ( a has elements )
          add a[0] to the end of c
          remove a[0] from a
     end while
     while ( b has elements )
          add b[0] to the end of c
          remove b[0] from b
     end while
     return c
end procedure
```

Complexities: Time Complexity: Best – O(nlogn), Average – O(nlogn), Worst – O(nlogn)
Space Complexity: O(n)
Stability: Yes

Applications: Merge sort is used in

- Inversion count problem
- External sorting
- E-commerce applications

Source Code:

```
using System;

namespace MergeSort
{
    class Program
    {
        static void Main(String[] args)
        {
            Input();
        }

        static void Input()
        {
            Console.Write("Enter Number of Items: ");
            int noOfItems = Convert.ToInt32(Console.ReadLine());

            int[] itemsList = new int[noOfItems];
            Console.Write("Enter Items: ");
            for (int i = 0; i < noOfItems; i++)
            {
                itemsList[i] = Convert.ToInt32(Console.ReadLine());
            }
            //Console.Write("For ascending write 'a' or descending write 'd': ");
            //char order = Convert.ToChar(Console.ReadLine());
            MergeSort(itemsList, 0, itemsList.Length-1);
            Output(itemsList);
        }

        static void MergeSort(int[] itemsList, int left, int right)
        {
            if(left >= right)
            {
                return;
            }
            int mid = left + (right - left) / 2;
            MergeSort(itemsList, left, mid);
            MergeSort(itemsList, mid+1, right);
            Merge(itemsList, left, mid, right);
        }

        static void Merge(int[] itemsList, int left, int mid, int right)
        {
            int leftSize = mid - left + 1;
            int rightSize = right - mid;

            int[] leftArray = new int[leftSize];
            int[] rightArray = new int[rightSize];

            int leftIndex, rightIndex, itemListIndex;

            for (leftIndex = 0; leftIndex < leftSize; leftIndex++)
            {
                leftArray[leftIndex] = itemsList[left+ leftIndex];
            }
            for (rightIndex = 0; rightIndex < rightSize; rightIndex++)
            {
                rightArray[rightIndex] = itemsList[mid+1+rightIndex];
            }

            leftIndex = 0;
            rightIndex = 0;
```

```
            for (itemListIndex = left; leftIndex < leftSize && rightIndex <
            rightSize; itemListIndex++)
            {
                if(leftArray[leftIndex] < rightArray[rightIndex])
                {
                    itemsList[itemListIndex] = leftArray[leftIndex];
                    leftIndex++;
                }
                else
                {
                    itemsList[itemListIndex] = rightArray[rightIndex];
                    rightIndex++;
                }
            }
            while(leftIndex < leftSize)
            {
                itemsList[itemListIndex] = leftArray[leftIndex];
                itemListIndex++;
                leftIndex++;
            }
            while (rightIndex < rightSize)
            {
                itemsList[itemListIndex] = rightArray[rightIndex];
                itemListIndex++;
                rightIndex++;
            }
        }

        static void Output(int[] itemList)
        {
            Console.Write("After sorting: ");
            for (int i = 0; i < itemList.Length; i++)
            {
                Console.Write($"{itemList[i]}\t");
            }
        }
    }
}
```