Algorithm - 03

-- MergeSort

A. Problem Description

The MergeSort-Algorithm is used to sort a set of elements using 'Divide and Conquer'. Its basic idea is to devide the set into two subsets with similar size and sort the subsets before merged.

B. Description of the algorithm

```
MergeSort(A, B, left, right)
    if left < right
        i = (left + right) /2
        MergeSort(A, left, i)
        MergeSort(A, i + 1, right)
        Merge(A, B, left, i, right)
        Copy(A, B, left, right)
```

$$T(n) = \begin{cases} O(1) & n <= 1 \\ 2T(n/2) + O(n) & n > 1 \end{cases}$$

$$\Rightarrow T(n) = O(n \lg n)$$

C. Code.[Python]

```
#!/usr/bin/python
# Filename: MergeSort.py

import random

def MergeSort(A, B, left, right):
```

```
    if left < right:
        i = (left + right) / 2
        MergeSort(A, B, left, i)
        MergeSort(A, B, i + 1, right)
        Merge(A, B, left, i, right)
        Copy(A, B, left, right)

def Copy(new, old, left, right):
    for i in range(left, right + 1):
        new[i] = old[i]

def Merge(A, B, left, i, right):
    AL = [A[j] for j in range(left, i + 1)]
    AL.append(0)
    AR = [A[j] for j in range(i + 1, right + 1)]
    AR.append(0)
    Bindex = left
    Lindex = Rindex = 0
    while 1:
        if AL[Lindex] == 0:
            for j in range(Rindex, len(AR) - 1):
                B[Bindex] = AR[j]
                Bindex += 1
            break
        elif AR[Rindex] == 0:
            for j in range(Lindex, len(AL) - 1):
                B[Bindex] = AL[j]
                Bindex += 1
            break
        elif AL[Lindex] < AR[Rindex]:
            B[Bindex] = AL[Lindex]
            Bindex += 1
            Lindex += 1
            continue
        elif AL[Lindex] >= AR[Rindex]:
            B[Bindex] = AR[Rindex]
            Bindex += 1
            Rindex += 1
        else:
            pass
```