NAME : SABARIVASAN V
REG NO : 205001085
CLASS : CSE - B

ASSIGNMENT 3

```
# 1. Implement Merge sort and plot graph for the number of steps
of execution
#  for variable sizes of 'n'.
# 2. Write recurrence relation for the above problem and
# derive time complexity

import matplotlib.pyplot as plt
from math import log
import numpy as np

def merge(arr, l, m, r):
    global count
    n1 = m - l + 1
    n2 = r - m
    L = [0] * (n1)
    R = [0] * (n2)
    for i in range(0, n1):
        L[i] = arr[l + i]
    for j in range(0, n2):
        R[j] = arr[m + 1 + j]
    i = 0
    j = 0
    k = l
    while i < n1 and j < n2:
        count += 1
        if L[i] <= R[j]:
            arr[k] = L[i]
```

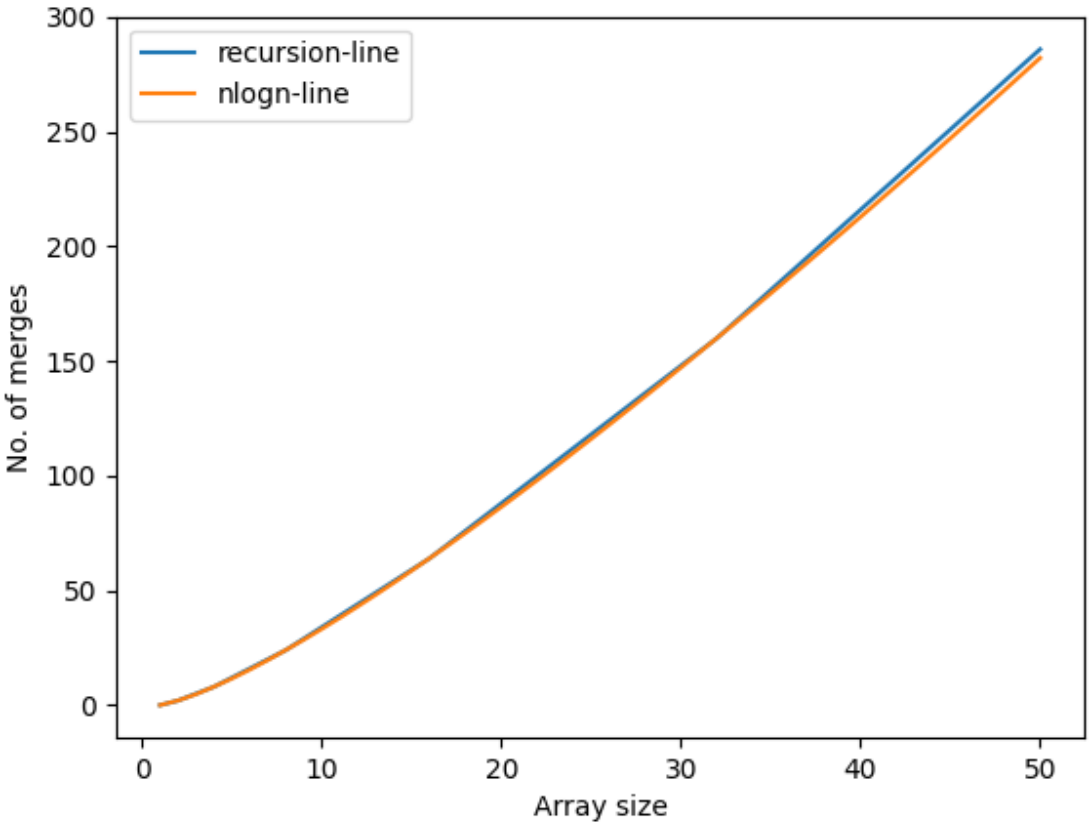
```
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    while i < n1:
        count += 1
        arr[k] = L[i]
        i += 1
        k += 1
    while j < n2:
        count += 1
        arr[k] = R[j]
        j += 1
        k += 1
def mergeSort(arr, l, r):
    if l < r:
        m = l+(r-l)//2
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)

xpoints = [ i for i in range(1,51) ]
ypoints = []
for i  in range(1,51) :
    count = 0
    array = np.random.randint(0,500,size=(i))
    mergeSort(array,0,i-1)
    ypoints.append(count)

zpoints = [ n*log(n,2) for n in range(1,51) ]
```

```
plt.plot(xpoints, ypoints,label ='recursion-line')
plt.plot(xpoints, zpoints,label ='nlogn-line')
plt.xlabel("Array size")
plt.ylabel("No. of merges")
plt.legend()
plt.show()
```



Time complexity : O(nlogn)
Recurrence Relation : T(n) = T (n / 2) + n