## Practical 2

**Aim:** Write a program to implement merge sot algorithm. Compare the time and memory complexity.

**Theory:**

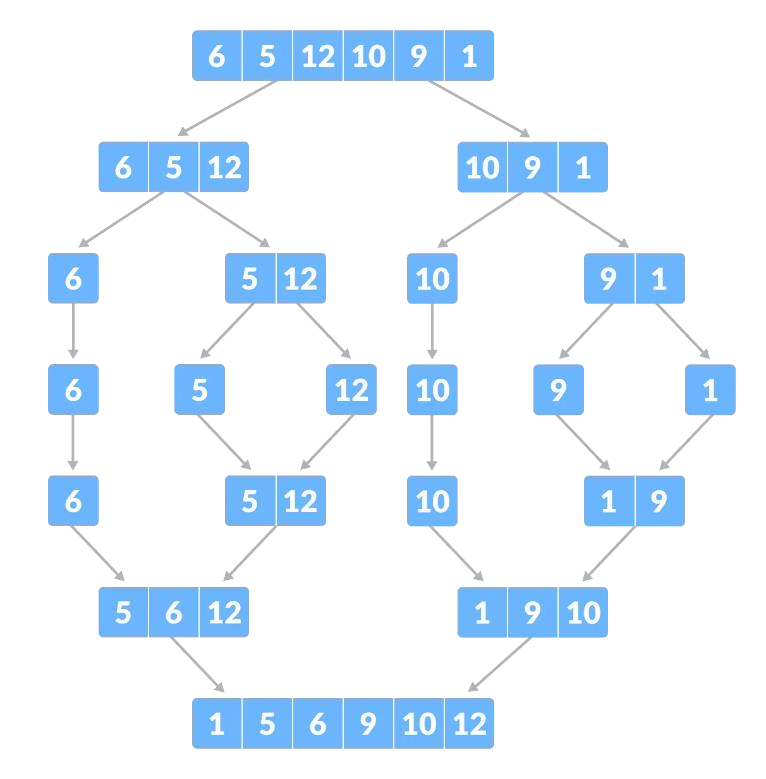
1. If there is only one element in the array then it is already sorted.

1. Insertion sort algorithm divides the array recursively into two halves until it can’t be divided

1. It Merge the smaller arrays into new array in sorted order

1. Now we get the fully sorted array

**Example:**



**Algorithm:**

1. MergeSort(arr, left, right):
2. if left > right
3. return
4. mid = (left+right)/2
5. mergeSort(arr, left, mid)
6. mergeSort(arr, mid+1, right)
7. merge(arr, left, mid, right)
8. end

**Code:**

def merge(arr, l, m, r):

n1 = m - l + 1

n2 = r - m

#Neeraj Appari 021

# create temp arrays

L = [0] \* (n1)

R = [0] \* (n2)

# Copy data to temp arrays L[] and R[]

for i in range(0, n1):

L[i] = arr[l + i]

for j in range(0, n2):

R[j] = arr[m + 1 + j]

# Merge the temp arrays back into arr[l..r]

i = 0 # Initial index of first subarray

j = 0 # Initial index of second subarray

k = l # Initial index of merged subarray

while i < n1 and j < n2:

if L[i] <= R[j]:

arr[k] = L[i]

i += 1

else:

arr[k] = R[j]

j += 1

k += 1

# Copy the remaining elements of L[], if there

# are any

while i < n1:

arr[k] = L[i]

i += 1

k += 1

# Copy the remaining elements of R[], if there

# are any

while j < n2:

arr[k] = R[j]

j += 1

k += 1

def mergeSort(arr, l, r):

if l < r:

# Same as (l+r)//2, but avoids overflow for

# large l and h

m = l+(r-l)//2

# Sort first and second halves

mergeSort(arr, l, m)

mergeSort(arr, m+1, r)

merge(arr, l, m, r)

# Driver code to test above

arr = [12, 11, 13, 5, 6, 7]

n = len(arr)

print("Given array is")

for i in range(n):

print("%d" % arr[i],end=" ")

mergeSort(arr, 0, n-1)

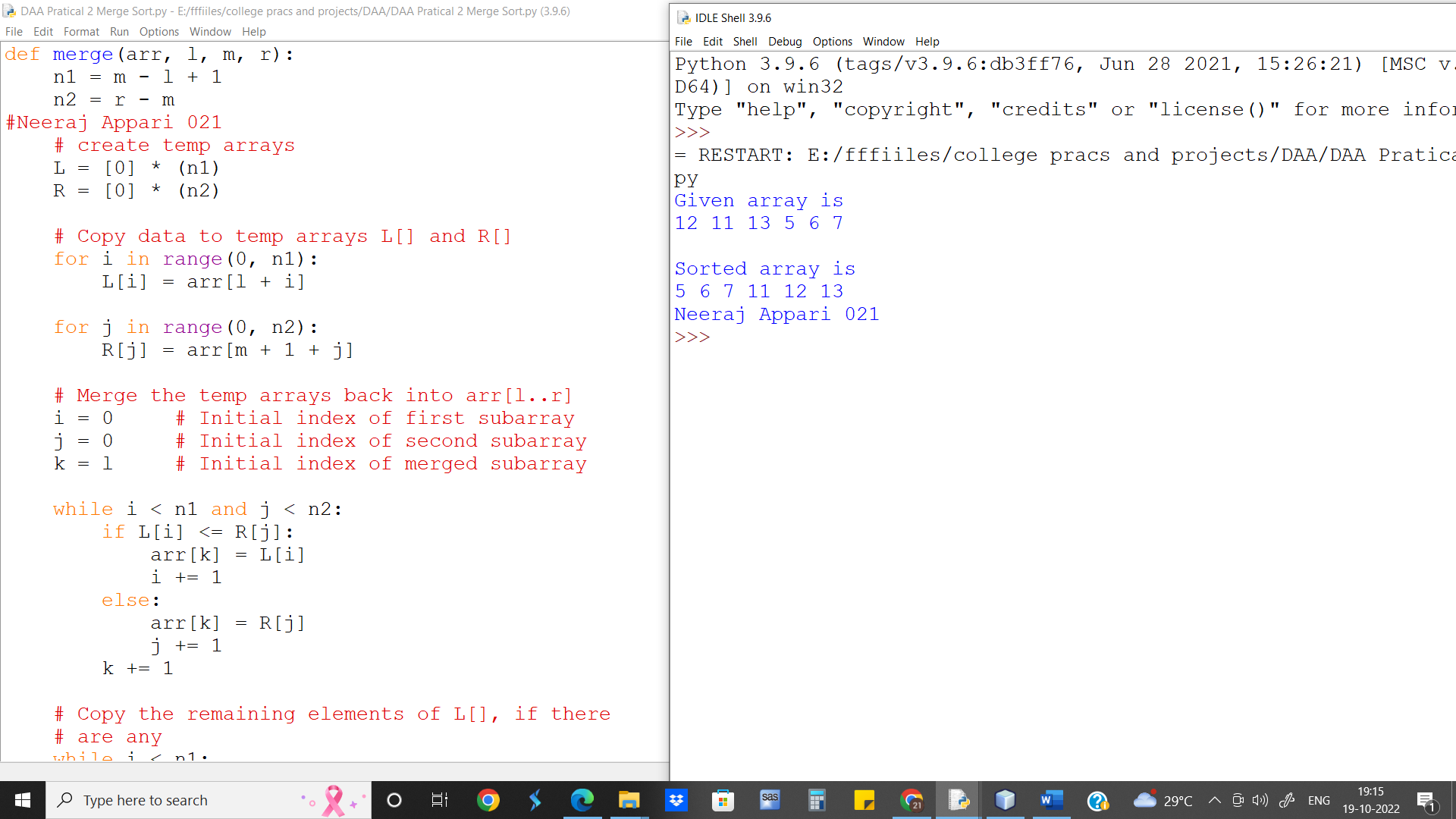
print("\n\nSorted array is")

for i in range(n):

print("%d" % arr[i],end=" ")

print("\nNeeraj Appari 021")

**Output**:1 5 6 9 10 12



**Loop Invariant:**

At the start of each iteration k of the for loop, the nonempty part of S contains the k − 1 smallest elements of L and R, in sorted order. Moreover, L[i] and R[j] are the smallest elements of their arrays that have not been copied to S.

**Run Time analysis of above Algorithm:**

Runtime for Insertion sort is O(nlogn)

**Conclusion:** Merge sort uses a divide-and-conquer method recursively sorts the elements of a list while Bubble, Insertion and Selection have a quadratic time complexity that limit its use to small number of elements. Merge sort uses divide-and-conquer to speed up the sorting.