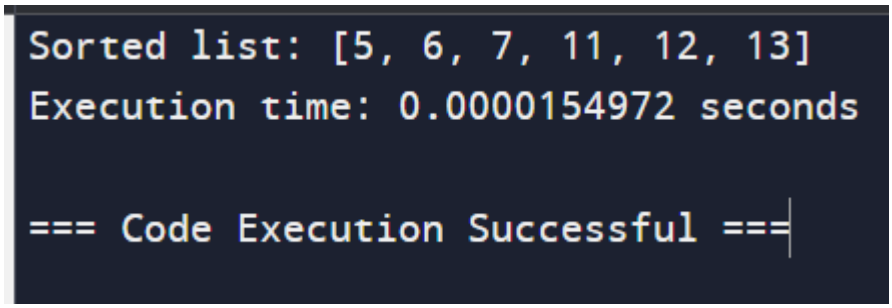
80. Merge sort

PROGRAM:-

```
import time

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2  # Finding the mid of the array
        left_half = arr[:mid]  # Dividing the array elements into 2 halves
        right_half = arr[mid:]

        merge_sort(left_half)  # Sorting the first half
        merge_sort(right_half)  # Sorting the second half

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

def find_merge_sort_time(arr):
    start_time = time.time()  # Start time measurement

    merge_sort(arr)  # Perform merge sort

    end_time = time.time()  # End time measurement
    elapsed_time = end_time - start_time

    return elapsed_time

# Example usage
example_list = [12, 11, 13, 5, 6, 7]
execution_time = find_merge_sort_time(example_list)
```

```
print(f"Sorted list: {example_list}")
print(f"Execution time: {execution_time:.10f} seconds")
```

OUTPUT:-



TIME COMPLEXITY:-O(n log n)