UIT2402 -- ADVANCED DATA STRUCTURES AND ALGORITHM ANALYSIS LAB

EX NO : 1

MERGE SORT AND QUICK SORT

1. Write a python program to sort the following elements in ascending order using merge sort algorithm.

'january', 'february', 'march', 'april','may','june','july',' august','september', 'october','november', 'december'.

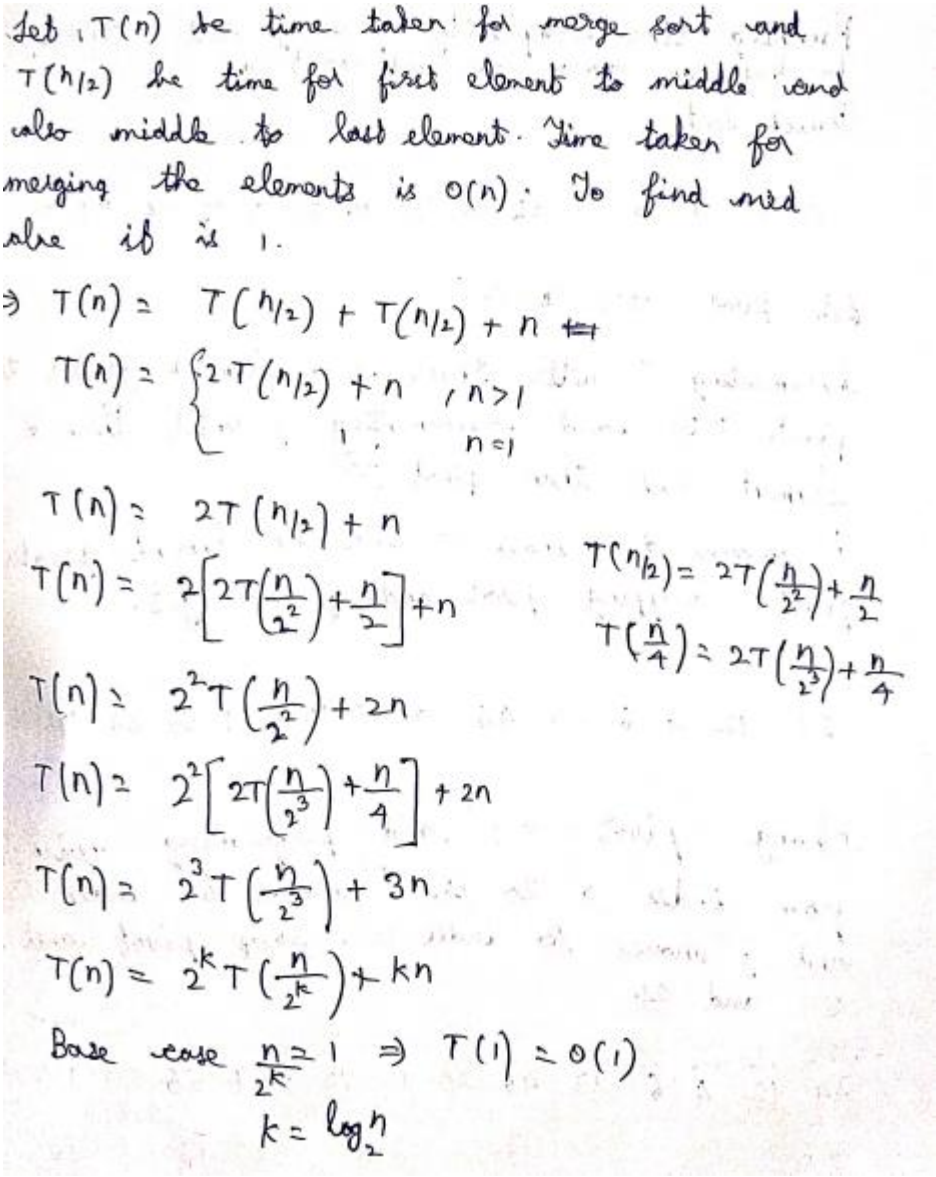
and derive the running time complexity for your code.

ALGORITHM:

- If the array has one or zero elements, return it (base case).
- Find the middle index of the array: mid = n // 2.
- Divide the array into two halves: left = arr[0:mid] and right = arr[mid:n].
- Recursively apply Merge Sort to the left half.
- Recursively apply Merge Sort to the right half.
- Merge the two sorted halves by comparing elements from both:
- Initialize two pointers for left and right.
- Compare elements and insert the smaller element into a new sorted list.

- If any elements remain in left or right, append them to the sorted list.
- Return the merged sorted list.
- Repeat the process until the entire array is sorted.
- The sorted array is now in ascending order.

RUNNING TIME COMPLEXITY:



Let, $T(n)$ be time taken for merge sort and $T(n/2)$ be time for first element to middle and also middle to last element. Time taken for merging the elements is $O(n)$. To find mid value it is 1.

$$\Rightarrow T(n) = T(n/2) + T(n/2) + n$$

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

$$T(n) = 2T(n/2) + n$$

$$T(n) = 2\left[2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right] + n \qquad T(n/2) = 2T\left(\frac{n}{2^2}\right) + \frac{n}{2}$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{2^3}\right) + \frac{n}{4}$$

$$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + 2n$$

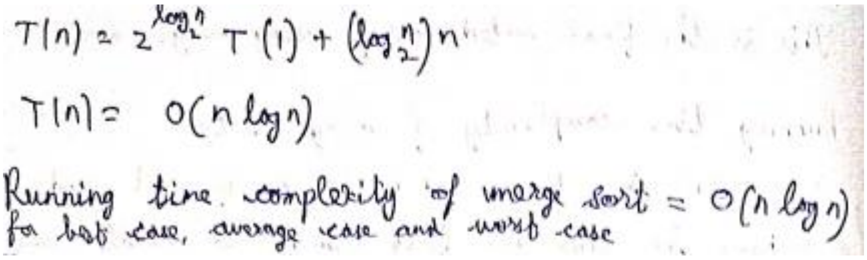
$$T(n) = 2^2\left[2T\left(\frac{n}{2^3}\right) + \frac{n}{4}\right] + 2n$$

$$T(n) = 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

Base case $\frac{n}{2^k} = 1 \Rightarrow T(1) = O(1)$.

$$k = \log_2 n$$



$T(n) = 2^{\log_2 n} T(1) + (\log_2 n) n$

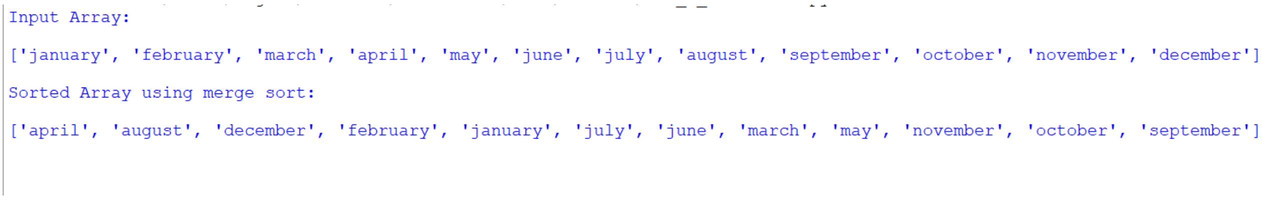
$T(n) = O(n \log n)$

Running time complexity of merge sort = $O(n \log n)$ for best case, average case and worst case

CODING:

```
def merge_sort(inp_arr):
    size=len(inp_arr)
    if size>1:
        middle=size//2
        left_arr=inp_arr[:middle]
        right_arr=inp_arr[middle:]
        merge_sort(left_arr)
        merge_sort(right_arr)
        p=0
        q=0
        r=0
        left_size=len(left_arr)
        right_size=len(right_arr)
        while p<left_size and q<right_size:
            if left_arr[p]<right_arr[q]:
              inp_arr[r]=left_arr[p]
              p+=1
            else:
                inp_arr[r]=right_arr[q]
                q+=1
            r+=1
        while p<left_size:
            inp_arr[r]=left_arr[p]
            p+=1
            r+=1
        while q<right_size:
            inp_arr[r]=right_arr[q]
            q+=1
            r+=1
inp_arr=['january','february','march','april','may','june','july','
august','september','october','november','december']
print("Input Array:\n")
print(inp_arr)
merge_sort(inp_arr)
print("\nSorted Array using merge sort:\n")
print(inp_arr)
```

OUTPUT:



```
Input Array:
['january', 'february', 'march', 'april', 'may', 'june', 'july', 'august', 'september', 'october', 'november', 'december']
Sorted Array using merge sort:
['april', 'august', 'december', 'february', 'january', 'july', 'june', 'march', 'may', 'november', 'october', 'september']
```

2. Write a python program to sort the following elements in descending order using a quick sort algorithm.

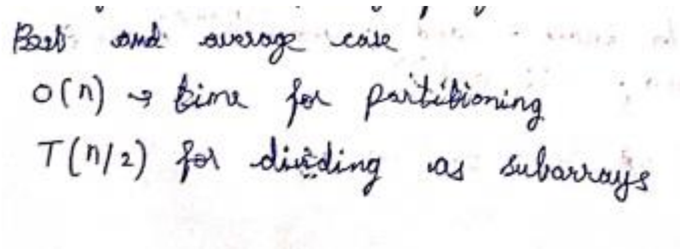
'january', 'february', 'march', 'april','may','june','july',' august','september', 'october','november', 'december'.

and derive the running time complexity for your code.

ALGORITHM:

- If the array has one or zero elements, return it (base case).
- Choose a pivot element (e.g., the last element).
- Partition the array into two subarrays:
- greater[] (elements greater than or equal to the pivot).
- lesser[] (elements smaller than the pivot).
- Recursively apply Quick Sort to the greater[] subarray.
- Recursively apply Quick Sort to the lesser[] subarray.
- Concatenate the sorted greater[], the pivot, and the sorted lesser[].
- Repeat this process for all subarrays.
- Optimize pivot selection to avoid worst-case scenarios.
- The sorted array is now in descending order.
- Return the final sorted array.

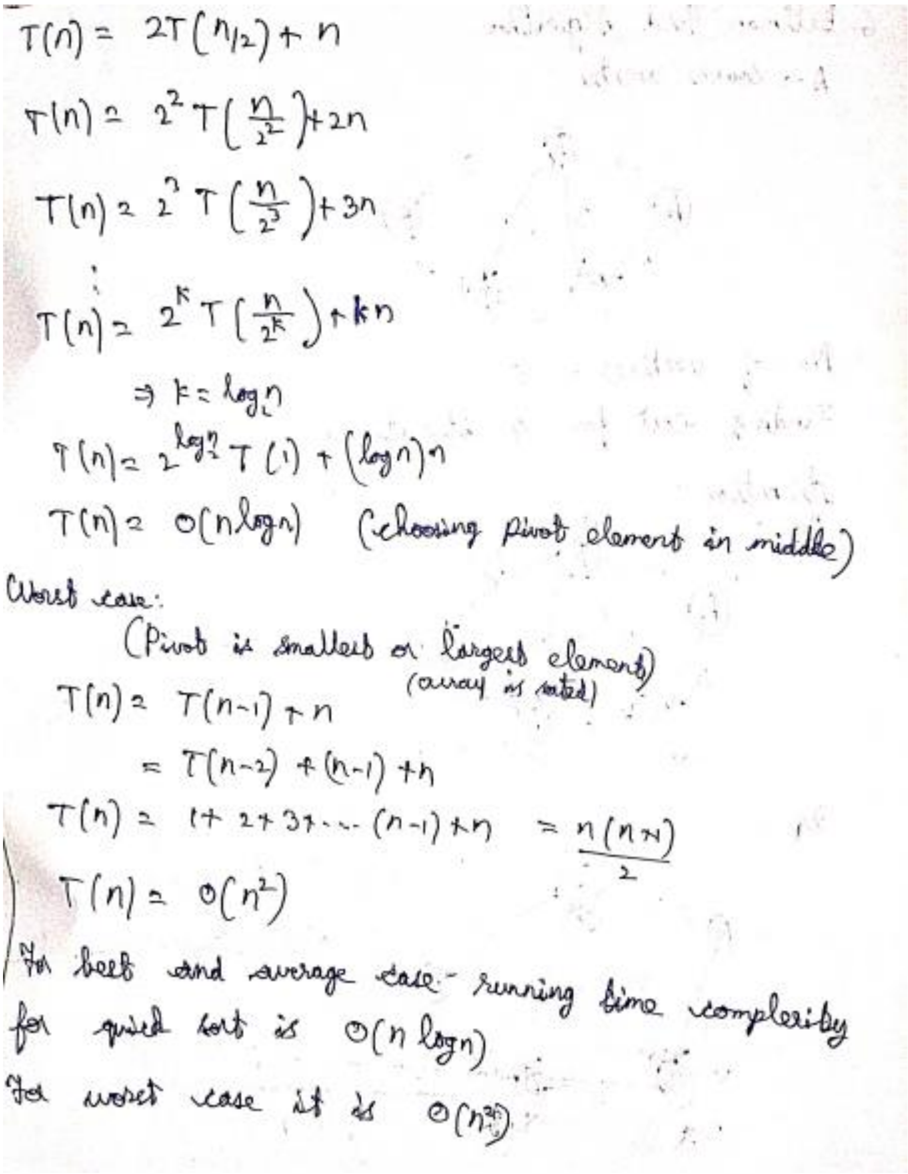
RUNNING TIME COMPLEXITY:



Best and average case

$O(n) \rightarrow$ time for partitioning

$T(n/2)$ for dividing as subarrays



$$T(n) = 2T(n/2) + n$$

$$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + 2n$$

$$T(n) = 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

$$\vdots$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\Rightarrow k = \log_2 n$$

$$T(n) = 2^{\log_2 n} T(1) + (\log n)n$$

$T(n) = O(n \log n)$ (choosing pivot element in middle)

Worst case:

(Pivot is smallest or largest element)
(array is sorted)

$$T(n) = T(n-1) + n$$

$$= T(n-2) + (n-1) + n$$

$$T(n) = 1 + 2 + 3 + \dots (n-1) + n = \frac{n(n+1)}{2}$$

$$T(n) = O(n^2)$$

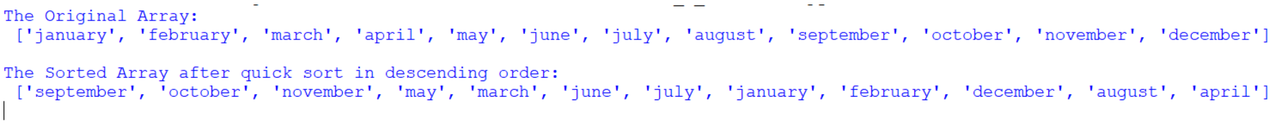
For best and average case - running time complexity for quick sort is $O(n \log n)$

For worst case it is $O(n^2)$

CODING:

```
def partition(arr,low,high):
  pivot=arr[high]
  i=low-1
  for j in range(low,high):
    if arr[j]>=pivot:
      i=i+1
      (arr[i],arr[j])=(arr[j],arr[i])
  (arr[i+1],arr[high])=(arr[high],arr[i+1])
  return i+1
def QuickSort(arr,low,high):
  if low<high:
    pivot=partition(arr,low,high)
    QuickSort(arr,low,pivot-1)
    QuickSort(arr,pivot+1,high)
array=['january','february','march','april','may','june','july','august','september','october','november','december']
print("The Original Array:\n",array)
size=len(array)
QuickSort(array,0,size-1)
print('\nThe Sorted Array after quick sort in descending order:\n',array)
```

<u>OUTPUT:</u>



```
The Original Array:
 ['january', 'february', 'march', 'april', 'may', 'june', 'july', 'august', 'september', 'october', 'november', 'december']

The Sorted Array after quick sort in descending order:
 ['september', 'october', 'november', 'may', 'march', 'june', 'july', 'january', 'february', 'december', 'august', 'april']
```