ADA Mini Homework 2

(1)

```
Func: merge(int arr[], int l, int m, int r, int &ans)
        L[m - l] = arr[l -> m]
        // L contains the elements of arr from l to m
        R[r - m] = arr[m + 1 -> r]
        // R contains the elements of arr from m + 1 to r
       i = 0, j = 0, k = l;
       while  i < n1 && j < n2
              if  L[i] <= R[j] :
                     arr[k] = L[i]
                     ans += j
// Count numbers that are smaller than L[i] and from the right side
                     i++
              else
                     arr[k] = R[j];
                     j++
              k++;
       while i < n1
          arr[k] = L[i]
          i++, k++
       while j < n2
          arr[k] = R[j]
          j++, k++

Func: mergeSort(int arr[], int l, int r, int &ans)
       if  l < r
       // Implement merge sort with divide and conquer
         m = l + (r - l) / 2;
         mergeSort(arr, l, m, ans);
         mergeSort(arr, m + 1, r, ans);
         merge(arr, l, m, r, ans);

Func main
         ans = 0
         mergeSort(arr, 0, N - 1, ans)
         print(ans)
```

(2)

- Simplify recurrences
- Ignore floors and ceilings (boundary conditions)
- Assume base cases are constant (for small n)

As a result, we can follow the following steps to prove the time complexity.

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2T(n/2) + O(n) & \text{if } n \geq 2 \end{cases}$$

$$\begin{aligned}
T(n) &\leq 2T(\frac{n}{2}) + cn \quad \text{1}^{\text{st}} \text{ expansion} \\
&\leq 2[2T(\frac{n}{4}) + c\frac{n}{2}] + cn = 4T(\frac{n}{4}) + 2cn \quad \text{2}^{\text{nd}} \text{ expansion} \\
&\leq 4[2T(\frac{n}{8}) + c\frac{n}{4}] + 2cn = 8T(\frac{n}{8}) + 3cn \\
&\vdots \\
&\leq 2^k T(\frac{n}{2^k}) + kcn \quad \text{k}^{\text{th}} \text{ expansion}
\end{aligned}$$

The expansion stops when $2^k = n$

$$\begin{aligned}
T(n) &\leq nT(1) + cn\log_2 n \\
&= O(n) + O(n \log n) \\
&= O(n \log n)
\end{aligned}$$