MERGESORT

Input : n-numbers $a_1, a_2, \ldots, a_n$.
Output : Arrange them in non-decreasing order.

Strategy : Let L be the list of n-numbers

Algorithm Mergesort(L, n):

// L is a list of $n \geq 0$ numbers.

if $n \geq 2$

then
- $m := \lfloor n/2 \rfloor$;
- L_1 := List containing first m elements of L ;
- L_2 := List containing the last n-m elements of L ;
- L_1 := Mergesort (L_1, m);
- L_2 := Mergesort (L_2, n-m);
- L := Merge (L_1, L_2);

Return L.

How does Merge (L_1, L_2) works ?

Input : Two sorted lists L_1 and L_2

Output : Output elements of $L_1 \cup L_2$ in sorted order

```
Algorithm Merge(L1, L2):
// L1 & L2 are two sorted lists.
L := Empty-List;
While L1 ≠ φ and L2 ≠ φ do
    x := First-element of L1;
    y := First-element of L2;
    If x ≤ y then remove x from L1;
                  append x to L;
    else        remove y from L2;
                append y to L;
If L1 is non-empty then append L1 to L;
If L2 is non-empty then append L2 to L;
Return L;
```

Correctness: Induction on n

Base Case $n=1$: obvious as a single number is sorted.

I.H: Let $n \geq 2$ and assume that for any integer k, $0 \leq k < n$ and for any list L' of k numbers, algorithm Mergesort(L', k) returns a list containing the elements of L' in sorted order

Let L be a list of n-numbers.

Observe

The call Mergesort(L_1, m) is on a list of size $m < n$.

By I.H. Mergesort returns L_1 in sorted order.

The call Mergesort$(L_2, n-m)$ is on a list of size $n-m < n$.

By I.H. Mergesort returns L_2 in sorted order

Then Merge(L_1, L_2) sorts them and returns L.

Thus the list L returned by Mergesort(L, n) is sorted.

Running Time:

Count how many times $x \leq y$ is checked in Merge(L_1, L_2)?

$\leq |L_1| + |L_2|$ as each time a comparison is made, we add an element to L.

Let $n = 2^k$ for some integer $k \geq 0$.

Let $T(n)$ = Maximum # comparisons made in Mergesort(L, n) on any input list of n-numbers.

Observe that $T(1) = 0$ {No comparisons are made when $n = 1$.}

Assume $n \geq 2$.

Call to Mergesort(L_1, m) is a recursive call and since $m = n/2$ {As n is a power of 2}

comparisons made in Mergesort$(L_1, m) \leq T\left(\frac{n}{2}\right)$.

Similarly

Comparison made in Mergesort$(L_2, n-m) \leq T\left(\frac{n}{2}\right)$

Comparisons made in Merge$(L_1, L_2) \leq |L_1| + |L_2| = n$.

Thus.

$$T(1) = 0$$

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

$$= 2T\left(\frac{n}{2}\right) + n \qquad \forall\, n \ge 2.$$

What is $T(n) = ?$ Assume $n = 2^k$ and k is large

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

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

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

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

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

$$\vdots$$

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

Since $\boxed{n = 2^k}$, $\frac{n}{2^k} = 1$ and $T\left(\frac{n}{2^k}\right) = T(1) = 0$ by base case.

$$\Rightarrow T(n) \le 2^k T(1) + kn = nT(1) + kn$$

$$\Leftrightarrow T(n) \le kn = n\log n.$$