Merge Sort O(nlogn)

- Based on divide & conquer

```
Merge(A[1 ... m], B[1 ... n], C[1 ... m+n])
  i = 1
  j = 1
  for k = 1 to m+n:
    if A[i] <= B[j]:
        C[k] = A[i]
        i = i + 1
    else:
        C[k] = B[j]
        j = j + 1
```

Proof of correctness:

Loop invariant: at beginning of the kth iteration, C[1 ... k-1] contains the k-1 smallest elements of $A \cup B$ in sorted order, and these elements are A[1 ... i-1] and B[1 ... j-1].

Initialization: When k = 1, i = j = 1, and invariant holds trivially.

Maintenance: Assume invariant holds for iteration k. We want to show also holds for iteration k+1. Assume A[i] ≤ B[j]. Then A[i] is smallest among all remaining items. Therefore, invariant holds also at iteration k+1. If A[i] > B[j], similar.

Termination: When k = m+n+1 (end of iteration), C contains all elements of $A \cup B$ in sorted order.

Runtime: O(m+n)

MERGESORT (A[1 ... n])

1. MERGESORT(A[1 ... n/2])
2. MERGESORT(A[n/2+1 ... n])
3. MERGE both output

Correctness follows easily from that MERGE.

Runtime: Denote by T(n) the runtime on input of size n.

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

Wrong Claim: T(n) = O(n)

Wrong Proof: By induction on n

Assume holds for 1, ... , n-1

Let’s prove for n: $T(n) = 2 \cdot T(n/2) + n = 2 \cdot O(n) + n = O(n)$

// can only assume $T(n) \le 100 \cdot n$ (a concrete number)

// inductive step will get $T(n) \le 101 \cdot n$, wrong

3 approaches

1. Recursion tree (a bit informal but usually OK)

T(n) = 2 · T(n/2) + n

= 4 · T(n/4) + 2n

= 8 · T(n/8) + 3n

... ($log_2 n$ steps)

= n$log_2 n$

2. Induction / Substitution

   Claim: for large enough c, T(n) ≤ c · nlogn

   Proof: Assume by induction that holds for 1, ..., n-1

   T(n) = 2 · T(n/2) + n

   ≤ 2 · c · n/2 log(n/2) + n

   = c · n · log(n/2) + n

   ≤ c · n · log(n)

3. Master theorem