

## Main Steps

There are four main steps for a divide and conquer solution.

**Step 1: Define your recursive sub-problem.** Describe in English what your sub-problem means, what its parameters are, and anything else necessary to understand it.

**Step 2: Define your base cases.** Your recursive algorithm has base cases, and you should state what they are.

**Step 3: Present your recursive cases.** Give a mathematical definition of your sub-problem in terms of “smaller” sub-problems. Make sure your recursive call uses the same number and type of parameters as in your original definition.

**Step 4: Prove correctness.** This will be an inductive proof that your algorithm does what it is supposed to. You should start by arguing that your base cases return the correct result, then for the inductive step, argue why your recursive cases combine to solve the overall problem.

**Step 5: Prove running time.** This is especially important for divide and conquer solutions, as there is usually an efficient brute-force solution, and the point of the question is to find something more efficient than brute-force.

## Common Running Times

Let  $T(n)$  denote the running time of your algorithm on input of size  $n$ .

- If  $T(n) = 2T(\frac{n}{2}) + O(n)$  and  $T(1) \in O(1)$  then  $T(n) \in O(n \log n)$ .
  - If  $T(n) = T(\frac{n}{2}) + O(1)$  and  $T(1) \in O(1)$  then  $T(n) \in O(\log n)$ .
  - If  $T(n) = T(\frac{n}{2}) + O(n)$  and  $T(1) \in O(1)$  then  $T(n) \in O(n)$  (seriously!)
  - If  $T(n) = 2T(\frac{n}{2}) + O(n^2)$  and  $T(1) \in O(1)$  then  $T(n) \in O(n^2)$ .
-

## Example: Mergesort

Given a list  $L$  of  $n$  elements, let  $\text{mergesort}(L)$  return the elements of  $L$  in non-decreasing sorted order. Then the following algorithm computes  $\text{mergesort}(L)$  in  $O(n \log n)$  time:

---

**Mergesort(L)****if**  $|L| = 1$  **then**    return  $L$ **else**    divide  $L$  into two halves  $A$  and  $B$  of size  $\lfloor n/2 \rfloor$  and  $\lceil n/2 \rceil$ , respectively.    let  $A^* = \text{mergesort}(A)$  and  $B^* = \text{mergesort}(B)$     let  $L^*$  be a new empty list    **while** both  $A^*$  and  $B^*$  are non-empty **do**        **if** first element of  $A^*$  is smaller than first element of  $B^*$  **then**            append the first element of  $A^*$  onto  $L^*$  and remove it from  $A^*$         **else**            append the first element of  $B^*$  onto  $L^*$  and remove it from  $B^*$         **end if**    **end while**    **if** one of  $A^*$  or  $B^*$  is non-empty **then**        append it to the end of  $L^*$     **end if**    return  $L^*$ **end if**

---

**Claim.** *The algorithm runs in  $O(n \log n)$  time.*

*Proof.* Let  $T(n)$  denote the running time of  $\text{mergesort}(L)$  when  $L$  is a list of  $n$  elements. Then

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

$$T(1) = O(1)$$

And therefore the running time is  $O(n \log n)$ . □

**Claim.** *Mergesort(L) correctly returns the elements of L in non-decreasing sorted order.*

*Proof.* Proof by induction on  $|L|$ .

□

---