Chapter 2: Divide-and-Conquer Algorithms

To solve a problem of size n:

- **Divide** the problem into subproblems, each of size $< n$.
- **Conquer**: Solve each subproblem recursively (and independantly of the other subproblems).
- **Combine/Merge** the solutions to the subproblems into a solution to the original problem.

Chapter 2: Divide-and-Conquer Algorithms

To solve a problem of size n:

- **Divide** the problem into subproblems, each of size $< n$.
- **Conquer**: Solve each subproblem recursively (and independantly of the other subproblems).
- **Combine/Merge** the solutions to the subproblems into a solution to the original problem.

For a given problem,

$\rightarrow$ How do we divide the problem, into how many subproblems?

$\rightarrow$ How to combine/merge?

Example: Merge Sort

To sort n numbers:

If $n \leq 1$: do nothing.

If $n \geq 2$: divide the n numbers arbitrarily into two sequences, both of size $n/2$, run Merge sort twice, once for each sequence.

Then merge the two sorted sequences into one sorted sequence.

Example: Merge Sort

To sort n numbers:

If $n \leq 1$: do nothing.

If $n \geq 2$: divide the n numbers arbitrarily into two sequences, both of size $n/2$, run Merge sort twice, once for each sequence.

Then merge the two sorted sequences into one sorted sequence.

What is the running time?

Let $T(n)$ be the running time of Merge Sort on an input of n numbers.

Let $T(n)$ be the running time of Merge Sort on an input of n numbers.

From CSI-2110, we know that the merge step takes $O(n)$ time.

Let $T(n)$ be the running time of Merge Sort on an input of n numbers.

From CSI-2110, we know that the merge step takes $O(n)$ time.

Hence, there is a constant $c > 0$ such that

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

Let $T(n)$ be the running time of Merge Sort on an input of n numbers.

From CSI-2110, we know that the merge step takes $O(n)$ time.

Hence, there is a constant $c > 0$ such that

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

What do we do with this?

Let $T(n)$ be the running time of Merge Sort on an input of n numbers.

From CSI-2110, we know that the merge step takes $O(n)$ time.

Hence, there is a constant $c > 0$ such that

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

What do we do with this?

We solve by *unfolding*!

Solve this recurrence by <u>unfolding</u>:

Assume $n = 2^K$, $c = 1$

$$T(n) \le n + 2T(n/2)$$

$$\le n + 2\cdot\left(\frac{n}{2} + 2\cdot T\left(\frac{n}{4}\right)\right)$$

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

$$\le 2n + 4\left(\frac{n}{4} + 2\cdot T\left(\frac{n}{8}\right)\right)$$

$$= 3n + 8T\left(\frac{n}{8}\right)$$

$$\le 3n + 8\left(\frac{n}{8} + 2\cdot T\left(\frac{n}{16}\right)\right)$$

$$= 4n + 16T\left(\frac{n}{16}\right)$$

$$\vdots$$

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

$$= Kn + n\cdot 1$$

$$= n\log_2(n) + n$$

since $\log_2(n) = \log_2(2^K) = K$

$$\le n\log_2(n) + n\cdot\log_2(n)$$

$$\le 2n\log_2(n)$$

In general, if we do not assume anything about the value of c, we find $T(n) \leq 2c\, n \log_2(n)$.

In general, if we do not assume anything about the value of c, we find $T(n) \leq 2c\, n \log_2(n)$.

So the running time of Merge Sort is $T(n) = O(n \log_2(n))$ if n is a power of two.

In general, if we do not assume anything about the value of c, we find $T(n) \leq 2c\, n \log_2(n)$.

So the running time of Merge Sort is $T(n) = O(n \log_2(n))$ if n is a power of two.

For a general n, we have

$$T(n) \leq \begin{cases} c & \text{if } n = 1, \\ T\left(\lfloor n/2 \rfloor\right) + T\left(\lceil n/2 \rceil\right) + c \cdot n & \text{if } n \geq 2, \end{cases}$$

In general, if we do not assume anything about the value of c, we find $T(n) \leq 2c\, n \log_2(n)$.

So the running time of Merge Sort is $T(n) = O(n \log_2(n))$ if n is a power of two.

For a general n, we have

$$T(n) \leq \begin{cases} c & \text{if } n = 1, \\ T\left(\lfloor n/2 \rfloor\right) + T\left(\lceil n/2 \rceil\right) + c \cdot n & \text{if } n \geq 2, \end{cases}$$

By induction, we can prove that $T(n) = O(n \log_2(n))$.