III. Divide-and-Conquer Recurrences and the Master Theorem
Divide-and-Conquer recurrences

- Divide-and-Conquer (DC) recurrence

\[ T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n) \]

where
- constants \( a \geq 1 \) and \( b > 1 \),
- function \( f(n) \) is nonnegative.
Divide-and-Conquer recurrences

▶ **Divide-and-Conquer (DC) recurrence**

\[
T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)
\]

where

▶ constants \( a \geq 1 \) and \( b > 1 \),
▶ function \( f(n) \) is nonnegative.

▶ **Example: the cost function of Merge Sort**

\[
T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n
\]

where

▶ \( a = 2 \) (the number of subproblems)
▶ \( b = 2 \) (\( n/2 \) is the size of subproblems)
▶ \( f(n) = n \) (the cost to divide and combine)
Solving DC recurrences by explicit substitution

- Explicit substitution can be illustrated by the following example

\[
T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n, \quad n = 2^k
\]

- By iterating the recurrence (i.e. explicit substitution), we have

\[
T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n
\]

\[
= 4 \cdot \left(4 \cdot T\left(\frac{n}{2}\right) + \frac{n}{2}\right) + n = 4^2 \cdot T\left(\frac{n}{2^2}\right) + 2n + n
\]

\[
= 4^3 \cdot T\left(\frac{n}{2^3}\right) + 2^2n + 2n + n
\]

\[
= \cdots
\]

\[
= 4^k \cdot T\left(\frac{n}{2^k}\right) + 2^{k-1}n + \cdots + 2n + n
\]

\[
= 4^k \cdot T(1) + (2^{k-1} + \cdots + 2 + 1)n
\]

\[
= 4^k \cdot T(1) + \frac{2^k - 1}{2 - 1}n
\]

\[
= n^2 \cdot T(1) + n(n - 1) = \Theta(n^2)
\]
The master theorem/method to solve DC recurrences

For the general DC recurrence, let $n = b^k$, then we have

$$T(n) = n^{\log_b a} \cdot T(1) + \sum_{j=0}^{k-1} a^j f\left(\frac{n}{b^j}\right)$$
The master theorem/method to solve DC recurrences

- For the general DC recurrence, let $n = b^k$, then we have

$$ T(n) = n^{\log_b a} \cdot T(1) + \sum_{j=0}^{k-1} a^j f\left(\frac{n}{b^j}\right) $$

- By carefully analyzing the terms in $T(n)$, we can provide asymptotic bounds on the growth of $T(n)$ in the following three cases.
The master theorem/method to solve DC recurrences

**Case 1:** If $n^{\log_b a}$ is polynomially larger than $f(n)$, i.e.,

$$
\frac{n^{\log_b a}}{f(n)} = \Omega(n^\epsilon) \quad \text{for some constant } \epsilon > 0,
$$

then

$$
T(n) = \Theta(n^{\log_b a}).
$$

Example: $T(n) = 7 \cdot T\left(\frac{n}{2}\right) + \Theta(n^2)$
The master theorem/method to solve DC recurrences

Case 2: If \( n^{\log_b a} \) and \( f(n) \) are on the same order, i.e.,

\[
f(n) = \Theta(n^{\log_b a}),
\]

then

\[
T(n) = \Theta(n^{\log_b a} \log n).
\]

Example: \( T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n) \)
Case 3: If $f(n)$ is polynomially greater than $n^{\log_b a}$, i.e.,

$$\frac{f(n)}{n^{\log_b a}} = \Omega(n^\epsilon) \quad \text{for some constant } \epsilon > 0$$

and $f(n)$ satisfies the regularity condition (see next slide), then

$$T(n) = \Theta(f(n)).$$

Example: $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n^3$
Remarks

1. $f(n)$ satisfies the *regularity condition* if

   $$a \cdot f\left(\frac{n}{b}\right) \leq c f(n)$$

   for some constant $c < 1$ and for all sufficient large $n$.

2. The proof of the master theorem is involved, shown in section 4.6, which we can safely skip for now.

3. The master method cannot solve every DC recurrences.