The divide-and-conquer design paradigm

1. Divide the problem (instance) into subproblems.
   - a subproblems, each of size \( n/b \)
2. Conquer the subproblems by solving them recursively.
3. Combine subproblem solutions.
   - Runtime for divide and combine is \( f(n) \)

The master method

The master method applies to recurrences of the form

\[
T(n) = a \cdot T(n/b) + f(n)
\]

where \( a \geq 1 \), \( b > 1 \), and \( f \) is asymptotically positive.
Master theorem (summary)

\[ T(n) = a T(n/b) + f(n) \]

**Case 1**: \( f(n) = O(n^{\log_b a - \varepsilon}) \)
\[ \Rightarrow T(n) = \Theta(n^{\log_b a}) . \]

**Case 2**: \( f(n) = \Theta(n^{\log_b a \log^k n}) \)
\[ \Rightarrow T(n) = \Theta(n^{\log_b a \log^{k+1} n}) . \]

**Case 3**: \( f(n) = \Omega(n^{\log_b a + \varepsilon}) \) and \( af(n/b) \leq cf(n) \) for some constant \( c < 1 \).
\[ \Rightarrow T(n) = \Theta(f(n)) . \]

Three common cases

Compare \( f(n) \) with \( n^{\log_b a} \):

1. \( f(n) = O(n^{\log_b a - \varepsilon}) \) for some constant \( \varepsilon > 0 \).
   - \( f(n) \) grows polynomially slower than \( n^{\log_b a} \) (by an \( n^\varepsilon \) factor).
   - **Solution**: \( T(n) = \Theta(n^{\log_b a}) . \)

2. \( f(n) = \Theta(n^{\log_b a \log^k n}) \) for some constant \( k \geq 0 \).
   - \( f(n) \) and \( n^{\log_b a} \) grow at similar rates.
   - **Solution**: \( T(n) = \Theta(n^{\log_b a \log^{k+1} n}) . \)

Examples

**Ex.** \( T(n) = 4T(n/2) + \sqrt{n} \)
\[ a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = \sqrt{n} . \]
**Case 1**: \( f(n) = O(n^{2-\varepsilon}) \) for \( \varepsilon = 1.5 \).
\[ \therefore T(n) = \Theta(n^2) . \]

**Ex.** \( T(n) = 4T(n/2) + n^2 \)
\[ a = 4, \ b = 2 \Rightarrow n^{\log_b a} = n^2; f(n) = n^2 . \]
**Case 2**: \( f(n) = \Theta(n^2 \log^0 n) \), that is, \( k = 0 \).
\[ \therefore T(n) = \Theta(n^2 \log n) . \]
Examples

Ex. $T(n) = 4T(n/2) + n^3$

$a = 4$, $b = 2$, $\Rightarrow n^{\log_b a} = n^2$; $f(n) = n^3$.

Case 3: $f(n) = \Omega(n^{a+\epsilon})$ for $\epsilon = 1$ and $4(n/2)^3 \leq cn^3$ (reg. cond.) for $c = 1/2$.

$\Rightarrow T(n) = \Theta(n^3)$.

Ex. $T(n) = 4T(n/2) + n^2/\log n$

$a = 4$, $b = 2$, $\Rightarrow n^{\log_b a} = n^2$; $f(n) = n^2/\log n$.

Master method does not apply. In particular, for every constant $\epsilon > 0$, we have $\log n \in o(n^\epsilon)$.

Example: merge sort

1. Divide: Trivial.
2. Conquer: Recursively sort 2 subarrays.
3. Combine: Linear-time merge.

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

\[
\begin{array}{ccc}
\text{# subproblems} & \text{subproblem size} & \text{work dividing and combining} \\
2n & n & 2n \\
\end{array}
\]

$n^{\log_b a} = n^{\log_2 2} = n^1 = n \Rightarrow \text{CASE 2 (} k = 0 \text{)}$ 

$\Rightarrow T(n) = \Theta(n \log n)$.

Recurrence for binary search

$T(n) = T(n/2) + \Theta(1)$

\[
\begin{array}{ccc}
\text{# subproblems} & \text{subproblem size} & \text{work dividing and combining} \\
\log n & n & 2n \\
\end{array}
\]

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \Rightarrow \text{CASE 2 (} k = 0 \text{)}$ 

$\Rightarrow T(n) = \Theta(\log n)$. 