The **Master Theorem** (verbatim from Cormen *et al.*, *Introduction to Algorithms*, 4th ed., MIT Press, 2022, pp. 102–103) states:

> Let $a > 1$ and $b > 1$ be constants, and let $f(n)$ be a driving function that is defined and non-negative on all sufficiently large reals. Define the recurrence $T(n)$ on $n \in \mathbb{N}$ by:
>
> $$T(n) = aT(n/b) + f(n)$$
>
> where $aT(n/b)$ actually means $a'T(\lfloor n/b \rfloor) + a''T(\lceil n/b \rceil)$ for some constants $a' \geq 0$ and $a'' \geq 0$ satisfying $a' + a'' = a$. Then the asymptotic behavior of $T(n)$ can be characterized as follows:
>
> (a) If there exists a constant $\epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
>
> (b) If there exists a constant $k \geq 0$ such that $f(n) = \Theta(n^{\log_b a} \log^k n)$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
>
> (c) If there exists a constant $\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and if $f(n)$ additionally satisfies the *regularity condition* $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

It can be used to determine the asymptotic behavior of an algorithm of which the running time is described by a recurrence of the form given above. This kind of recurrence arises when using a divide-and-conquer approach to solve a problem, i.e. where the problem is split into a subproblems, each of which is $1/b$ the size of the original problem, and $f(n)$ is the time taken to split the problem and combine the results of the subproblems.

Solution 1: Master Theorem

Use this theorem to determine the asymptotic behavior of the following recurrences:

(a) $T(n) = 2T(n/4) + 1$

(b) $T(n) = 2T(n/4) + \sqrt{n}$

(c) $T(n) = 2T(n/4) + \sqrt{n} \log^2 n$

(d) $T(n) = 2T(n/4) + n$

(e) $T(n) = 2T(n/4) + n^2$

Let $n_c = n^{\log_b a}$. In all cases, we have $a = 2$ and $b = 4$. We therefore have $n_c = n^{\log_4 2} = n^{1/2} = \sqrt{n}$. We need to compare $f(n)$ with n_c and test which of the conditions (if any) of the Master Theorem is satisfied.

(a) $T(n) = 2T(n/4) + 1$

- We compare $f(n) = 1$ with $n_c = \sqrt{n}$.
- We test Case 1 of the Master Theorem: Is $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$? $1 = O(n^{1/2-\varepsilon})$. Let $\varepsilon = 1/2$. Then $n^{1/2-1/2} = n^0 = 1$. So we check if $1 = O(1)$, which is true.
- Thus, Case 1 applies.
- Therefore, $T(n) = \Theta(n^{\log_b a}) = \Theta(\sqrt{n})$.

(b) $T(n) = 2T(n/4) + \sqrt{n}$

- We compare $f(n) = \sqrt{n}$ with $n_c = n^{1/2}$.
- We test Case 2 of the Master Theorem: Is $f(n) = \Theta(n^{\log_b a} \log^k n)$ for some $k \geq 0$? $n^{1/2} = \Theta(n^{1/2} \log^0 n)$. This is true for $k = 0$.

- Thus, Case 2 applies.
- Therefore, $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^{1/2} \log^{0+1} n) = \Theta(\sqrt{n} \log n)$.

(c) $T(n) = 2T(n/4) + \sqrt{n} \log^2 n$

- We compare $f(n) = \sqrt{n} \log^2 n$ with $n_c = n^{1/2}$.
- We test Case 2 of the Master Theorem: Is $f(n) = \Theta(n^{\log_b a} \log^k n)$ for some $k \geq 0$? $n^{1/2} \log^2 n = \Theta(n^{1/2} \log^2 n)$. This is true for $k = 2$.
- Thus, Case 2 applies.
- Therefore, $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^{1/2} \log^{2+1} n) = \Theta(\sqrt{n} \log^3 n)$.

(d) $T(n) = 2T(n/4) + n$

- We compare $f(n) = n$ with $n_c = n^{1/2}$.
- We test Case 3 of the Master Theorem: Is $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$? $n = \Omega(n^{1/2+\varepsilon})$. Let $\varepsilon = 1/2$. Then $n^{1/2+1/2} = n^1$. So we check if $n = \Omega(n)$, which is true.
- We also need to check the regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and sufficiently large n. $af(n/b) = 2 \cdot (n/4) = n/2$. We need $n/2 \leq c \cdot n$. This implies $1/2 \leq c$. Since we require $c < 1$, we can choose $c = 1/2$ (or indeed any $c \in [1/2, 1)$), so the condition holds.
- Thus, Case 3 applies.
- Therefore, $T(n) = \Theta(f(n)) = \Theta(n)$.

(e) $T(n) = 2T(n/4) + n^2$

- We compare $f(n) = n^2$ with $n_c = n^{1/2}$.
- We test Case 3 of the Master Theorem: Is $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$? $n^2 = \Omega(n^{1/2+\varepsilon})$. Let $\varepsilon = 3/2$. Then $n^{1/2+3/2} = n^2$. So we check if $n^2 = \Omega(n^2)$, which is true.
- We also need to check the regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and sufficiently large n. $af(n/b) = 2 \cdot (n/4)^2 = 2 \cdot (n^2/16) = n^2/8$. We need $n^2/8 \leq c \cdot n^2$. This implies $1/8 \leq c$. Since we require $c < 1$, we can choose $c = 1/8$ (or indeed any $c \in [1/8, 1)$), so the condition holds.
- Thus, Case 3 applies.
- Therefore, $T(n) = \Theta(f(n)) = \Theta(n^2)$.

Solution 2: Karatsuba Multiplication

The naive, elementary school algorithm for multiplying two large n-digit integers x and y takes $\Theta(n^2)$ time, since it involves n^2 multiplications of single digits, followed by $\Theta(n^2)$ additions of the results.

The *Karatsuba multiplication algorithm* is a more sophisticated method for multiplying two large integers x and y. Let n be the number of digits (in base 10, for this exercise) of the larger of the two numbers. The algorithm recursively splits the factors into two halves, $x = x_1 \cdot 10^m + x_0$ and $y = y_1 \cdot 10^m + y_0$, where $m = \lceil n/2 \rceil$. It makes use of the fact that $x \times y = (x_1 \cdot 10^m + x_0) \times (y_1 \cdot 10^m + y_0) = x_1 \times y_1 \cdot 10^{2m} + (x_1 \times y_0 + x_0 \times y_1) \cdot 10^m + x_0 \times y_0$. The three multiplications on the right-hand side can be done recursively. Addition and subtraction of two m-digit numbers takes $\Theta(m)$ time. Multiplication, division, or modulo operations of an m-digit number by a power of 10 can be done in $O(m)$ time; it is arguably constant time, but may involve copying digits to a result, which takes $\Theta(m)$ time.

We use the notation $|x|_{10}$ to denote the number of digits of x in base 10.

Algorithm 1 *KaratsubaMultiply*$(x : \mathbb{N},\ y : \mathbb{N})$

if $x < 10$ and $y < 10$ **then** ▷ base case: one-digit product
    return $x \times y$
end if
$n \leftarrow \max\big(|x|_{10},\ |y|_{10}\big)$
$m \leftarrow \lceil n/2 \rceil$
$B \leftarrow 10^m$ ▷ split base
$x_1 \leftarrow \lfloor x/B \rfloor$, $x_0 \leftarrow x \bmod B$ ▷ splitting x and y is $O(n)$ time, since it just manipulates the digits
$y_1 \leftarrow \lfloor y/B \rfloor$, $y_0 \leftarrow y \bmod B$
$p_0 \leftarrow \textsc{KaratsubaMultiply}(x_0,\ y_0)$
$p_2 \leftarrow \textsc{KaratsubaMultiply}(x_1,\ y_1)$
$p_1 \leftarrow \textsc{KaratsubaMultiply}(x_0 + x_1,\ y_0 + y_1)$ ▷ At this point, $p_1 = x_1 \times y_0 + x_0 \times y_1 + p_0 + p_2$
$p_1 \leftarrow p_1 - p_0 - p_2$
return $p_2 \times B^2 + p_1 \times B + p_0$

What is the asymptotic running time of this algorithm, in terms of the number of digits n of the larger of the two numbers, expressed as $\Theta(f(n))$?

Let $T(n)$ be the worst-case time complexity for multiplying two n-digit integers using the Karatsuba algorithm, where n is the maximum number of digits in the two input integers.

The algorithm proceeds as follows:

- **Base Case:** For $n = 1$, corresponding to $x < 10$ and $y < 10$ in the pseudocode, the multiplication is performed directly. This takes constant time, $\Theta(1)$.
- **Recursive Step (for larger n):**
  (a) The input numbers x and y are split into two halves. $m = \lceil n/2 \rceil$, so that $x = x_1 \cdot 10^m + x_0$ and $y = y_1 \cdot 10^m + y_0$. The numbers x_0, y_0 have at most m digits, and x_1, y_1 have at most $n - m = \lfloor n/2 \rfloor$ digits. This splitting process (which involves calculating n, m, and then performing divisions and modulo operations by 10^m) takes $\Theta(n)$ time, as stated in the problem.
  (b) Three recursive multiplications are performed:
    - $p_0 \leftarrow \textsc{KaratsubaMultiply}(x_0,\ y_0)$. This is a multiplication of numbers with at most $m = \lceil n/2 \rceil$ digits.
    - $p_2 \leftarrow \textsc{KaratsubaMultiply}(x_1,\ y_1)$. This is a multiplication of numbers with at most $n - m = \lfloor n/2 \rfloor$ digits.
    - To calculate $p_1 = x_1 y_0 + x_0 y_1$, Karatsuba's trick is to compute $S_x \leftarrow x_0 + x_1$ and $S_y \leftarrow y_0 + y_1$. These additions take $\Theta(m) = \Theta(n)$ time. The sums S_x, S_y have at most $m + 1 = \lceil n/2 \rceil + 1$ digits. Then, $p_{sum} \leftarrow \textsc{KaratsubaMultiply}(S_x,\ S_y)$ is computed. This is the third recursive call, on numbers of size at most $\lceil n/2 \rceil + 1$.

    Thus, there are three recursive calls. The sizes of the numbers involved are $\lceil n/2 \rceil$, $\lfloor n/2 \rfloor$, and $\lceil n/2 \rceil + 1$. For asymptotic analysis, these are all considered to be of size $n/2$.
  (c) The term $p_1 = x_1 y_0 + x_0 y_1$ is then obtained by $p_1 \leftarrow p_{sum} - p_0 - p_2$. The numbers p_0, p_2, and p_{sum} are results of multiplying numbers with $\sim n/2$ digits, so they can have up to $2(\lceil n/2 \rceil + 1) \approx n$ digits. These two subtractions take $\Theta(n)$ time.
  (d) The final result is assembled as $p_2 \cdot 10^{2m} + p_1 \cdot 10^m + p_0$. Multiplying by powers of 10 (10^m or 10^{2m}) corresponds to digit shifts. Since p_0, p_1, p_2 have $\Theta(n)$ digits, these shifts take $\Theta(n)$ time. The two final additions also involve $\Theta(n)$-digit numbers and thus take $\Theta(n)$ time.

The total time spent on non-recursive operations (splitting, additions, subtractions, and shifts) is $\Theta(n)$. Therefore, the recurrence relation for the time complexity $T(n)$ is:

$$T(n) = 3T(n/2) + \Theta(n)$$

with the base case $T(1) = \Theta(1)$.

We apply the Master Theorem to solve this recurrence:

- Identify $a = 3$ (the number of recursive subproblems), $b = 2$ (the factor by which the subproblem size is reduced), and $f(n) = \Theta(n)$ (the cost of the work done outside the recursive calls).
- Calculate $n^{\log_b a} = n^{\log_2 3}$.
- We compare $f(n) = n$ with $n^{\log_2 3}$. Since $\log_2 3 \approx 1.585$, it is clear that $n^{\log_2 3}$ grows faster than n.
- We check Case 1 of the Master Theorem: Is $f(n) = O(n^{\log_b a-\varepsilon})$ for some constant $\varepsilon > 0$? Let $\varepsilon = \log_2 3 - 1$. Since $\log_2 3 > 1$, ε is a positive constant (approx 0.585). Then $n^{\log_b a-\varepsilon} = n^{\log_2 3-(\log_2 3-1)} = n^1 = n$. The condition requires $f(n) = O(n)$. Since $f(n) = \Theta(n)$, this condition is satisfied.
- Thus, Case 1 of the Master Theorem applies.
- Therefore, the asymptotic running time of the Karatsuba algorithm is $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3})$.

Solution 3: Secant Method

The *secant method* is a method for finding the roots of a function f, when the derivative f' is not known. It works similar to Newton's method, but instead of using the derivative f', it uses a secant line through two points x_{n-1} and x_n on the function f to approximate the derivative for finding the next point x_{n+1}.

x_{n+1} is then given by:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

The secant method can thus be used to determine the value of $\sqrt{2}$ by finding the root of $f(x) = x^2 - 2$, using the iteration:

$$x_{t+1} = x_t - (x_t^2 - 2)\frac{x_t - x_{t-1}}{x_t^2 - x_{t-1}^2}$$

with initial values $x_0 = 1$ and $x_1 = 2$.

Prove that the iteration converges to $\sqrt{2}$ and determine the Q-order of convergence p.

Part 1: Proof of Convergence to $\sqrt{2}$

a. Simplify the iteration formula

The denominator $x_t^2 - x_{t-1}^2$ can be factored as $(x_t - x_{t-1})(x_t + x_{t-1})$. Assuming $x_t \neq x_{t-1}$ (which is true for initial steps and generally unless convergence is achieved) and $x_t + x_{t-1} \neq 0$ (true for our positive iterates), the formula becomes:

$$x_{t+1} = x_t - \frac{x_t^2 - 2}{x_t + x_{t-1}}$$

Combining the terms:

$$x_{t+1} = \frac{x_t(x_t + x_{t-1}) - (x_t^2 - 2)}{x_t + x_{t-1}} = \frac{x_t^2 + x_t x_{t-1} - x_t^2 + 2}{x_t + x_{t-1}} = \frac{x_t x_{t-1} + 2}{x_t + x_{t-1}}$$

This simplified form will be used for analysis. The initial values are $x_0 = 1$ and $x_1 = 2$. For illustration: $x_2 \approx 1.3333$; $x_3 = 1.4$; $x_4 \approx 1.414634$. The true value is $\sqrt{2} \approx 1.41421356$.

b. Limit of the sequence

(Not strictly necessary for the proof of convergence, since further down we show that the difference between x_t and $\sqrt{2}$ converges to 0, but it's a good sanity check.)

If the sequence $\{x_t\}$ converges to a limit α, then taking $t \to \infty$ in the simplified iteration:

$$\alpha = \frac{\alpha \cdot \alpha + 2}{\alpha + \alpha} = \frac{\alpha^2 + 2}{2\alpha}$$

This implies $2\alpha^2 = \alpha^2 + 2$, which simplifies to $\alpha^2 = 2$. Since $x_0 = 1 > 0$ and $x_1 = 2 > 0$, and the recurrence $x_{t+1} = \frac{x_t x_{t-1}+2}{x_t+x_{t-1}}$ preserves positivity if x_t, x_{t-1} are positive, all x_t in the sequence must be positive. Therefore, if the limit exists, it must be $\alpha = \sqrt{2}$ (as opposed, to, say, $-\sqrt{2}$).

c. Error recurrence relation

Let $\epsilon_t = x_t - \sqrt{2}$ be the error at iteration t. So $x_t = \sqrt{2} + \epsilon_t$. Substituting this into the simplified iteration formula:

$$\sqrt{2} + \epsilon_{t+1} = \frac{(\sqrt{2} + \epsilon_t)(\sqrt{2} + \epsilon_{t-1}) + 2}{(\sqrt{2} + \epsilon_t) + (\sqrt{2} + \epsilon_{t-1})}$$

$$\epsilon_{t+1} = \frac{2 + \sqrt{2}\epsilon_t + \sqrt{2}\epsilon_{t-1} + \epsilon_t\epsilon_{t-1} + 2}{2\sqrt{2} + \epsilon_t + \epsilon_{t-1}} - \sqrt{2}$$

$$\epsilon_{t+1} = \frac{\epsilon_t\epsilon_{t-1}}{2\sqrt{2} + \epsilon_t + \epsilon_{t-1}}$$

The denominator is $x_t + x_{t-1}$. So, $\epsilon_{t+1} = \frac{\epsilon_t\epsilon_{t-1}}{x_t + x_{t-1}}$.

d. Convergence of errors

Assume that $x_t, x_{t-1} \in [1, 2]$. What is the range of possible values for x_{t+1}?

$$x_{t+1} = \frac{x_t x_{t-1} + 2}{x_t + x_{t-1}} \geq \frac{x_t + 2}{x_t + x_{t-1}} \geq \frac{x_t + 2}{x_t + 2} = 1.$$

$$x_{t+1} = \frac{x_t x_{t-1} + 2}{x_t + x_{t-1}} \leq \frac{2x_t + 2}{x_t + x_{t-1}} \leq \frac{2x_t + 2}{x_t + 1} = 2.$$

Since $x_0, x_1 \in [1, 2]$, by induction, $x_t \in [1, 2]$ for all $t \geq 0$.

Since $x_t + x_{t-1} \geq 2$, we have $|\epsilon_{t+1}| \leq \frac{|\epsilon_t||\epsilon_{t-1}|}{2}$.

It also follows from $x_t \in [1, 2]$ that all $|\epsilon_t| = |x_t - \sqrt{2}| \leq 1$.

Hence, $|\epsilon_{t+1}| \leq \frac{|\epsilon_t||\epsilon_{t-1}|}{2} \leq \frac{|\epsilon_t|}{2}$, and therefore $\lim_{t\to\infty} |\epsilon_t| = 0$ and $x_t \to \sqrt{2}$.

Part 2: Q-order of Convergence

The error recurrence is $\epsilon_{t+1} = \frac{\epsilon_t\epsilon_{t-1}}{x_t + x_{t-1}}$.

We look for p such that for some $0 < c < \infty$,

$$\limsup_{t\to\infty} \frac{|\epsilon_{t+1}|}{|\epsilon_t|^p} = c. \tag{1}$$

Two slightly different approaches are given below.

Let's assume the limit of $\frac{|\epsilon_{t+1}|}{|\epsilon_t|^p}$ exists, in which case the lim sup is the same as the limit.

Substituting $\epsilon_{t+1} = \frac{\epsilon_t\epsilon_{t-1}}{x_t + x_{t-1}}$, this becomes

$$\lim_{t\to\infty} \frac{|\epsilon_t\epsilon_{t-1}|}{|\epsilon_t|^p|x_t + x_{t-1}|} = c,$$

where we divide by $|\epsilon_t|$ in the numerator and denominator, and plug in $|x_t + x_{t-1}| \to 2\sqrt{2}$ as $t \to \infty$.

$$\lim_{t\to\infty} \frac{|\epsilon_{t-1}|}{|\epsilon_t|^{p-1} 2\sqrt{2}} = c$$

$$\lim_{t\to\infty} \frac{|\epsilon_{t-1}|}{|\epsilon_t|^{p-1}} = 2\sqrt{2}c.$$

Some rearrangement:

$$\lim_{t\to\infty} \frac{|\epsilon_{t-1}|}{|\epsilon_t|^{p-1}} = \lim_{t\to\infty} \frac{|\epsilon_t|^{1-p}}{|\epsilon_{t-1}|^{-1}} = \lim_{t\to\infty} \left(\frac{|\epsilon_t|}{|\epsilon_{t-1}|^{\frac{1}{p-1}}}\right)^{1-p} = \left(\lim_{t\to\infty} \frac{|\epsilon_t|}{|\epsilon_{t-1}|^{\frac{1}{p-1}}}\right)^{1-p} = 2\sqrt{2}c.$$

This works assuming p is not 1. The penultimate step is the reason why we need the limit, not the lim sup. We now have

$$\lim_{t\to\infty} \frac{|\epsilon_t|}{|\epsilon_{t-1}|^{\frac{1}{p-1}}} = \left(2\sqrt{2}c\right)^{\frac{1}{1-p}}. \tag{2}$$

If the limit exists and is nonzero, the exponent is unique, since if $\limsup_{t\to\infty} \frac{|\epsilon_{t+1}|}{|\epsilon_t|^p} = c$, then $\limsup_{t\to\infty} \frac{|\epsilon_{t+1}|}{|\epsilon_t|^{p+x}} = \limsup_{t\to\infty} \frac{|\epsilon_{t+1}|}{|\epsilon_t|^p|\epsilon_t|^x} = c \limsup_{t\to\infty} |\epsilon_t|^x$, which is 0 for $x > 0$ and ∞ for $x < 0$.

Therefore the exponent needs to be equal in (1) and (2) (shifting the index $t-1$ to t), so we get $p = \frac{1}{p-1}$.

Multiplying by $p - 1$ (assuming $p \neq 1$):

$$p^2 - p = 1$$

$$p^2 - p - 1 = 0$$

This is a quadratic equation for p. Using the quadratic formula $p = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$:

$$p = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

Since the order of convergence p must be positive, we take the positive root:

$$p = \frac{1 + \sqrt{5}}{2}$$

This value is the golden ratio, often denoted by ϕ, and is approximately 1.618.

This is the general Q-order of convergence for the secant method, not just for the square root problem.