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.

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

Exercise 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 $\Theta(f(n))$?

Exercise 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.