1. Show that if $f(n)$ and $g(n)$ are monotonically increasing functions, then so are the functions $f(n) + g(n)$ and $f(g(n))$, and if $f(n)$ and $g(n)$ are in additional nonnegative, then $f(n) \cdot g(n)$ is monotonically increasing.

A function f is *monotonically increasing* if $n \leq m$ implies $f(n) \leq f(m)$. Suppose that f and g are monotonically increasing. Then $n \leq m$ implies $f(n) \leq f(m)$ and $g(n) \leq g(m)$, and hence $f(n) + g(n) \leq f(m) + g(m)$ and so $f(n) + g(n)$ is monotonically increasing.

Since g is monotonically increasing, if $n \leq m$ then $g(n) \leq g(m)$. Let $n' = g(n)$ and $m' = g(m)$, so that $n' \leq m'$. Since f is monotonically increasing and $n' \leq m'$, we have $f(n') \leq f(m')$. Hence $f(g(n)) \leq f(g(m))$ and $f(g(n)$ is monotonically increasing.

Suppose that f and g are nonnegative, so that $f(n) \geq 0$ and $g(n) \geq 0$. Furthermore, f and g are monotonically increasing so that $n \leq m$ implies $f(n) \leq f(m)$ and $g(n) \leq g(m)$. Because f and g are nonnegative we can multiply the inequalities to obtain $f(n) \cdot g(n) \leq f(m) \cdot g(m)$. Hence $f(n) \cdot g(n)$ is monotonically increasing.

2. Use the definition of O-notation to show that $T(n) = n^{O(1)}$ if and only if there exists a constant $k > 0$ such that $T(n) = O(n^k)$.

Suppose that $T(n) = n^{O(1)}$. The set $O(1)$ is defined as $O(1) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq f(n) \leq c \cdot 1 = c$ for all $n \geq n_0\}$. If other words, we have $f(n) = O(1)$ only if f is bounded above by some positive constant c. Clearly, for any positive constant $k > 0$ we have $k = O(1)$, since we can always choose another positive constant $c_1 > 0$ such that $0 < k \leq c_1$. Since $k = O(1)$, we can write $T(n) = n^k$.

Since $T(n) = n^k$, we have that $T(n) = O(n^k)$, since we can find a positive constants c and n_0 such that $0 \leq T(n) \leq c \cdot n^k$ for all $n \geq n_0$. Specifically, choose $n_0 = 1$ and $c = 2$ to establish that $T(n) = O(n^k)$.

This establishes that $T(n) = n^{O(1)} \Rightarrow T(n) = O(n^k)$.

Suppose that $T(n) = O(n^k)$, where $k > 0$. Then we can find positive constants c_1 and n_1 such that $0 \leq T(n) \leq c_1 \cdot n_1^k$ for all $n \geq n_1$. To demonstrate that a function f is $f = O(1)$, we must find positive constants c_2 and n_2 such that $0 \leq f(n) \leq c_2 \cdot 1 = c_2$ for all $n \geq n_2$. Since $T(n) = O(n^k)$ we know that $T(n) \leq c_1 \cdot n^k$, and to show that $T(n) = n^{O(1)}$ we must find c_2 and n_2 such that $T(n) \leq n^{c_2}$ for all $n \geq n_2$.

Let $c_2 = k + 1$, hence we can write $c_1 \cdot n^k \leq n^{k+1}$. Dividing both sides by n^k, we have $c_1 \leq n$. Hence, if we choose positive constants $c_2 = k + 1$ and $n_2 \geq c_1$, we have that $T(n) = n^{O(1)}$.

This establishes that $T(n) = O(n^k) \Rightarrow T(n) = n^{O(1)}$.

Hence we conclude that $T(n) = n^{O(1)} \iff T(n) = O(n^k)$.

3. Prove equation (2.9).

Equation (2.9) is given as:

$$a^{\log_b n} = n^{\log_b a}$$

We can make use of the identity $a = b^{\log_b a}$ as follows:

$$a^{\log_b n} = b^{\log_b a \cdot \log_b n} = b^{\log_b n \cdot \log_b a} = n^{\log_b a}$$

4. Prove that $\lg(n!) = \Theta(n \lg n)$ and that $n! = o(n^n)$.

For the first part, if $n \geq 1$ we have:

$$\lg(n!) = \sum_{k=1}^{n} \lg k \leq n \lg n$$

Hence by choosing $n \geq 1$ and $c \geq 1$, we have $\lg(n!) = O(n \lg n)$. To prove that $\lg(n!) = \Omega(n \lg n)$, we can perform a similar calculation, utilizing only the second half of the summation:

$$\frac{1}{2} n \lg n - \frac{1}{2} n = \left(\frac{n}{2}\right) \cdot \lg \frac{n}{2} \leq \sum_{k=n/2}^{n} \lg k \leq \sum_{k=1}^{n} \lg k = \lg(n!)$$

Suppose we have $n \geq 4$, then $\frac{1}{4} n \lg n \leq \frac{1}{2} n \lg n - \frac{1}{2} n$ and $\frac{1}{4} n \lg n \leq \lg(n!)$. Hence by choosing $n \geq 4$ and $c \leq 1/4$, we have $\lg(n!) = \Omega(n \lg n)$.

We can therefore establish $\lg(n!) = \Theta(n \lg n)$ by choosing $c_1 = 1/8$, $c_2 = 2$ and $n_0 = 8$ so that $0 \leq c_1 \cdot n \lg n \leq \lg(n!) \leq c_2 \cdot n \lg n$ for all $n \geq n_0$.

For the second part, let $f(n) = n!$ and $g(n) = n^n$ and consider $\lim_{n \to \infty} \frac{f(n)}{g(n)}$. If this limit tends to zero, we know that $f(n) = o(g(n))$. For $n \geq 1$, we have:

$$\frac{n!}{n^n} \leq \frac{1}{n}$$

$$\lim_{n \to \infty} \frac{n!}{n^n} \leq \lim_{n \to \infty} \frac{1}{n} = 0$$

Hence we conclude that $n! = o(n^n)$.

5. Is the function $\lceil \lg n \rceil!$ polynomially bounded? Is the function $\lceil \lg \lg n \rceil!$ polynomially bounded?

A function is polynomially bounded if $f(n) = O(n^d)$ for some constant d. We must be able to find positive constants c and n_0 such that $0 \le f(n) \le c \cdot n^d$ for all $n \ge n_0$. Let $f(n) = \lceil \lg n \rceil! = \prod_{k=1}^{\lceil \lg n \rceil} k$, so that:

$$\prod_{k=\lceil \lg n \rceil/2}^{\lceil \lg n \rceil} k \le \prod_{k=1}^{\lceil \lg n \rceil} k = f(n)$$

$$\sum_{k=\lceil \lg n \rceil/2}^{\lceil \lg n \rceil} \lg k = \lg \left(\prod_{k=\lceil \lg n \rceil/2}^{\lceil \lg n \rceil} k \right) \le \lg(f(n))$$

$$\frac{1}{2} \lg(n) \cdot \lg \left(\frac{1}{2} \lg n \right) \le \frac{1}{2} \lceil \lg n \rceil \cdot \lg \left(\frac{1}{2} \lceil \lg n \rceil \right) \le \lg(f(n))$$

$$\lg \left(\sqrt{n} \right) \cdot \lg \lg \left(\sqrt{n} \right) \le \lg(f(n))$$

$$\left(2^{\lg \sqrt{n}} \right)^{\lg \lg \sqrt{n}} \le f(n)$$

$$\sqrt{n}^{\lg \lg \sqrt{n}} \le f(n)$$

If we merely had $\sqrt{n} \le f(n)$, then we could say that f is bounded from below by a polynomial n^k with $k = 1/2$. Moreover, the function $\lg \lg \sqrt{n}$ indeed grows very, very slowly, but the fact is that with a lower bound of $\sqrt{n}^{\lg \lg \sqrt{n}}$ there is no polynomial that we can use to bound this function.

Since f is larger than this function, we know that $f(n)$ grows faster than any polynomial function and therefore is not polynomially bounded.

Now consider the function $f(n) = \lceil \lg \lg n \rceil!$. Without the loss of generality, we can restrict attention to values of n where $n = 2^{2^k}$, where $k \in \mathbb{N}$ since if $n_1 = 2^{2^{k-1}}$ and $n_2 = 2^{2^k}$ then for all $n \in \{n_1 + 1, n_1 + 2, ..., n_2\}$, the function $f(n) = \lceil \lg \lg n \rceil!$ will evaluate to the same value. For the same reason, we can disregard the ceiling function in the analysis that follows.

To show that f is polynomially bounded, we must find positive constants c and n_0 such that $0 \le f(n) \le c \cdot n^d$ for some constant d and for all $n \ge n_0$. Since $n = 2^{2^k}$, we have $\lg \lg n = k$ and hence:

$$f(n) = \lceil \lg \lg n \rceil! = k! \le k^k \le c \cdot n^d$$

Taking the logarithm of both sides:

$$k \lg k \leq \lg c + d \lg n$$

Since $\lg n = 2^k$, we have:

$$k \lg k \leq \lg c + d \cdot 2^k$$

The expression $k \lg k \leq 2^k$ is true for all $k \geq 1$, so the expression above is true for all $c > 1$ and $d > 1$. For the sake of specificity, we select $c = 2$, $d = 2$ and $n_0 = 2$, so that $f(n) = \lceil \lg \lg n \rceil! \leq 2n^2$ for all $n \geq n_0$, and hence $\lceil \lg \lg n \rceil!$ is polynomially bounded.

6. Which is asymptotically larger: $\lg(\lg^* n)$ or $\lg^*(\lg n)$?

$\lg^*(\lg n)$ is asymptotically larger than $\lg(\lg^*(n))$.

Let $\lg^*(n) = m$, then $\lg(\lg^*(n)) = \lg(m)$. Moreover, the iterated logarithm function simply applies the $\lg(n)$ function repeatedly, i times, while the $\lg^*$ function applies $\lg(n)$ repeatedly until its value is equal to or less than one. Hence, by definition, $\lg^*(\lg(n)) = m - 1$.

$$\lg^*(\lg(n)) \rightarrow m - 1$$
$$\lg(\lg^*(n)) \rightarrow \lg(m)$$

Since m grows asymptotically faster than $\lg(m)$, so we conclude that $\lg^*(\lg(n))$ grows faster than $\lg(\lg^*(n))$.

The following Python code implements the $\log^*$ algorithm:

```python
import math

def log2(n):
  return math.log(n)/math.log(2)

def logi(n,i):
  if i == 0:
    return n
  return log2(logi(n,i-1))

def log_star(n):
  def log_star_iter(k):
    value = logi(n,k)
    if value <= 1.0:
      return k
    return log_star_iter(k+1)
  return log_star_iter(0)
```

7. Prove by the induction that the ith Fibonacci number satisfies the equality $F_i = (\phi^i - \hat{\phi}^i)/\sqrt{5}$, where ϕ is the golden ratio and $\hat{\phi}$ is its conjugate.

The *golden ratio* ϕ and its conjugate $\hat{\phi}$ are given by:

$$\phi = \frac{1+\sqrt{5}}{2}, \hat{\phi} = \frac{1-\sqrt{5}}{2}$$

The Fibonacci sequence is given by:

$$F = \{0, 1, 1, 2, 3, 5, 8, 13, 21, ...\}$$

Suppose we have $i = 0$, then $F_0 = 0$, and $\phi^0 = 1$ and $\hat{\phi}^0 = 1$, so $\phi^0 - \hat{\phi}^0 = 0$ and the condition is satisfied. Suppose next that we have $i = 1$, then $F_1 = 1$, and $\phi^1 = (1+\sqrt{5})/2$ and $\hat{\phi}^1 = (1-\sqrt{5})/2$, so $(\phi^1 - \hat{\phi}^1)/\sqrt{5} = (2\sqrt{5})/(2\sqrt{5})$ $= 1$ and the condition is satisfied.

Suppose next that the condition is satisfied for the nth Fibonacci number:

$$F_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}}$$

The nth Fibonacci number is defined by:

$$F_n = F_{n-1} + F_{n-2}$$

We proceed by induction, and demonstrate that the condition holds for $n+1$ given that the condition holds for n and $n-1$:

$$F_{n+1} = F_n + F_{n-1}$$

$$F_{n+1} = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}} + \frac{\phi^{n-1} - \hat{\phi}^{n-1}}{\sqrt{5}}$$

$$F_{n+1} = \frac{\phi^{n-1}(\phi+1) - \hat{\phi}^{n-1}(\hat{\phi}+1)}{\sqrt{5}}$$

We have $\phi + 1 = (3+\sqrt{5})/2 = (1+2\sqrt{5}+5)/4 = \{(1+\sqrt{5})/2\}^2 = \phi^2$.

Likewise $\hat{\phi} + 1 = (3-\sqrt{5})/2 = (1-2\sqrt{5}+5)/4 = \{(1-\sqrt{5})/2\}^2 = \hat{\phi}^2$.

Hence we can write:

$$F_{n+1} = \frac{\phi^{n-1} \cdot \phi^2 - \hat{\phi}^{n-1} \cdot \hat{\phi}^2}{\sqrt{5}} = \frac{\phi^{n+1} - \hat{\phi}^{n+1}}{\sqrt{5}}$$

8. Prove that for $i \geq 0$, the $(i+2)$nd Fibonacci number satisfies $F_{i+2} \geq \phi^i$.

The *golden ratio* is given by:

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

The Fibonacci sequence is given by:

$$F = \{0, 1, 1, 2, 3, 5, 8, 13, 21, ...\}$$

Suppose that we have $i = 0$, then $F_2 = 1$ and $\phi^0 = 1$, and the condition is satisfied. Suppose next that we have $i = 1$, then $F_3 = 2$ and $\phi^1 = (1+\sqrt{5})/2$ ≈ 1.618, which establishes the result.

The $(n+2)$the Fibonacci number is given by:

$$F_{n+2} = F_{n+1} + F_n$$

and by induction we have that $F_{n+1} \geq \phi^{n-1}$ and $F_n \geq \phi^{n-2}$, hence:

$$F_{n+2} \geq \phi^{n-1} + \phi^{n-2} = \phi^{n-2}(\phi + 1)$$

In the previous problem we showed that $\phi + 1 = \phi^2$, hence:

$$F_{n+2} \geq \phi^{n-2} \cdot \phi^2 = \phi^n$$