Module 1: Problem Set 1

shenyu li

September 2024

1 0.1

a. Given the functions $f(n) = n - 100$ and $g(n) = n - 200$, both are asymptotically $\mathcal{O}(n)$. Thus, the relationship between $f(n)$ and $g(n)$ falls under the category of $f(n) = \Theta(g(n))$, as their growth rates are essentially equivalent.

- $f(n) = \mathcal{O}(g(n))$ indicates that the growth of $f(n)$ is not asymptotically larger than $g(n)$.
- $f(n) = \Omega(g(n))$ shows that $f(n)$ is not asymptotically smaller than $g(n)$, meaning that they grow at comparable rates.

Thus, the functions exhibit the relation $f(n) = \Theta(g(n))$, as their difference becomes negligible at larger scales, and can also be rewritten as:

$$n - 100 = \Theta(n - 200)$$

b. Given the functions $f(n) = n^{1/2}$ and $g(n) = n^{2/3}$, it can be observed that $n^{1/2} < n^{2/3}$ when comparing the powers.

The function $f(n) = O(g(n))$ holds because when comparing the growth rates, the computational speed of $g(n)$ is greater than that of $f(n)$.

- The Big-O notation $f(n) = O(g(n))$ indicates that the function $g(n)$ grows faster than $f(n)$ as n increases.
- According to simplification rules for such functions, $n^{1/2}$ is dominated by $n^{2/3}$, which means $g(n)$ is asymptotically superior to $f(n)$.

Thus, the correct relationship is $f(n) = O(g(n))$, and this can be written as:

$$n^{1/2} = O(n^{2/3})$$

c. Given the functions $f(n) = 100n + \log n$ and $g(n) = n + (\log n)^2$, we observe that both functions are dominated by their respective linear terms. This implies that their asymptotic growth is comparable, meaning that both are $\mathcal{O}(n)$.

- $f(n) = \mathcal{O}(g(n))$ states that the growth of $f(n)$ is bounded above by $g(n)$. In other words, the computational speed of $f(n)$ does not exceed that of $g(n)$ as n grows larger.

- $f(n) = \Omega(g(n))$ means that the growth of $f(n)$ is bounded below by $g(n)$. This indicates that $g(n)$ does not grow significantly faster than $f(n)$ as n increases.

- Thus, the functions are said to be $f(n) = \Theta(g(n))$, which means they grow at the same rate asymptotically, up to constant factors.

To analyze the growth of these functions more precisely: - In the function $f(n) = 100n + \log n$, the term $100n$ is linear, whereas $\log n$ grows much more slowly. As n increases, the logarithmic term becomes negligible compared to the linear term. Therefore, $f(n)$ is dominated by the term $100n$, meaning the overall growth rate is driven by $100n$. - Similarly, in the function $g(n) = n + (\log n)^2$, the linear term n dominates the growth, since any polynomial grows faster than a logarithmic term squared. Hence, $g(n)$ is asymptotically dominated by n.

Because both functions are primarily driven by their linear terms, we can conclude that $f(n)$ and $g(n)$ grow at the same rate asymptotically. Therefore, we have:

$$f(n) = \Theta(g(n))$$

Thus, the functions can be rewritten as:

$$100n + \log n = \Theta\left(n + (\log n)^2\right)$$

d. The functions $f(n) = n \log n$ and $g(n) = 10n \log 10n$.

- $f = O(g)$ means that the growth of $f(n)$ is at most as fast as $g(n)$.

- $f = \Omega(g)$ means that $f(n)$ grows at least as fast as $g(n)$.

In this case, both functions fall under $O(n \log n)$, which means their growth rates are the same. Therefore, we can express the relationship as:

$$f(n) = \Theta(g(n))$$

where Θ represents that both functions have the same order of growth.

Any polynomial function will always dominate a logarithmic function in terms of growth rate. Therefore, the two functions have the same complexity, which can be written as:

$$n \log n = \Theta(10n \log 10n)$$

e. Compare the functions $f(n) = \log 2n$ and $g(n) = \log 3n$. Based on the comparison:

- $f = O(g)$ means that the growth of $f(n)$ does not exceed that of $g(n)$.
- $f = \Omega(g)$ indicates that the growth of $f(n)$ is not slower than $g(n)$.

Since both functions have the same complexity and belong to $O(n \log n)$, can write the following relationship:

$$\log 2n = \Theta(\log 3n)$$

Compare $f(n) = 10 \log n$ and $g(n) = \log(n^2)$. Similarly, the complexity of these two functions matches, and have:

- $f = O(g)$ implies $f(n)$ grows no faster than $g(n)$.
- $f = \Omega(g)$ means that $f(n)$ grows at least as fast as $g(n)$.

Thus, the complexity of these functions is the same, which can be expressed as:

$$10 \log n = \Theta(\log(n^2))$$

f. The two functions $f(n) = 10 \log n$ and $g(n) = \log(n^2)$ have identical growth rates, as both fall under the category $O(n \log n)$. This means that the complexity of both functions is comparable, which can be expressed as:

$$f(n) = \Theta(g(n))$$

The following observations hold:

- $f(n) = O(g(n))$ signifies that the growth of $f(n)$ does not exceed that of $g(n)$.
- $f(n) = \Omega(g(n))$ indicates that the growth of $f(n)$ is not slower than $g(n)$.

Thus, conclude that the relationship between the two functions is:

$$10 \log n = \Theta(\log(n^2))$$

g. G compares the complexity between two functions $f(n) = n^{1.01}$ and $g(n) = n \log^2 n$.

The comparison reveals that $f(n)$ grows faster than $g(n)$, and thus, we have:

$$f(n) = \Omega(g(n))$$

This means that the growth rate of $f(n)$ is not smaller than that of $g(n)$. The following points further clarify the comparison:

- If both functions are divided by n, a more detailed numerical comparison can be made, but this will take more time.
- According to simplification rules, functions with higher powers (like $f(n) = n^{1.01}$) typically grow faster than logarithmic functions (such as $g(n) = n \log^2 n$).

Thus, the final relationship between the two functions is:

$$n^{1.01} = \Omega(n \log^2 n)$$

h. Given the functions $f(n) = \frac{n^2}{\log n}$ and $g(n) = n(\log n)^2$, determine that:

$$f(n) = \Omega(g(n))$$

This means that the growth rate of $f(n)$ is not dominated by $g(n)$. Additional considerations include:

- Dividing both functions by $n/\log n$ allows a more detailed comparison, but such a comparison itself requires more time.
- According to rules of simplification, functions with higher powers (like $f(n)$) grow faster, so $f(n)$ grows faster than $g(n)$.

Thus, the relationship between the two functions can be written as:

$$\frac{n^2}{\log n} = \Omega(n(\log n)^2)$$

i. Given the functions $f(n) = n^{0.1}$ and $g(n) = (\log n)^{10}$, the comparison leads to the conclusion that:

$$f(n) = \Omega(g(n))$$

This indicates that the growth rate of $f(n)$ is not dominated by $g(n)$. Further considerations include:

- Dividing both functions by n allows for more detailed comparisons, although such comparisons require more time.
- Simplification rules suggest that functions with power values (like $f(n)$) typically grow faster than logarithmic functions (like $g(n)$).

Thus, the final relationship between the two functions is expressed as:

$$n^{0.1} = \Omega((\log n)^{10})$$

j. Given the functions $f(n) = (\log n)^{\log n}$ and $g(n) = \frac{n}{\log n}$, the comparison reveals:

$$f(n) = \Omega(g(n))$$

This indicates that $f(n)$ grows faster than $g(n)$. Thus, express the relationship as:

$$(\log n)^{\log n} = \Omega\left(\frac{n}{\log n}\right)$$

k. For the functions $f(n) = \sqrt{n}$ and $g(n) = (\log n)^3$, the comparison yields:

$$f(n) = \Omega(g(n))$$

This shows that $f(n)$ grows faster than $g(n)$. Therefore, the relationship can be written as:

$$\sqrt{n} = \Omega\left((\log n)^3\right)$$

i. Given the functions $f(n) = n^{1/2}$ and $g(n) = 5^{\log_2 n}$:

$$f(n) = O(g(n))$$

This means that the growth rate of $g(n)$ dominates $f(n)$.

- The function $g(n)$ can be rewritten as $g(n) = n^{\log_2 5} \approx n^{2.32}$.
- The big-O notation $f(n) = O(g(n))$ indicates that $g(n)$ grows faster than $f(n)$.
- By comparing the powers, $n^{1/2}$ is slower than $n^{2.32}$, meaning $g(n)$ dominates $f(n)$.

Thus, the relationship can be expressed as:

$$n^{1/2} = O(5^{\log_2 n})$$

m. Given the functions $f(n) = n^{2^n}$ and $g(n) = 3^n$, the complexity comparison reveals:

$$f(n) = O(g(n))$$

This means that the growth rate of $g(n)$ dominates $f(n)$.

- When comparing the powers, see that $2^n < 3^n$, indicating that $g(n)$ grows faster.
- The big-O notation $f(n) = O(g(n))$ confirms that the computational speed of $g(n)$ is greater than $f(n)$.
- Simplification rules further demonstrate that 2^n is dominated by 3^n, hence $g(n)$ is superior to $f(n)$.

Thus, the final relationship can be expressed as:

$$n^{2^n} = O(3^n)$$

n. Given the functions $f(n) = 2^n$ and $g(n) = 2^{n+1}$, the complexity comparison shows that:

$$f(n) = \Theta(g(n))$$

This means that both functions grow at the same rate, and their complexities are equivalent.

- $f = O(g)$ indicates that $f(n)$ grows no faster than $g(n)$.
- $f = \Omega(g)$ suggests that $g(n)$ grows no faster than $f(n)$.

- Since both functions belong to the same growth class $O(n)$, their growth rates are essentially equivalent.

Thus, the final relationship between the two functions can be expressed as:

$$2^n = \Theta(2^{n+1})$$

o. Given the functions $f(n) = n!$ and $g(n) = 2^n$, the comparison yields:

$$f(n) = \Omega(g(n))$$

This indicates that $f(n)$ grows faster than $g(n)$.

- The value of $n!$ is known to follow Stirling's approximation:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

- Since $n!$ grows faster than 2^n, we can conclude that $f(n)$ dominates $g(n)$.
- $f = \Omega(g)$ confirms that $g(n)$ does not dominate $f(n)$.

Thus, the relationship between the two functions can be written as:

$$n! = \Omega\left(2^n\right)$$

p. Given the functions $f(n) = (\log n)^{\log n}$ and $g(n) = 2^{(\log_2 n)^2}$, the comparison shows:

$$f(n) = O(g(n))$$

This indicates that $f(n)$ grows slower than $g(n)$.

- $f(n) = n^{\log \log n}$
- $g(n) = n^{\log_2 n}$

Since $f(n)$ grows slower than $g(n)$:

$$(\log n)^{\log n} = O\left(2^{(\log_2 n)^2}\right)$$

q. Given the functions $f(n) = \sum_{i=1}^{n} i^k$ and $g(n) = n^{k+1}$, the comparison shows:

$$f(n) = \Theta(g(n))$$

This means that both functions grow at the same rate, and their complexities are equivalent.

- The summation $f(n) = \sum_{i=1}^{n} i^k$ can be approximated by n^{k+1}, which matches the form of $g(n)$.

Thus, the final relationship between the two functions is:

$$\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$$

2 0.2

Show that, if c is a positive real number, then the geometric series $g(n) = 1 + c + c^2 + \cdots + c^n$ has the following complexities based on different values of c:

0.2.a: Case $c < 1$

When $c < 1$, the series is strictly decreasing, and the sum converges to a constant as n increases. For example, if $c = 0.5$, the series becomes:

$$g(n) = 1 + 0.5 + 0.25 + 0.125 + \cdots$$

As the terms decrease rapidly, the sum approaches a finite value. Therefore, the complexity is:

$$g(n) = \Theta(1)$$

0.2.b: Case $c = 1$

When $c = 1$, the series becomes a simple sum of ones:

$$g(n) = 1 + 1 + 1 + \cdots + 1 = n + 1$$

This is a linear sum of $n + 1$ terms. Thus, the complexity is:

$$g(n) = \Theta(n)$$

0.2.c: Case $c > 1$

When $c > 1$, the terms of the series grow exponentially, and the sum is dominated by the last term c^n. For example, if $c = 2$, the series becomes:

$$g(n) = 1 + 2 + 4 + 8 + \cdots + 2^n$$

The last term 2^n dominates the sum, and the complexity is:

$$g(n) = \Theta(c^n)$$

For this complexity would be $\Theta(2^n)$.

Based on the value of c, the complexity of the geometric series $g(n) = 1 + c + c^2 + \cdots + c^n$ can be categorized as follows:

- If $c < 1$, the complexity is $\Theta(1)$.
- If $c = 1$, the complexity is $\Theta(n)$.
- If $c > 1$, the complexity is $\Theta(c^n)$.

3 0.4

0.4.a

Show that multiplying two 2x2 matrices requires 4 additions and 8 multiplications.

Given two matrices A and B:

$$A = abcd, \quad B = efgh$$

The product matrix $A \cdot B$ is:

$$A \cdot B = ae + bgaf + bhce + dgcf + dh$$

Thus, multiplying two 2x2 matrices involves 4 additions and 8 multiplications, as each element in the resulting matrix requires 2 multiplications and 1 addition.

0.4.b

To compute X^n efficiently, use a divide-and-conquer approach called exponentiation by squaring. The key idea is:

- If n is even, then $X^n = (X^{n/2})^2$.
- If n is odd, then $X^n = X \cdot X^{n-1}$.

This approach reduces the number of matrix multiplications to $O(\log n)$.

For example, to compute X^8 compute:

$$X^8 = (X^4)^2 = ((X^2)^2)^2$$

Thus, matrix exponentiation can be done in logarithmic time.