Big O: How fast can this function possibly grow as n becomes large?

Big Omega: What is the least fast that this function will grow as n becomes large?

Example 1 f(n) = (n^2)*(logn) + n + 3
Small Explanation

When evaluating the Big O of a function like **f(n) = (n^2)*(logn) + n + 3** we focus on the **term that grows the fastest** as **n** becomes large, since it will dominate the overall growth rate of the function.

So, in Big O notation, we say the function is O((n2)*(logn)).This means, "In the worst-case scenario, the function's complexity grows as fast as (n2)*(logn), and the other parts (linear growth and constant) aren't really significant when n is very large."

Q1) Can a function have the same Big O and Big Omega?
Yes, when a function's growth rate is tightly bounded, meaning we're confident about its growth rate in both best and worst cases, its Big O and Big Omega can be the same. This is called Theta (Θ) notation.

Example 2 f(n) = n! + 100nlogn
Small Explanation

When comparing these two parts, n! grows so rapidly that 100nlog(n)becomes almost insignificant for large values of n!

Therefore, in Big O notation, we say f(n) is O(n!). This means, "At worst, the function's complexity grows as fast as n! and the linear-logarithmic part doesn't significantly affect this growth for large n."

Q1) Why do we only care about n! and not 100nlog(n)

Because n! grows so fast and adds so much 'weight', the other part (100nlog(n)) doesn't add much in comparison when n is big.

Example 3 f(n) = 3n3 + 5n2 + 100 + 22n
Small Explanation

In this function, 2^{2n} grows so much faster than the other parts that it's the only one we really care about for Big O. So, we say f(n) is O(2^{2n}). This means, for very large values of n, the way f(n) grows is mostly determined by 2^{2n}

Q) Why do we ignore the $3n^3$, $5n^2$, and 100 in Big O notation.
Because 2^{2n} grows so much faster than these other parts, they become almost irrelevant for very large n. It's like comparing a rocket's speed to a car's - the rocket's speed is just on a totally different level.

Example 4 f(n) = 10n2 +sqrt(n) + log n
Small Explanation

In this function, $10n^2$ is the part that grows the fastest. The square root and logarithmic parts also increase as n gets bigger, but they don't add as much "weight" to the function compared to $10n^2$.

Therefore, when determining Big Omega, we focus on the term that most significantly impacts the growth rate from below. In this function, the $10n^2$ term is the most significant for large values of n, making Big O (n2) the appropriate notation.

Big Omega

Big Omega: What is the least fast that this function will grow as n becomes large?

Example 1 f(n) = n! + n2 +n3
Small Explanation

For Big Omega, we look at the term that significantly contributes to the function's growth from below. In this case, the factorial term n! is the dominant factor. Even though it represents the fastest growth, no other term in the function provides a significant contribution to the growth rate compared to n!.

Therefore, the correct Big Omega notation for this function is Omega(n!). **This means the function grows at least as fast as factorial growth for large values of n**. The presence of the polynomial terms n^2 and n^3 does not significantly affect the minimum growth rate, which is dominated by the factorial term.

Example 2 f(n) = log n + n + 10
Small Explanation

For Big Omega, we focus on the part of the function that grows the most, but at the slowest rate compared to other parts. In this case, that's n. Even though log n grows slower, its impact is negligible compared to n for large values of n.

So, the Big Omega of the function is Omega(n). **This means that as n gets large, the function will grow at least as quickly as a linear function.**

Q) Why isn't Big Omega log n, since it grows slower?

Because Big Omega is about finding the slowest significant growth. While log n grows slower than n, it's not significant when compared to the linear growth of n for large values.

Example 3 f(n) = 2^n + n^3
Small Explanation

For Big Omega, we're interested in the **slowest part** of the function that still **significantly** contributes to its growth. In this case, the cubic term n^3 is overshadowed by the exponential term 2^n. Even though 2^n grows faster, it also sets the minimum growth rate because it becomes the dominant term very quickly as n increases.

So, the Big Omega of this function is Omega(2^n). **This means that as n gets large, the function will grow at least as quickly as an exponential function.**

Example 4 f(n) = log(n!)
Small Explanation

So, the Big Omega of this function is not straightforward like a simple polynomial or exponential function. It's more complex due to the factorial. However, we can approximate the growth rate using Stirling's approximation, which suggests that log(n!) grows similarly to n log n. Thus, a reasonable Big Omega for this function could be expressed as Omega(n log n).

Q) Why do we use Stirling's approximation for this function?

Stirling's approximation is used because it simplifies the understanding of how factorial growth behaves. It allows us to approximate the behavior of log(n!) in a more manageable form, making it easier to understand and work with.

Example 5 f(n) = sqrt(n) + logn
Small Explanation

For Big Omega, we consider the term that grows the slowest but still contributes notably to the function's growth. In this case, even though logn grows slower, sqrt(n) is more significant in terms of its contribution to the overall growth of the function.

So, the Big Omega of this function is Omega(sqrt(n)). This means that, at a minimum, the function grows as fast as the square root of n