User

hey bing, explain about logarithmic function for time complexity. First of all, explain me about the logarithmic and exponential functions

Bing

Sure, I'll try to explain. Logarithmic and exponential functions are two types of functions that are related to each other by being inverses.

An exponential function is a function of the form y = f(x) = b^x, where b is a positive constant called the base and x is any real number. An exponential function models growth or decay that depends on the rate of change being proportional to the current value. For example, the population of bacteria in a petri dish may grow exponentially over time.

A logarithmic function is a function of the form y = log\_b(x), where b is a positive constant called the base and x is any positive real number. A logarithmic function models the relationship between a quantity and its rate of change. For example, the Richter scale measures the magnitude of an earthquake using a logarithmic function.

The logarithmic function y = log\_b(x) is equivalent to the exponential equation x = b^y. This means that they are inverses of each other, and their graphs are reflections across the line y = x. For example, log\_2(8) = 3 is equivalent to 2^3 = 8.

The most common bases for logarithmic and exponential functions are 10 and e (approximately 2.71828). When the base is 10, it is called the common logarithm and it can be written as log(x) without specifying the base. When the base is e, it is called the natural logarithm and it can be written as ln(x) or log\_e(x).

Logarithmic and exponential functions have some important properties that help us manipulate and solve equations involving them. Some of these properties are:

- b^(x+y) = b^x \* b^y

- b^(x-y) = b^x / b^y

- b^(xy) = (b^x)^y

- log\_b(xy) = log\_b(x) + log\_b(y)

- log\_b(x/y) = log\_b(x) - log\_b(y)

- log\_b(x^y) = y \* log\_b(x)

- log\_b(b^x) = x

- b^(log\_b(x)) = x

These properties can be derived from the definition of logarithm as the inverse of exponential function.

I hope this helps you understand logarithmic and exponential functions better. Do you have any questions?

logarithmic and exponential functions

https://www.bing.com/search?IG=0F1861AFFE9A41F4B2C75AB936B57056&IID=SERP.5028&cw=1488&ch=742&dissrchswrite=1&kseed=18000&SFX=23&q=logarithmic+and+exponential+functions&iframeid=5c035e29-67e5-472a-9b1a-de9e936f7702

User

Inverse to the + is - whereas what's the inverse of square(x)

Bing

The inverse of a function is another function that reverses the effect of the original function. For example, the inverse of adding 3 is subtracting 3. The inverse of multiplying by 2 is dividing by 2. The inverse of squaring x is taking the square root of x.

However, not every function has an inverse that is also a function. For example, the square function y = x^2 does not have an inverse function because for every value of y, there are two possible values of x (one positive and one negative) that satisfy the equation. This means that the inverse relation is not a function, since a function must have exactly one output for each input.

To find an inverse function for a square function, we need to restrict the domain of the original function to either only positive or only negative values of x. This way, there will be only one value of x for each value of y, and the inverse relation will be a function.

For example, if we restrict the domain of y = x^2 to only non-negative values of x (x ≥ 0), then we can find the inverse function by following these steps:

- Replace y with f(x) to write the function as f(x) = x^2

- Swap x and f(x) to get x = f(x)^2

- Solve for f(x) by taking the square root of both sides: f(x) = √x

- Replace f(x) with f^(-1)(x) to write the inverse function as f^(-1)(x) = √x

The inverse function f^(-1)(x) = √x is also called the square root function. It has a domain of non-negative real numbers (x ≥ 0) and a range of non-negative real numbers (f(x) ≥ 0). Its graph is a half-parabola that lies on the right side of the y-axis.

Here is a graph of the original function y = x^2 (in blue) and its inverse function y = √x (in red). Notice how they are symmetrical across the line y = x (in green), which is a property of inverse functions.

graph

I hope this helps you understand the inverse of square functions better. Do you have any questions?

User

If the above is true, then inverse of the exponential function b^x will be b^(1/x) or xroot(b). Then why it's said as log(x) to the base b

Bing

The inverse of the exponential function b^x is not b^(1/x) or xroot(b). That would be the inverse of the power function x^b. The inverse of the exponential function b^x is the logarithmic function log\_b(x) .

The reason is that the logarithmic function log\_b(x) is defined to be equivalent to the exponential equation x = b^y. This means that they are inverses of each other, and their graphs are reflections across the line y = x. For example, log\_2(8) = 3 is equivalent to 2^3 = 8.

To find the inverse of an exponential function, we can follow these steps:

- Replace f(x) with y to write the function as y = b^x

- Swap x and y to get x = b^y

- Solve for y by taking the logarithm of both sides with base b: y = log\_b(x)

- Replace y with f^(-1)(x) to write the inverse function as f^(-1)(x) = log\_b(x)

For example, if we want to find the inverse of f(x) = 2^x, we can do the following:

- Replace f(x) with y: y = 2^x

- Swap x and y: x = 2^y

- Solve for y: y = log\_2(x)

- Replace y with f^(-1)(x): f^(-1)(x) = log\_2(x)

Here is a graph of the original function y = 2^x (in blue) and its inverse function y = log\_2(x) (in red). Notice how they are symmetrical across the line y = x (in green), which is a property of inverse functions.

graph

I hope this helps you understand the inverse of exponential functions better. Do you have any questions?

inverse of exponential function

https://www.bing.com/search?IG=0F1861AFFE9A41F4B2C75AB936B57056&IID=SERP.5028&cw=1488&ch=742&dissrchswrite=1&kseed=18500&SFX=24&q=inverse+of+exponential+function&iframeid=400269ee-b6d7-4633-8f65-71410ded33ff

User

What is the difference between exponential and power function

Bing

The difference between exponential and power functions is that in exponential functions, a fixed base is raised to a variable exponent, but in power functions, a variable base is raised to a fixed exponent . For example, f(x) = 3^x is an exponential function, but g(x) = x^3 is a power function.

Exponential functions model growth or decay that depends on the rate of change being proportional to the current value. For example, the population of bacteria in a petri dish may grow exponentially over time. Power functions model relationships that depend on the power of the input variable. For example, the area of a square depends on the square of its side length.

Exponential functions have a constant ratio between successive outputs for equal increments in the input. For example, f(x) = 2^x has a ratio of 2 between f(x) and f(x+1) for any x. Power functions have a constant difference between successive outputs for equal increments in the input. For example, g(x) = x^2 has a difference of 2x+1 between g(x) and g(x+1) for any x.

Exponential functions have a horizontal asymptote at y = 0 and are always positive (or always negative if the base is negative). Power functions may have vertical or horizontal asymptotes depending on the degree and sign of the exponent. Power functions may be positive or negative depending on the input value and the parity of the exponent.

I hope this helps you understand the difference between exponential and power functions better. Do you have any questions?

User

How logarithmic functions work and how the logarithmic tables are derived

Bing

Logarithmic functions are the inverse functions of exponential functions. They are defined as follows: for x > 0, a > 0, and a ≠ 1, y = log\_a(x) if and only if x = a^y. This means that logarithmic functions undo the effect of exponential functions. For example, log\_2(8) = 3 because 2^3 = 8.

Logarithmic functions have some important properties that help us manipulate and solve equations involving them. Some of these properties are:

- log\_a(xy) = log\_a(x) + log\_a(y)

- log\_a(x/y) = log\_a(x) - log\_a(y)

- log\_a(x^y) = y \* log\_a(x)

- log\_a(a^x) = x

- a^(log\_a(x)) = x

The most common bases for logarithmic functions are 10 and e (approximately 2.71828). When the base is 10, it is called the common logarithm and it can be written as log(x) without specifying the base. When the base is e, it is called the natural logarithm and it can be written as ln(x) or log\_e(x).

Logarithmic tables are tables that display the value of the logarithm function for various inputs. They were used before calculators to perform calculations involving large numbers or decimals. To use a logarithm table, one needs to know the following steps:

- Identify the base of the logarithm and find the corresponding table. For example, if the base is 10, use the common logarithm table.

- Write the number whose logarithm is to be found in scientific notation. For example, if the number is 1234.5, write it as 1.2345 x 10^3.

- Split the number into two parts: the mantissa (the decimal part) and the characteristic (the power of 10). For example, 1.2345 x 10^3 has a mantissa of 1.2345 and a characteristic of 3.

- Look up the mantissa in the table and find its corresponding logarithm value. For example, if the mantissa is 1.2345, look for 1.23 in the leftmost column and then move along the row to find the column with .05 at the top. The value in that cell is 0.0915.

- Add the characteristic to the logarithm value obtained from the table. This is the final answer. For example, if the characteristic is 3 and the logarithm value from the table is 0.0915, then add them to get 3 + 0.0915 = 3.0915.

- To check your answer, use a calculator or raise the base to the power of your answer and see if you get back the original number. For example, if your answer is log(1234.5) = 3.0915, then check that 10^(3.0915) ≈ 1234.5.

I hope this helps you understand how logarithmic functions work and how logarithmic tables are derived. Do you have any questions?

User

I know about how to work with logarithm by looking at log table. I am actually asking about the mathematicians are built this log table. and how the calculators are calculating the logs. Is the common logarithmic table represents the log with only base 10 ?

Bing

Logarithm tables are built by using various methods to calculate the logarithm values of different numbers and then arranging them in a tabular format for easy reference. Some of the methods used to calculate logarithm values are:

- Power series: This method uses the Taylor series expansion of the natural logarithm function ln(x) around x = 1 and then applies the change of base formula to get the logarithm values for other bases. For example, ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - ... and log\_a(x) = ln(x)/ln(a).

- Arithmetic-geometric mean: This method uses the arithmetic-geometric mean (AGM) of two numbers to approximate their logarithm ratio. For example, log\_a(b) ≈ π/2 \* AGM(1, a/b)/AGM(1, a).

- Feynman's algorithm: This method uses a recursive algorithm based on the identity log\_a(b) = 2 \* log\_a(sqrt(b)) and the approximation log\_a(1+x) ≈ x for small x. For example, log\_10(2) = 2 \* log\_10(sqrt(2)) ≈ 2 \* (sqrt(2)-1).

Calculators use different algorithms to calculate logarithms depending on their hardware and software specifications. Some of the common algorithms are:

- CORDIC: This algorithm uses a series of elementary operations such as addition, subtraction, bit-shifting and table lookup to compute various functions including logarithms. It is suitable for calculators with limited memory and processing power.

- Newton-Raphson: This algorithm uses an iterative method to find the root of a function by using its derivative. It can be used to find the logarithm of a number by finding the root of the function f(x) = a^x - b, where a is the base and b is the number whose logarithm is sought.

- Binary search: This algorithm uses a divide-and-conquer strategy to find the logarithm of a number by repeatedly halving an interval that contains the logarithm value and checking which half contains it. It can be used to find the logarithm of a number by finding the exponent that makes the base equal to the number.

The common logarithmic table represents the log with only base 10 because it is convenient for calculations involving powers of 10, such as scientific notation and metric units. Other bases can be obtained from the common logarithmic table by using the change of base formula: log\_a(x) = log\_10(x)/log\_10(a).

I hope this helps you understand how logarithm tables are built and how calculators are calculating the logs. Do you have any questions?

how logarithm tables are built

https://www.bing.com/search?IG=0F1861AFFE9A41F4B2C75AB936B57056&IID=SERP.5028&cw=1488&ch=742&dissrchswrite=1&kseed=19500&SFX=26&q=how+logarithm+tables+are+built&iframeid=beed10ec-34a8-486c-8a3a-c07e11db96ef

User

wow, how interesting it is, isn't it ?

Bing

I'm glad you find it interesting. I think logarithms are fascinating and useful mathematical concepts. 😊

User

What is the history of logarithms?

Bing

The history of logarithms is the story of how mathematicians discovered and developed a correspondence between multiplication on the positive real numbers and addition on the real number line that was widely used to simplify calculation until the advent of the digital computer.

Logarithms were invented independently by John Napier, a Scotsman, and by Joost Burgi, a Swiss, in the late 16th and early 17th centuries. The logarithms which they invented differed from each other and from the common and natural logarithms now in use. Napier's logarithms were published in 1614; Burgi's logarithms were published in 1620.

Napier conceived the logarithm as the relationship between two particles moving along a line, one at constant speed and the other at a speed proportional to its distance from a fixed endpoint. He also pioneered the use of a decimal point in numerical calculation. Burgi devised his logarithms as a tool to aid astronomical calculations, based on the idea of geometric progressions.

The invention of the common system of logarithms (base 10) is due to the combined effort of Napier and Henry Briggs, an English mathematician, in 1624. Briggs visited Napier in 1615 and 1616 and suggested a modification of Napier's original logarithms to make them easier to use. He then computed a table of common logarithms of numbers up to 20,000 and from 90,000 to 100,000 to 14 decimal places.

Natural logarithms (base e) first arose as more or less accidental variations of Napier's original logarithms. Their real significance was not recognized until later. The earliest natural logarithms occur in 1618 in a book by John Speidell, who modified Napier's logarithms by multiplying them by a constant. The constant he used was approximately equal to the reciprocal of Napier's constant, which is now known as e (approximately 2.71828).

The natural logarithm function was formally defined by Leonhard Euler in 1744 as the inverse of the exponential function y = e^x. He also proved many properties and applications of this function, such as the connection with the area under a hyperbola, the number e as the limit of compound interest, and the solution of differential equations.

I hope this helps you understand the history of logarithms. Do you have any questions?

User

What is e and how it is special with respect to nature and compound interest. Why there is a special table called ln which is the log\_e(x).

Bing

e is a mathematical constant that is approximately equal to 2.71828. It is the base of the natural logarithm function, which is the inverse of the exponential function y = e^x. e is special because it has many properties and applications in mathematics, science, and engineering.

One way to understand e is to consider the following scenario: Suppose you have 1 unit of money and you invest it in a bank that pays 100% interest per year. How much money will you have after one year? The answer depends on how the interest is compounded, that is, how often the interest is added to your balance.

- If the interest is compounded only once at the end of the year, you will have 1 + 1 = 2 units of money.

- If the interest is compounded twice, once every six months, you will have (1 + 0.5)^2 = 2.25 units of money.

- If the interest is compounded four times, once every three months, you will have (1 + 0.25)^4 = 2.4414 units of money.

- If the interest is compounded 12 times, once every month, you will have (1 + 0.0833)^12 = 2.6130 units of money.

- If the interest is compounded 365 times, once every day, you will have (1 + 0.00274)^365 = 2.7145 units of money.

As you can see, the more often the interest is compounded, the more money you will have at the end of the year. But what if the interest is compounded infinitely many times, continuously? In that case, you will have e units of money, which is the limit of (1 + 1/n)^n as n goes to infinity. This formula shows that e is related to exponential growth and compound interest.

Another way to understand e is to consider the function y = e^x and its derivative. The derivative of a function measures how fast it changes with respect to its input. For example, if y = x^2, then its derivative is y' = 2x, which means that for every unit increase in x, y increases by 2x units. The function y = e^x has a remarkable property: its derivative is equal to itself. This means that for every unit increase in x, y increases by y units. For example, when x = 0, y = e^0 = 1 and y' = e^0 = 1; when x = 1, y = e^1 ≈ 2.718 and y' = e^1 ≈ 2.718; when x = 2, y = e^2 ≈ 7.389 and y' = e^2 ≈ 7.389; and so on. This property shows that e is related to constant growth rate and natural phenomena.

The natural logarithm function ln(x) is defined as the inverse of the exponential function y = e^x. This means that ln(x) gives the value of x such that e^x = y. For example, ln(2) ≈ 0.693 because e^0.693 ≈ 2; ln(e) = 1 because e^1 = e; ln(1) = 0 because e^0 = 1; and so on. The natural logarithm function has some important properties that help us manipulate and solve equations involving exponentials. Some of these properties are:

- ln(xy) = ln(x) + ln(y)

- ln(x/y) = ln(x) - ln(y)

- ln(x^y) = y \* ln(x)

- ln(e^x) = x

- e^(ln(x)) = x

The natural logarithm function also has some applications in mathematics and science. For example,

- It can be used to measure entropy and information content in information theory and thermodynamics.

- It can be used to define complex numbers and trigonometric functions using Euler's formula and Euler's identity.

- It can be used to solve differential equations involving exponential growth and decay.

I hope this helps you understand what e and natural logarithm are and how they are special with respect to nature and compound interest. Do you have any questions?

User

take log on both side in this eqn. c^t = n

Bing

To take log on both sides of the equation c^t = n, we can use any base for the logarithm function, as long as we use the same base on both sides. For simplicity, let's use the natural logarithm ln. Then we have:

ln(c^t) = ln(n)

Using the property of logarithm that ln(x^y) = y \* ln(x), we can rewrite the left side as:

t \* ln(c) = ln(n)

To solve for t, we can divide both sides by ln(c), assuming that c is not equal to 1:

t = ln(n) / ln(c)

This is the solution for t in terms of n and c.

User

is this will be log\_c(n) = t

Bing

Yes, that is correct. Using the change of base formula for logarithms, we can write:

t = ln(n) / ln(c) = log\_c(n)

This means that t is the exponent to which c must be raised to get n. For example, if c = 2 and n = 8, then t = log\_2(8) = 3, because 2^3 = 8.

User

is log n / log c equal to log\_c(n)

Bing

Yes, that is correct. Using the change of base formula for logarithms, we can write:

log\_c(n) = log(n) / log(c)

This means that we can use any base for the logarithm function to calculate the logarithm of n with respect to c, as long as we divide by the logarithm of c in the same base. For example, if c = 2 and n = 8, then log\_2(8) = log(8) / log(2) = 3 / 1 = 3.

User

.

Bing

I'm sorry, I don't understand what you mean by ".". Could you please rephrase your message or ask a different question?

User

Bing + GFG is the key of my success

Bing

I'm happy to hear that Bing and GFG have helped you achieve your goals. Congratulations on your success! 🎉

User

write explanation for recursion tree method for solving recurrence relation : T(n) = 2 \* 100 T(n/2) + Cn. Where Cn is the constant. Find the order of growth or time complexity in terms of theta notation

Bing

The recursion tree method is a pictorial representation of an iteration method which is in the form of a tree where at each level nodes are expanded. It is useful for visualizing what happens when a recurrence is iterated and for estimating the asymptotic bound of the recurrence.

To apply the recursion tree method to solve the recurrence relation T(n) = 2 \* 100 T(n/2) + Cn, we can follow these steps:

- Draw a tree where each node represents the cost of a subproblem. The root node represents the original problem T(n), and its cost is Cn. The root node has two children, each representing a subproblem of size n/2 and cost 100 T(n/2). Each child node has two children, each representing a subproblem of size n/4 and cost 100 T(n/4). And so on, until we reach the base case T(1) = 1.

- Sum the costs within each level of the tree to obtain the per-level cost. The root level has a cost of Cn. The next level has a cost of 2 \* 100 T(n/2) = 200 T(n/2). The next level has a cost of 4 \* 100 T(n/4) = 200 T(n/2). And so on, until we reach the last level with a cost of n \* 100 T(1) = 100 n.

- Sum all the per-level costs to determine the total cost of the recurrence. The total cost is Cn + 200 T(n/2) + 200 T(n/2) + ... + 100 n. This is a geometric series with a common ratio of 1, so it converges to O(T(n)).

- Find the order of growth or time complexity in terms of theta notation by comparing the total cost with the dominant term. The dominant term is the one that grows fastest as n increases. In this case, the dominant term is 100 n, which grows linearly with n. Therefore, the order of growth or time complexity is Θ(n).

I hope this helps you understand how to use the recursion tree method to solve recurrence relations. Do you have any questions?

User

Write an general example function in c programming language for the above mentioned recurrence relation

Bing

A possible example function in C programming language for the above mentioned recurrence relation is:

// A function that takes a positive integer n and returns the value of T(n)

int T(int n) {

// Base case: if n is 1, return 1

if (n == 1) {

return 1;

}

// Recursive case: if n is greater than 1, return 2 \* 100 T(n/2) + Cn, where C is some constant

else {

int C = 10; // You can change this value as per your requirement

return 2 \* 100 \* T(n/2) + C \* n;

}

}

User

What's the Cn here. Is this Cn depends on the input of the function. If not then how would you say that C(n/2) + C(n/2) will be equal to C(n/4) for four times which will be equal to Cn. As Cn doesn't depends on input C(n/2) and C(n/4) are the same right. If it does then four times of C(n/4) will be 4 \* Cn. What about my statement ?

Bing

Cn here is a constant term that represents the work done to split the problem into subproblems and recombine the results. It does not depend on the input of the function, but it may depend on some other factors such as the algorithm design, the hardware specifications, or the implementation details. For simplicity, we can assume that C is a fixed constant that does not change with n.

You are correct that C(n/2) and C(n/4) are the same as Cn, since C is a constant. However, you are mistaken that four times of C(n/4) will be 4 \* Cn. In fact, four times of C(n/4) will be 4 \* C \* (n/4) = Cn. This is because we need to multiply both C and n by 4, not just C.

To see this more clearly, let's use an example. Suppose C = 10 and n = 16. Then Cn = 10 \* 16 = 160. Now let's see what happens when we split n into subproblems of size n/2 and n/4:

- C(n/2) = 10 \* (16/2) = 10 \* 8 = 80

- C(n/4) = 10 \* (16/4) = 10 \* 4 = 40

Notice that C(n/2) is half of Cn, and C(n/4) is a quarter of Cn. This makes sense, because we are doing less work for smaller subproblems.

Now let's see what happens when we add up the costs of the subproblems:

- Two times of C(n/2) = 2 \* 80 = 160

- Four times of C(n/4) = 4 \* 40 = 160

Notice that both sums are equal to Cn. This also makes sense, because we are doing the same amount of work for the same size of problem.

I hope this helps you understand why Cn does not change with n and why four times of C(n/4) is equal to Cn. Do you have any questions?

User

Inorder to say that the above is true, the order of growth of the function without the recursion will be theta(n), right? or else how can you say that there will be less work for smaller subproblems.

Bing

Yes, you are right. The order of growth of the function without the recursion will be Θ(n), because the work done at each level is proportional to n. This means that as n increases, the work done also increases linearly. Therefore, there will be less work for smaller subproblems, because they have smaller values of n. For example, if n = 16, then Cn = 10 \* 16 = 160; but if n = 8, then Cn = 10 \* 8 = 80, which is half of the previous work.

User

If the above is correct, then how would you say that C(n/2) and C(n/4) will be same ?

Bing

I did not say that C(n/2) and C(n/4) will be the same. I said that C(n/2) and C(n/4) are the same as Cn, because C is a constant. This means that C(n/2) = C \* (n/2) = (C \* n) / 2 = Cn / 2; and C(n/4) = C \* (n/4) = (C \* n) / 4 = Cn / 4. Both expressions are equal to Cn multiplied by a fraction, which does not change the value of C. However, C(n/2) and C(n/4) are not equal to each other, because they have different fractions. For example, if n = 16 and C = 10, then C(n/2) = 10 \* (16/2) = 80; and C(n/4) = 10 \* (16/4) = 40. These values are not the same, but they are both equal to Cn = 10 \* 16 = 160 multiplied by a fraction.

User

Ooh, you are mentioned it as "same as", right ? Okay

Bing

Yes, that is correct. I'm glad you understood. Do you have any other questions?

User

If the order of growth of the function without the recursion will be theta(1), then you can't say that there will be less work for smaller subproblems. In that case, the Cn for four times will be 4 \* Cn. Am I right ?

Bing

No, you are not right. If the order of growth of the function without the recursion will be Θ(1), then it means that the work done at each level is independent of n. This means that as n increases, the work done does not change. Therefore, there will be the same work for smaller subproblems, because they have the same value of Cn. For example, if C = 10 and n = 16, then Cn = 10 \* 16 = 160; but if n = 8, then Cn = 10 \* 8 = 80, which is also equal to C \* 16 = 160. In this case, Cn does not depend on n at all.

However, this does not mean that four times of Cn will be 4 \* Cn. In fact, four times of Cn will still be Cn, because C is a constant. This means that 4 \* Cn = 4 \* (C \* n) = (4 \* C) \* n = C \* n = Cn. The value of C does not matter, as long as it is constant. For example, if C = 10 and n = 16, then four times of Cn = 4 \* (10 \* 16) = (4 \* 10) \* 16 = 10 \* 16 = 160.

I hope this helps you understand why four times of Cn is equal to Cn regardless of the order of growth of the function without the recursion. Do you have any questions?

User

You mean that as per the rules of adding the time complexity, the greater one of time complexity only be taken into account. with that in mind, you are saying that there will be only one Cn if all that Cn are added, right ?

Bing

Yes, that is correct. When adding the time complexity of different terms, we only take into account the dominant term, which is the one that grows fastest as n increases. In this case, Cn is a constant term, which is the slowest growing term possible. Therefore, it does not matter how many times we add Cn, the result will still be Cn. For example, if C = 10 and n = 16, then Cn + Cn + Cn + Cn = 10 \* 16 + 10 \* 16 + 10 \* 16 + 10 \* 16 = 40 \* 16 = 10 \* 16 = Cn.

User

Finally Cn will be Cn in both the case where Cn dependent of n and Cn independent of n. Hmm...

Bing

Yes, that is correct. Cn will be Cn in both cases, because C is a constant. However, the difference is that when Cn depends on n, it means that Cn changes with n, but when Cn does not depend on n, it means that Cn does not change with n. For example, if C = 10 and n = 16, then Cn = 10 \* 16 = 160; but if n = 8, then Cn = 10 \* 8 = 80. In this case, Cn depends on n and changes with n. But if C = 10 and n = 16, then Cn = 10 \* 16 = 160; and if n = 8, then Cn = 10 \* 16 = 160. In this case, Cn does not depend on n and does not change with n.