

COMP 250

Assignment Project Exam Help

INTRODUC TER SCIENCE

<https://eduassistpro.github.io/>

Week 6-8: Asympt

Add Wechat edu_assist_pro

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Properties of Asymptotic notations

- Big-Omega, $\Omega(\cdot)$

- Big-Theta, $\Theta(\cdot)$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

RULES OF BIG-OH

- Scaling
- Sum rule
- Product Rule
- Transitivity

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

SCALING

For all constant factors $a > 0$,

if $f(n)$ is $O(g(n))$, then $a \cdot f(n)$ is also $O(g(n))$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

(This rule is obvious if you understand the definition of big O)

SCALING

For all constant factors $a > 0$,
if $f(n)$ is $O(g(n))$, then $a \cdot f(n)$ is also $O(g(n))$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Proof: By definition, if $f(n) = O(g(n))$,
there exist constants n_0 and c such that, for all n

Add WeChat edu_assist_pro

sitive

$$f(n) \leq c \cdot g(n).$$

Thus, ...?

SCALING

For all constant factors $a > 0$,

if $f(n)$ is $O(g(n))$, then $a \cdot f(n)$ is also $O(g(n))$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Proof: By definition, if $f(n) = O(g(n))$

sitive

constants n_0 and c such that, for all n

Add WeChat edu_assist_pro

$$f(n) \leq c g(n).$$

Thus,

$$a \cdot f(n) \leq \underbrace{a c}_{\text{constant}} g(n)$$

This constant satisfies the definition that $a \cdot f(n)$ is $O(g(n))$

SUM RULE

If $f_1(n)$ is $O(g(n))$ and $f_2(n)$ is $O(g(n))$, then $f_1(n) + f_2(n)$ is $O(g(n))$.

Proof: ...

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

SUM RULE

If $f_1(n)$ is $O(g(n))$ and $f_2(n)$ is $O(g(n))$, then $f_1(n) + f_2(n)$ is $O(g(n))$.

Assignment Project Exam Help

Proof: Let n_1, c_1 and n_2, c_2

<https://eduassistpro.github.io/>

$f_1(n) \leq c_1 g(n)$ for all $n \geq n_1$ and $f_2(n) \leq c_2 g(n)$ for all $n \geq n_2$

Add WeChat edu_assist_pro

SUM RULE

If $f_1(n)$ is $O(g(n))$ and $f_2(n)$ is $O(g(n))$, then $f_1(n) + f_2(n)$ is $O(g(n))$.

Assignment Project Exam Help

Proof: Let n_1, c_1 and n_2, c_2

<https://eduassistpro.github.io/>

$f_1(n) \leq c_1 g(n)$ for all $n \geq n_1$ and $f_2(n) \leq c_2 g(n)$ for all $n \geq n_2$

Then,

$$f_1(n) + f_2(n) \leq \underbrace{(c_1 + c_2)}_{\text{These constants satisfy the big } O \text{ definition}} \underbrace{g(n)}_{\text{These constants satisfy the big } O \text{ definition}} \text{ for all } n \geq \max(n_1, n_2)$$

These constants satisfy the big O definition

SUM RULE (MORE GENERAL)

If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$,

Assignment Project Exam Help

Then $f_1(n) + f_2(n)$ is $O(\quad)$

<https://eduassistpro.github.io/>

Proof: Try it!

Add WeChat edu_assist_pro

PRODUCT RULE

If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$, then $f_1(n) * f_2(n)$ is $O(g_1(n) * g_2(n))$.

Assignment Project Exam Help

Proof: ...

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

PRODUCT RULE

If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$, then $f_1(n) * f_2(n)$ is $O(g_1(n) * g_2(n))$.

Assignment Project Exam Help

Proof: Let n_1, c_1 and n_2, c_2

<https://eduassistpro.github.io/>

$f_1(n) \leq c_1 g_1(n)$ for all $n \geq n_1$ and $f_2(n) \leq c_2 g_2(n)$ for all $n \geq n_2$

Add WeChat edu_assist_pro

PRODUCT RULE

If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$, then $f_1(n) * f_2(n)$ is $O(g_1(n) * g_2(n))$.

Assignment Project Exam Help

Proof: Let n_1, c_1 and n_2, c_2

<https://eduassistpro.github.io/>

$f_1(n) \leq c_1 g_1(n)$ for all $n \geq n_1$ and $f_2(n) \leq c_2 g_2(n)$ for all $n \geq n_2$

Add WeChat edu_assist_pro

Then,

$$f_1(n) * f_2(n) \leq \underbrace{c_1 c_2}_{\text{constants}} \underbrace{g_1(n) * g_2(n)}_{\text{product of functions}} \text{ for all } n \geq \max(n_1, n_2)$$

These constants satisfy the big O definition

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then... ?

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

Assignment Project Exam Help

Proof: Let n_1, c_1 and n_2 ,

<https://eduassistpro.github.io/>

$f(n) \leq c_1 g(n)$ for all $n \geq n_1$ and $g(n) \leq c_2 h(n)$ for all $n \geq n_2$

Add WeChat edu_assist_pro

TRANSITIVITY RULE

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

Assignment Project Exam Help

Proof: Let n_1, c_1 and $n_2,$

<https://eduassistpro.github.io/>

$f(n) \leq c_1 g(n)$ for all $n \geq n_1$ and $g(n) \leq c_2 h(n)$ for all $n \geq n_2$

Add WeChat edu_assist_pro

Then,

$$f(n) \leq \underbrace{c_1 c_2}_{\text{constant}} \underbrace{h(n)}_{\text{function}} \text{ for all } n \geq \max(n_1, n_2)$$

These constants satisfy the big O definition

COMMON FUNCTIONS

Claim: each of the following holds for n sufficiently large

$$\underbrace{1 < \log_2 n < n}_{n \geq 3} \quad \underbrace{2^2 < 2^3 < \dots < 2^n < n!}_{n \geq 4}$$

Assignment Project Exam Help
<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

$$n^3 < 2^n \quad \text{for } n \geq 10$$

COMMON FUNCTIONS

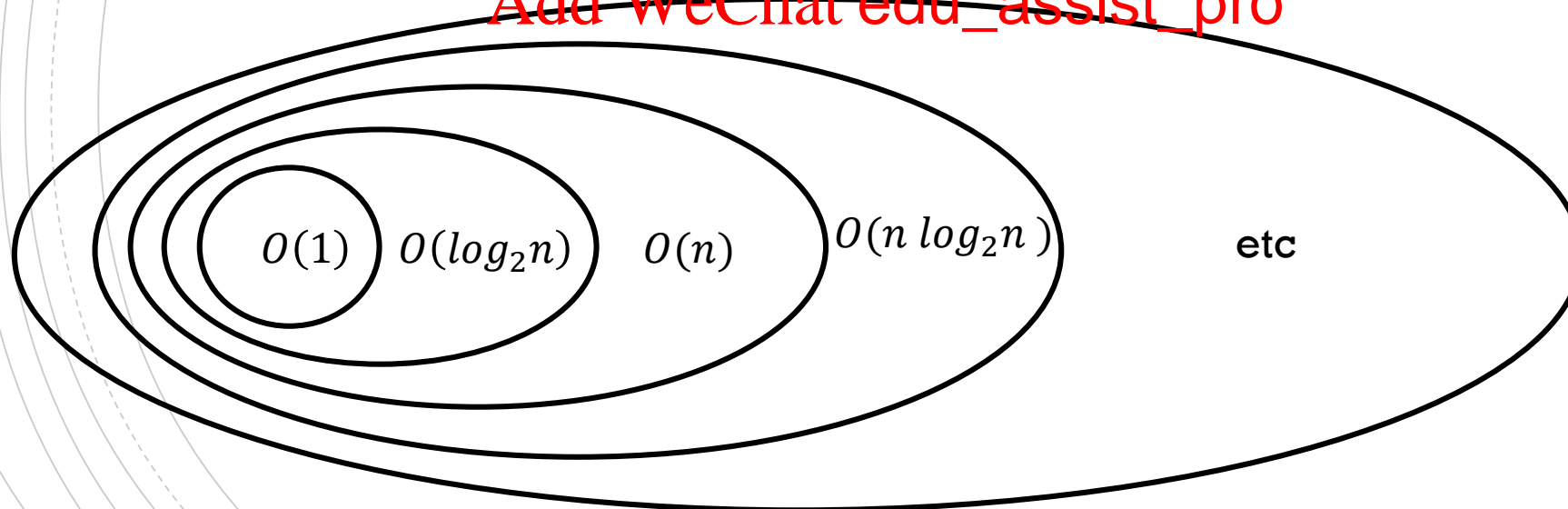
Each of the following holds for n sufficiently large:

$$1 < \log_2 n < n < n \log_2 n < n^2 < n^3 < \dots < 2^n < n!$$

Assignment Project Exam Help

Thus we have the following st <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



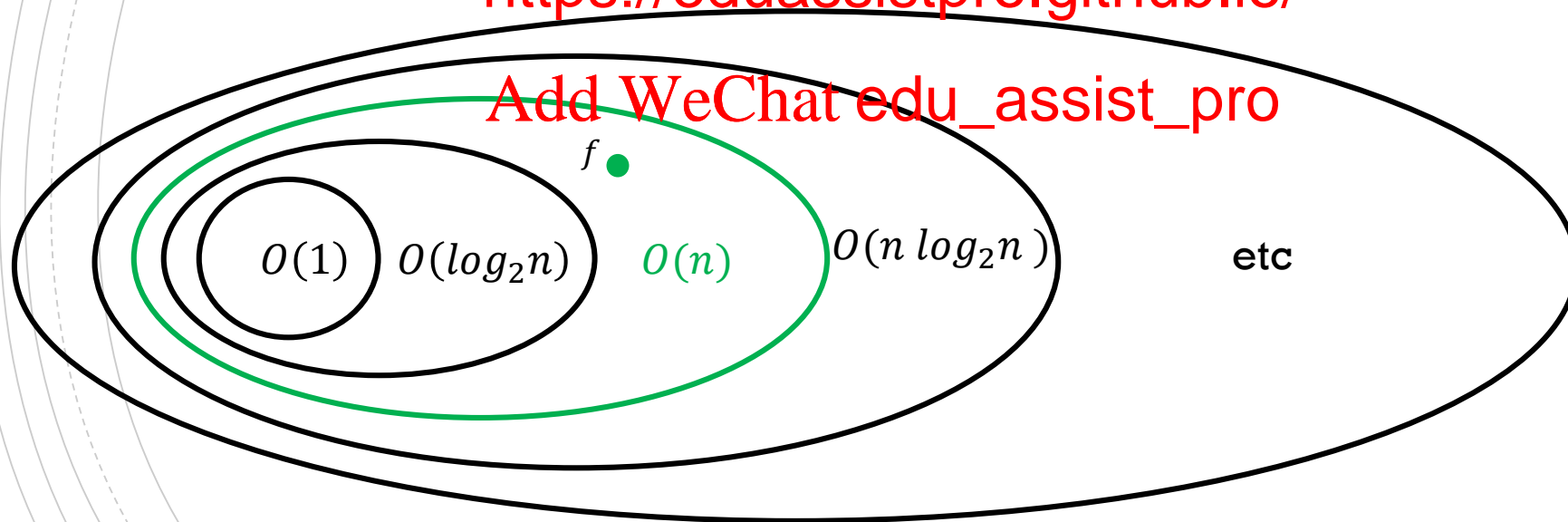
BACK TO TIGHT BOUNDS

If we consider the function $f(n) = 5n + 7$, then the **tight upper bound** for f is $O(n)$ and not $O(n \log_2 n)$ for instance.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



EXAMPLE

Using these claims/rules allow us to say, for example, that

Assignment Project Exam Help

$f(n) = 3 \frac{\text{https://eduassistpro.github.io/}}{2}$ is $O(n^2)$.

Add WeChat edu_assist_pro

GENERAL OBSERVATION

Never write $O(3n)$, $O(5 \log_2 n)$, etc.

Assignment Project Exam Help

Instead, write $O(n)$, $O(\log$ <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Why? The set $O(3n)$ is exactly the same set defined by $O(n)$, and so are the others.

It is still *technically* correct to write the above. We just don't do it to avoid dealing with constant factors.

Assignment Project Exam Help

B <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

ASYMPTOTIC LOWER BOUNDS

Sometimes we want to say that algorithms take *at least* a certain time to run as a function of the input size n .

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

PRELIMINARY DEFINITION (LOWER BOUND)

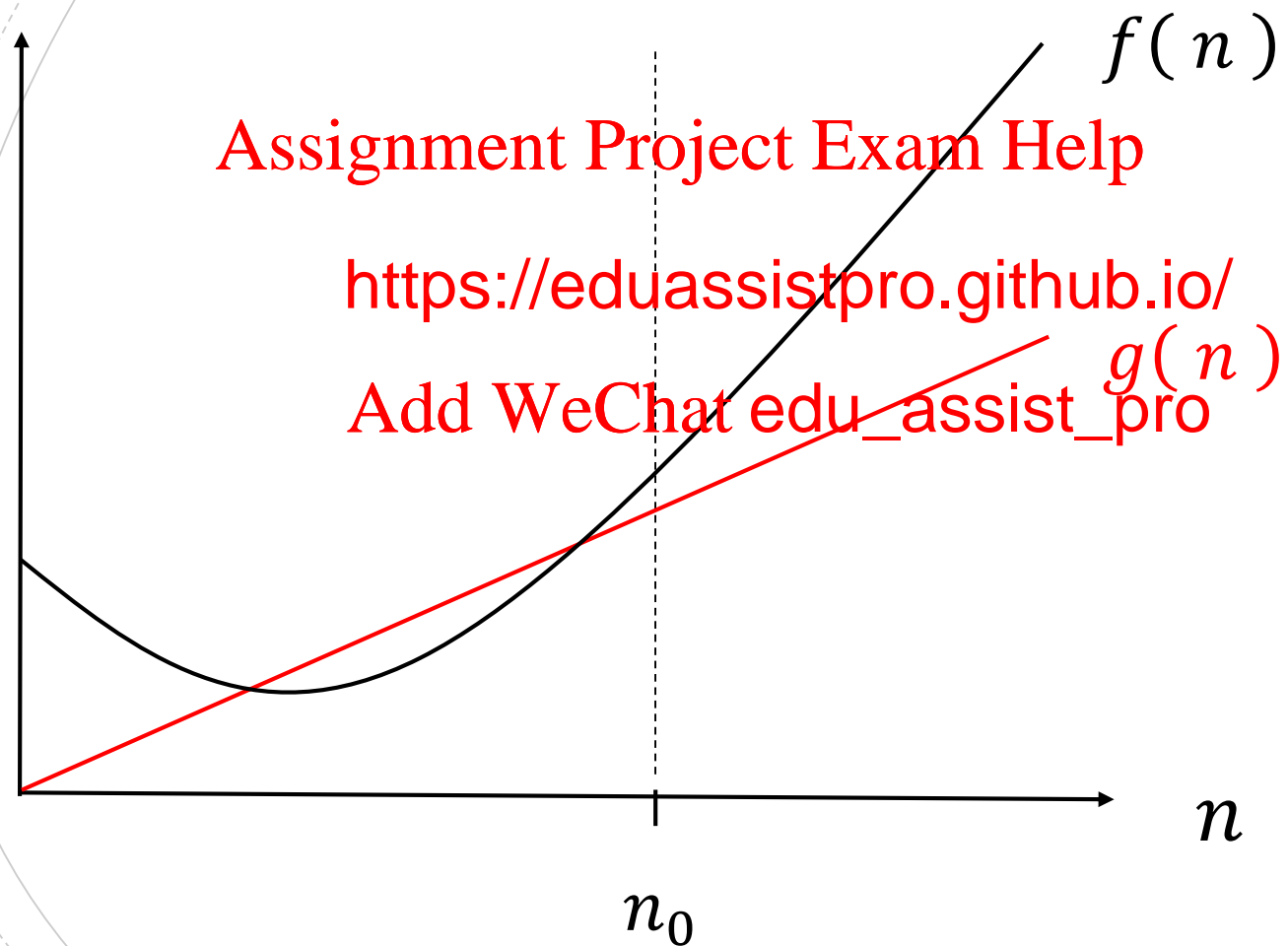
$f(n)$ is *asymptotically bounded below* by $g(n)$ if there exists an n_0 such that, for all $n \geq n_0$,

<https://eduassistpro.github.io/>

$f(n) \geq g(n)$
Add WeChat edu_assist_pro

Note: As with big O , the constant n_0 is not unique. If the definition works for some n_0 then it will work for larger n_0 too.

GRAPHICALLY



EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is asymptotically bounded below by $g(n) = \frac{n^2}{4}$.

Assignment Project Exam Help

To prove: show that there exists n_0 such that $\frac{n(n-1)}{2} \geq \frac{n^2}{4}$ for all $n \geq n_0$.

<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is asymptotically bounded below by $g(n) = \frac{n^2}{4}$.

Assignment Project Exam Help

Proof: $\frac{n(n-1)}{2} \geq \frac{n^2}{4}$

$$\Leftrightarrow 2n(n-1) \geq n^2$$

$$\Leftrightarrow 2n^2 - 2n \geq n^2$$

$$\Leftrightarrow n^2 \geq 2n$$

$$\Leftrightarrow n \geq 2$$

So, we can use $n_0 = 2$.

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

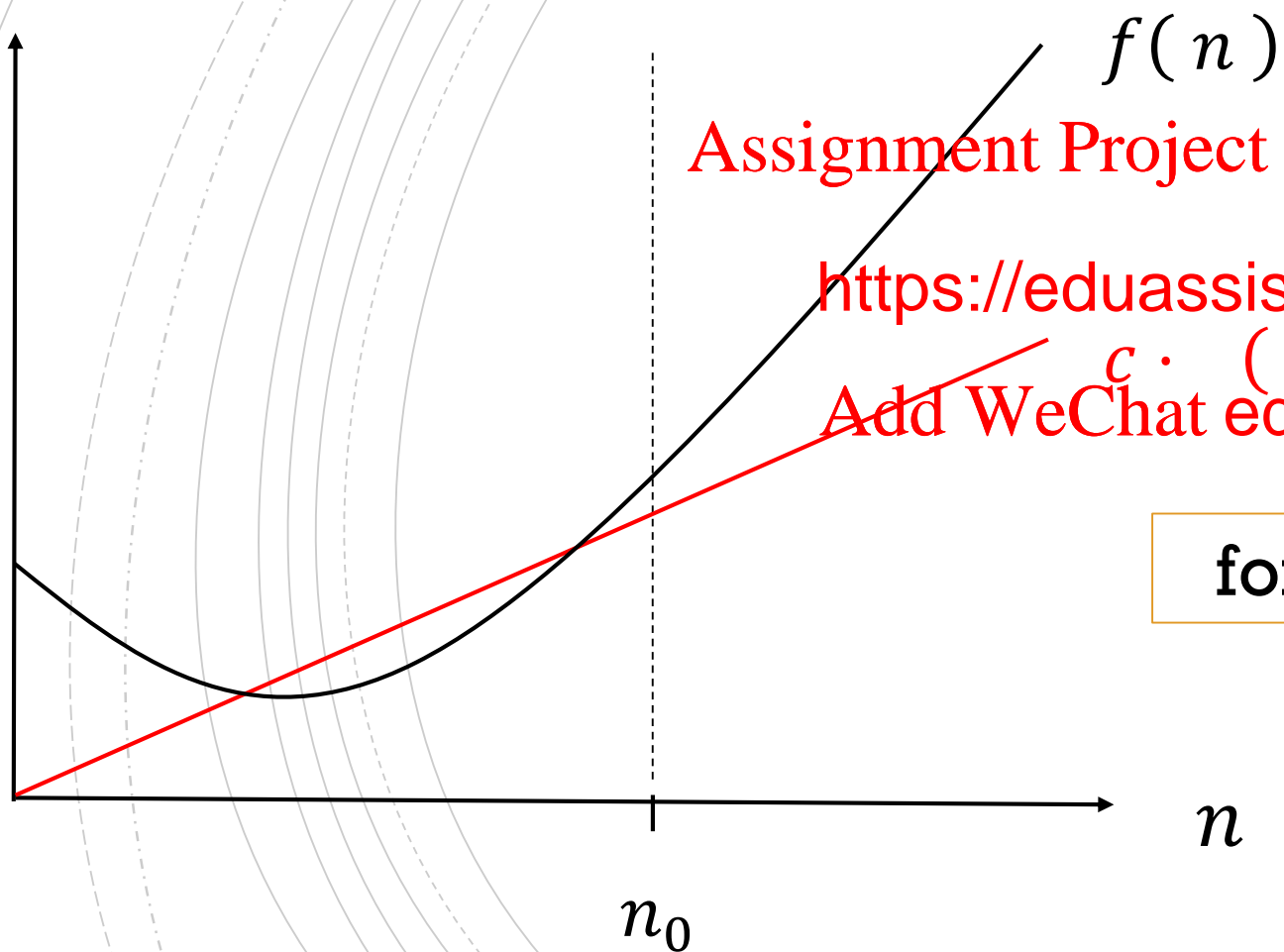
FORMAL DEFINITION OF BIG OMEGA (Ω)

Given a function $g(n)$, we denote by $\Omega(g(n))$ (“big-omega of g of n ”) the following set of functions:

$$\Omega(g(n)) = \{f(n) : \text{there exist positive } c \text{ and } n_0 \text{ such that } f(n) \geq cg(n) \text{ for } n \geq n_0\}$$

We use the Ω -notation to describe an **asymptotic lower bound**.

GRAPHICALLY



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

for all $n_0 \geq n$, $f(n) \geq c \cdot g(n)$

EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is $\Omega(n^2)$.

Proof(1): Use $c = \frac{1}{4}$ and the de

$$\frac{n(n-1)}{2} \geq \frac{n^2}{4}$$

$\Leftrightarrow :$

$$\Leftrightarrow n \geq 2$$

So, we can take $n_0 = 2$ and $c = \frac{1}{4}$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE

Claim: $f(n) = \frac{n(n-1)}{2}$ is $\Omega(n^2)$.

Proof (2): Let's try $c = \frac{1}{3}$

$$\frac{n(n-1)}{2} \geq \frac{n^2}{3}$$

$$\Leftrightarrow 3n(n-1) \geq 2n^2$$

$$\Leftrightarrow n^2 \geq 3n$$

$$\Leftrightarrow n \geq 3$$

So, we can take $n_0 = 3$ and $c = \frac{1}{3}$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

BACK TO INSERTION SORT

At the beginning of last lecture we found the function describing the best-case running time for insertion sort.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

where a , and b are some constants, a

Add WeChat edu_assist_pro

Claim: $T_{best}(n)$ is $\Omega(n)$

$T_{best}(n)$ IS $\Omega(n)$ – PROOF

Claim: $T_{best}(n)$ is $\Omega(n)$

Proof: $T_{best}(n) = an + b$

$$\geq an + \text{https://eduassistpro.github.io/}$$
$$= (a + b)n \text{ Add WeChat edu_assist_pro}$$

So we can take $c = a + b$ (which is positive since it is equal to $T_{best}(1)$) and $n_0 = 1$.

OBSERVATION ON BEST-CASE LOWER BOUNDS

- Since Ω -notation describes a lower bound, when we use it to bound the best-case running time, we have a lower bound on the running time for every input.

That is,

Since $T(n) \geq T_{best}(n)$, if $T_{best}(n) = \Omega(g(n))$ then $T(n) = \Omega(g(n))$

INSERTION SORT

What do we know about the running time of insertion sort up to know?

- We computed $T(n)$, $T_{best}(n)$, and $T_{worst}(n)$
- We have proved that $T_{worst}(n)$ is $O(n^2)$, and $T_{best}(n)$ is $\Omega(n)$
- Therefore, $T(n)$ is both $O(n^2)$ and $\Omega(n)$.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

TRY IT!

Prove that the scaling, sum, product, and transitivity rules all hold for big Omega also.

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

BACK TO THE COMMON FUNCTIONS

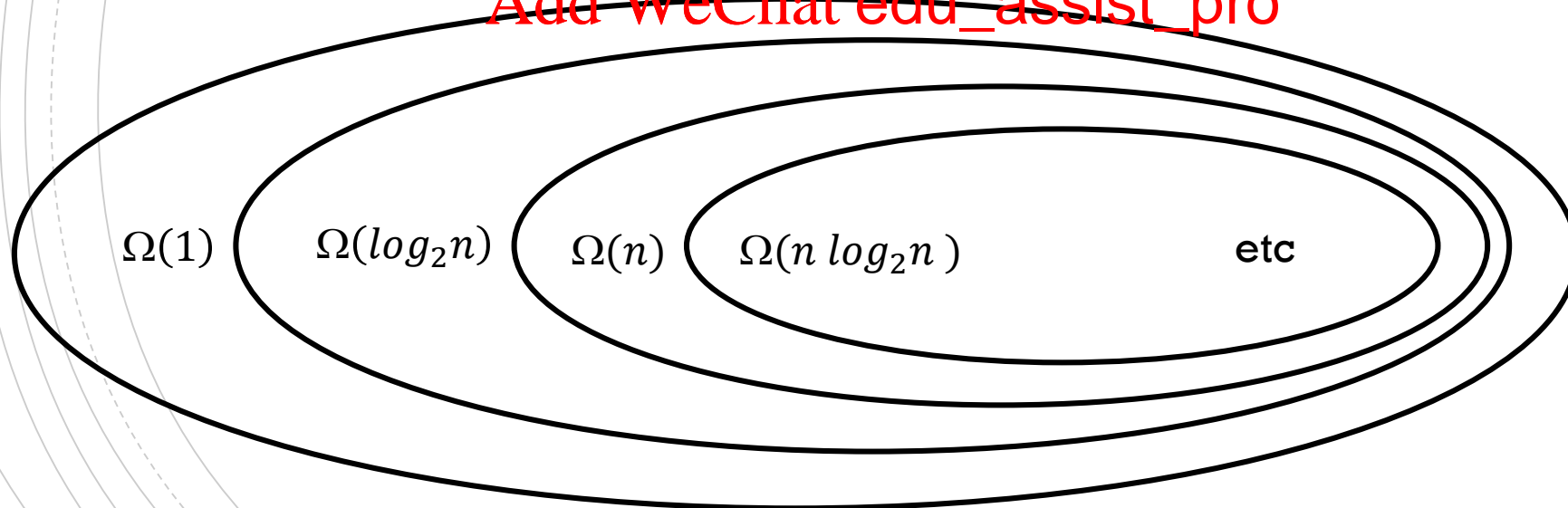
Each of the following holds for n sufficiently large:

$$1 < \log_2 n < n < n \log_2 n < n^2 < n^3 < \dots < 2^n < n!$$

Assignment Project Exam Help

Thus we have the following st <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

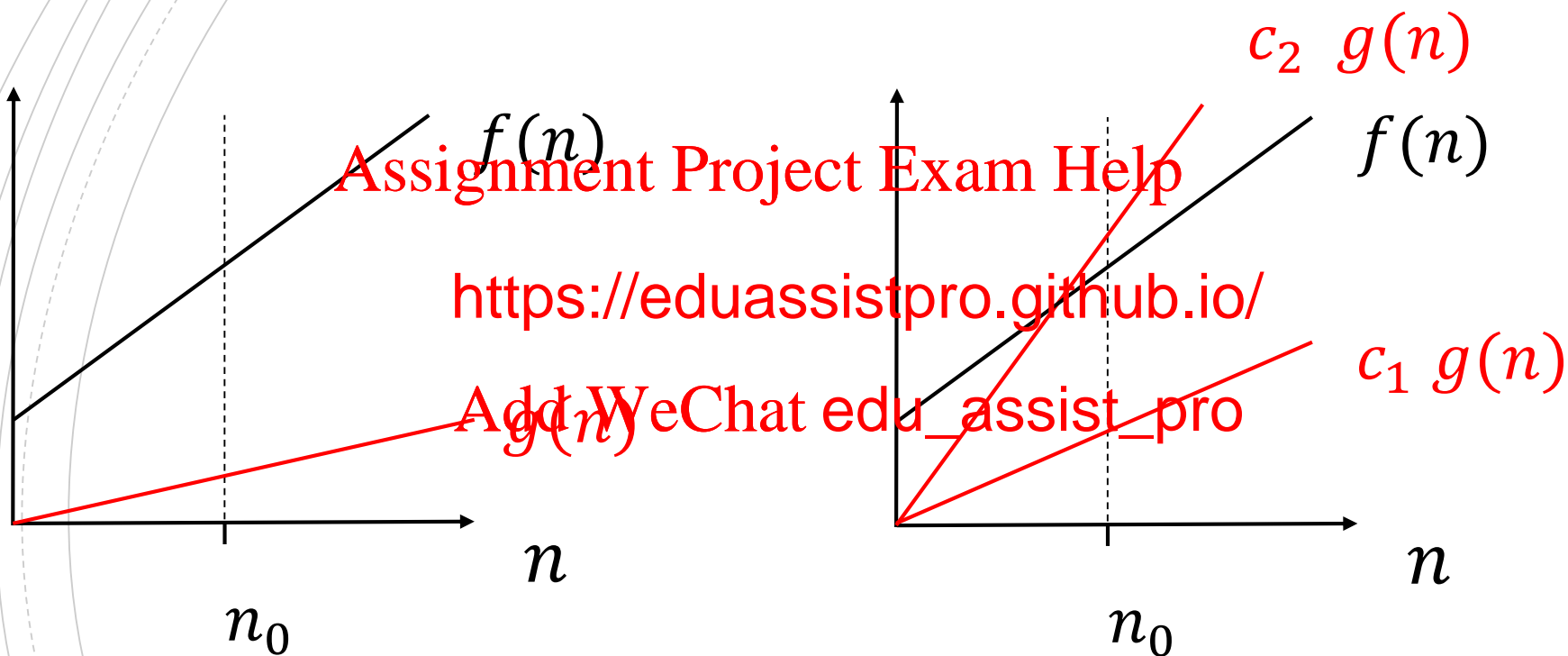
FORMAL DEFINITION OF BIG THETA (Θ)

Given a function $g(n)$, we denote by $\Theta(g(n))$ (“big-theta of g of n ”) the following set of functions

$$\Theta(g(n)) = \{f(n) : \text{there exist positive } c_1, c_2 \text{ and } n_0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for } n \geq n_0\}$$

We use the Θ -notation to describe an **asymptotic tight bound**.

GRAPHICALLY



$f(n)$ is $\Theta(g(n))$.

EXAMPLE

Claim: $f(n) = \frac{1}{2}n^2 - 3n$ is $\Theta(n^2)$.

Proof: We need to find 3 posit

uch that

<https://eduassistpro.github.io/>

$c_1 n^2 \leq \frac{1}{2} n^2 -$
Add WeChat edu_assist_pro

for all $n \geq n_0$.

EXAMPLE

Claim: $f(n) = \frac{1}{2}n^2 - 3n$ is $\Theta(n^2)$.

Proof: We need to find 3 positive constants c_1, c_2 , and n such that

<https://eduassistpro.github.io/>

$c_1 n^2 \leq \frac{1}{2}n^2 -$
Add WeChat edu_assist_pro

for all $n \geq n_0$.

Dividing by n^2 we get

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

The right hand inequality holds for all $n \geq 1$ if we chose any $c_2 \geq \frac{1}{2}$.

The left hand inequality holds for all $n \geq 7$ if we chose any $c_1 \leq \frac{1}{14}$.

EXAMPLE

Claim: $f(n) = \frac{1}{2}n^2 - 3n$ is $\Theta(n^2)$.

Proof: We need to find 3 positive constants c_1, c_2 and n_0 such that

Assignment Project Exam Help

uch that

<https://eduassistpro.github.io/>

$c_1 n^2 \leq \frac{1}{2}n^2 - 3n$
Add WeChat edu_assist_pro

for all $n \geq n_0$.

The right hand inequality holds for all $n \geq 1$ if we chose any $c_2 \geq \frac{1}{2}$.

The left hand inequality holds for all $n \geq 7$ if we chose any $c_1 \leq \frac{1}{14}$.

Pick $n_0 = 7, c_1 = 1/14, c_2 = 1/2$.

DEFINITION OF BIG THETA (Θ)

Let $f(n)$ and $g(n)$ be two functions of $n \geq 0$.

Assignment Project Exam Help

We say $f(n)$ is $\Theta(g(n))$ if there exist positive constants

n_0 , c_1 , c_2 such that for all $n \geq n_0$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$f(n)$ is $O(g(n))$

DEFINITION OF BIG THETA (Θ)

Let $f(n)$ and $g(n)$ be two functions of $n \geq 0$.

Assignment Project Exam Help

We say $f(n)$ is $\Theta(g(n))$ if there exist positive constants

n_0 , c_1 , c_2 such that for all $n \geq n_0$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

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

THEOREM

For any two functions $f(n)$ and $g(n)$,

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

<https://eduassistpro.github.io/>

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

Add WeChat edu_assist_pro

if and only if

EXAMPLE 2

Claim: $f(n) = 4 + 17 \log_2 n + 3n + 9n \log_2 n + \frac{n(n-1)}{2}$ is $\Theta(n^2)$.

Assignment Project Exam Help

Proof:

$$\frac{n^2}{4} \leq f(n) \leq \left(4 + 1 + \frac{1}{2}\right) n^2$$

<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

In general, you want to set c_1 to be a value that is slightly smaller than the coefficient of the highest-order term and c_2 to be a value that is slightly larger.

DOES A TIGHT BOUND ALWAYS EXIST?

For every $f(n)$, does there exist a “simple” $g(n)$ such that $f(n)$ is $\Theta(g(n))$?

No, as this contrived example shows:

$$\text{Let } f(n) = \begin{cases} n, & n \text{ is odd} \\ n^2, & n \text{ is even} \end{cases}$$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

$f(n)$ is $O(n^2)$, but $f(n)$ is *not* $O(n)$.

$f(n)$ is $\Omega(n)$, but $f(n)$ is *not* $\Omega(n^2)$.

DOES A TIGHT BOUND ALWAYS EXIST?

We can also think about the function representing the running time of insertion sort.

Assignment Project Exam Help

$T_{best}(n) \in \Theta(n)$ and $T_{worst}(n) \in \Theta(n^2)$

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

$\Rightarrow T(n) \in O(n^2), T(n) \in \Omega(n)$

AND

$T(n) \notin \Theta(n), T(n) \notin \Theta(n^2)$

One that it is improper to say that is $O(n^2)$ (for instance), since for even n , the actual running time varies, depending on the particular input. When we say that, what we mean is that there exists a function $f(n)$ which is $O(n^2)$ and T is bounded above by f , no matter the particular input of size n .

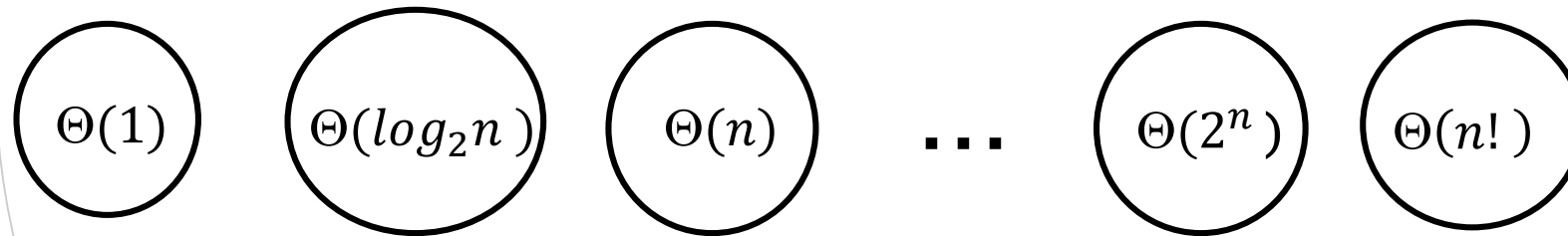
SETS OF $\Theta()$ FUNCTIONS

Each of the following holds for n sufficiently large:

$$1 < \log_2 n < n < n \log_2 n < n^2 < n^3 < \dots < 2^n < n!$$

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



An orange paint roller with a red handle, positioned horizontally. The roller is covered in orange paint, which is dripping down the left side. The text "Coming Soon" is written in white on the orange surface of the roller.

Coming Soon

Assignment Project Exam Help

In the next

- Stacks a <https://eduassistpro.github.io/>
Add WeChat edu_assist_pro