# Lecture 6 - January 23

## Asymptotic Analysis of Algorithms

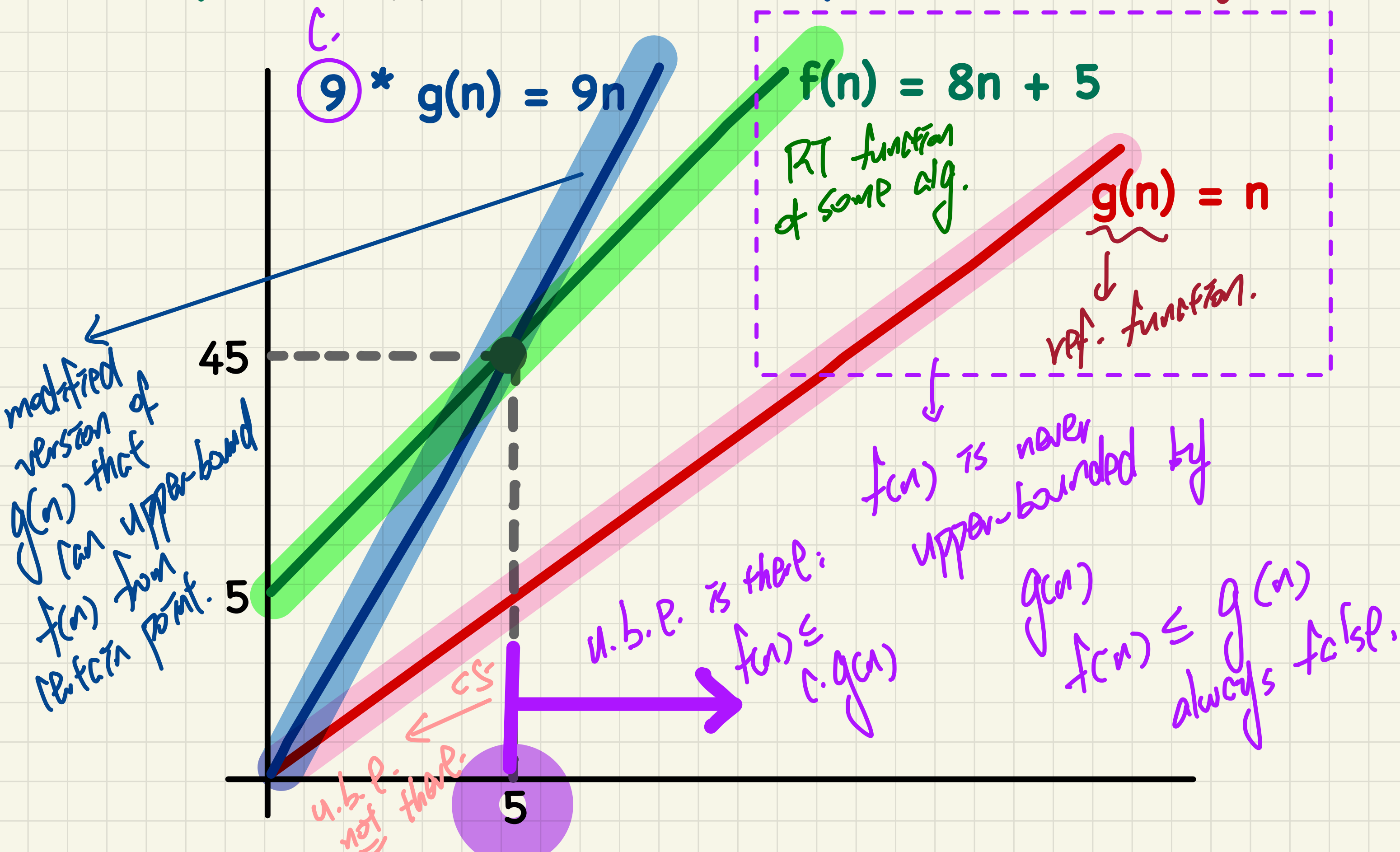
***Big-O: Pred. Def., Properties, Examples***
***Correct vs. Accurate Asymptotic U.B.***
***Deriving U.B. from Code: Basic Examples***

# Announcements/Reminders

- Assignment 1 due next Monday
- ***splitArrayHarder***: an extended version released
- Office Hours: 3pm to 4pm, Mon/Tue/Wed/Thu
- Contact Information of TAs on common eClass site

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

# Asymptotic Upper Bound: Example



# Asymptotic Upper Bound (Big-O): Alternative Formulation

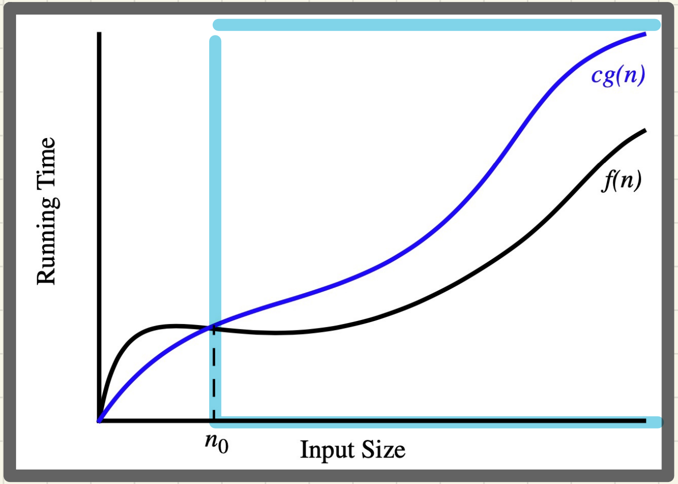
## Known:

$f(n) \in O(g(n))$ if there are:

- A real *constant* $c > 0$
- An integer *constant* $n_0 \geq 1$

such that:

$$f(n) \leq c \cdot g(n) \quad \text{for } n \geq n_0$$




**Q. Formulate** the definition of "**f(n)** is order of O(**g(n)**)" using **logical** operator(s): ¬, ∧, ∨, ⇒, ∀, ∃

$$f(n) \in O(g(n)) \iff \exists c, n_0 \cdot \underset{\substack{c > 0 \,\wedge \\ n_0 \geq 1}}{\wedge} \left( \forall n \cdot n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n) \right)$$

$* \; 1^0 = 1^1 = \cdots = 1^d = 1$

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

We prove by choosing

$$\boxed{\begin{array}{c} c \\ n_0 \end{array}} \begin{array}{cl} = & \underline{|a_0| + |a_1| + \cdots + |a_d|} \\ = & 1 \end{array}$$

$(1)\; f(1) \le c \cdot \cancel{n}^d_{1}$

$(2)\; f(n) \le c \cdot n^d$

$(n > 1)$

If $f(n)$ is a polynomial of degree $d$, i.e.,

$$f(\cancel{n}^{1}) = \underline{a_0 \cdot \cancel{n}^{0} + a_1 \cdot \cancel{n}^{1} + \cdots + a_d \cdot n^d}$$

and $a_0, a_1, \ldots, a_d$ are integers (i.e., negative, zero, or positive), then $f(n)$ is $O(n^d)$. $\rightarrow g(n)$

$\checkmark$

$**$

$a_0 \le |a_0|$

$a_1 \le |a_1|$

$\vdots$

$a_d \le |a_d|$

**Upper-bound effect: $n_0 = 1$?** $\quad [f(1) \le (|a_0| + |a_1| + \cdots + |a_d|) \cdot 1^d]$

$$f(1) = a_0 \cdot 1^0 + a_1 \cdot 1^1 + \cdots + a_d \cdot 1^d$$

$$\overset{*}{=} (a_0 + a_1 + \cdots + a_d) \cdot 1^d \underset{**}{\le} (|a_0| + |a_1| + \cdots + |a_d|) \cdot 1^d$$

$***$

$n^0 \le n^d$

$n^1 \le n^d$ $\checkmark$

$\vdots$

$n^d \le n^d$

**Upper-bound effect holds?** $\quad [f(n) \le (|a_0| + |a_1| + \cdots + |a_d|) \cdot n^d]$ $\boxed{n > 1}$

$$f(n) = a_0 \cdot \overset{\le n^d}{n^0} + a_1 \cdot \overset{\le n^d}{n^1} + \cdots + a_d \cdot \overset{\le n^d}{n^d}$$

$$\overset{***}{\le} (a_0 + a_1 + \cdots + a_d) \cdot n^d \overset{**}{\le} (|a_0| + |a_1| + \cdots + |a_d|) \cdot n^d$$

## Exercise: Prove $f(n) = 5n^4 - 3n^3 + 2n^2 - 4n + 1$ is $O(n^4)$

$g(n)$

highest power

$$f(n) = \cancel{5}\boxed{n^4} - \cancel{3n^3} + \cancel{2n^2} - \cancel{4n} + \cancel{1 \cdot n^0}$$

(1) Derive/Guess: $f(n)$ is $O(n^4)$

(2) Prove:

choose $c$: $|5| + |(-3)| + |2| + |(-4)| + |1| = 15$

$\boxed{n_0: 1}$

Verify

$f(1) \le c \cdot g(1)$

$5 - 3 + 2 - 4 + 1 \le 15 \cdot 1$

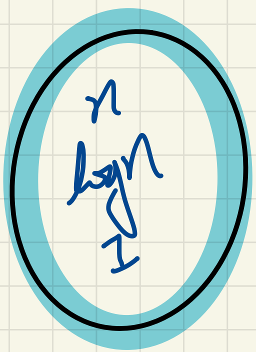
True.

# Big-O Properties (1): Members in a Family

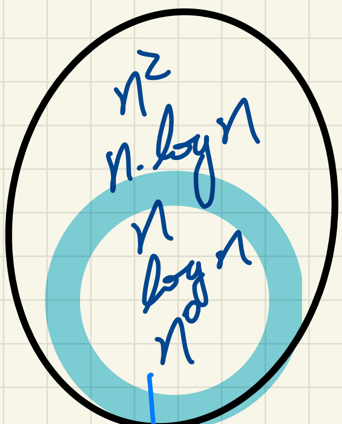
Each member **f(n)** in O( **g(n)** ) is such that:

Higest Power of **f(n)** <= Highest Power of **g(n)**

## O(n)

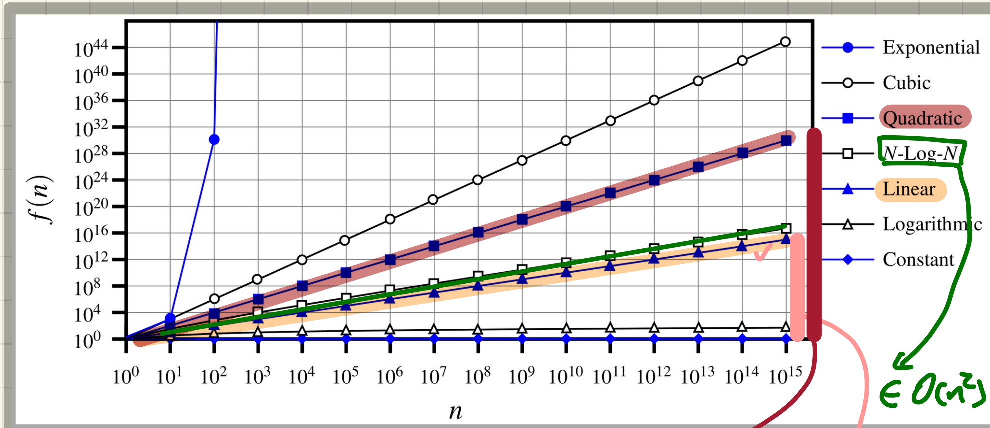



## O(n²)




O(n) "is contained" within O(n²)

## Functions: Rates of Growth




$\in O(n^2)$

$\notin O(n)$

O(n²) contains O(n) plus other functions growing not as fast as n²

= O(n) contains all functions growing no faster than n.

# Big-O Properties (2): Relating Families

$O(n) \ \underline{\ ?\ }\ O(n^2)$

Correct

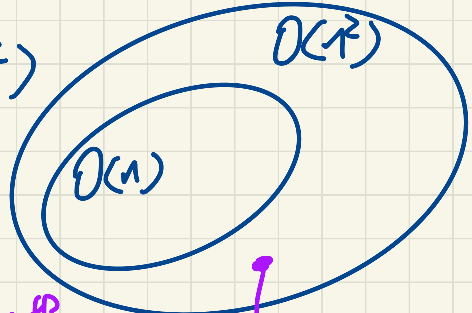
(1) $\subseteq$

(2) $\boxed{\subset}$

(3) $\supseteq$

(4) $\supset$

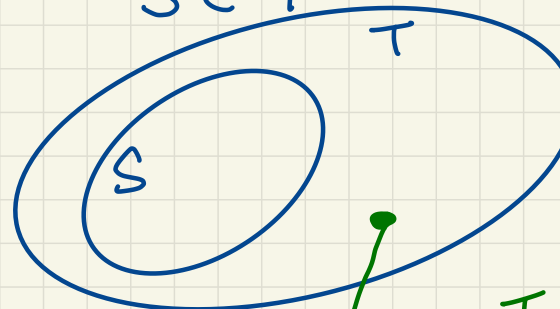
more accurate.




a function that can be u.b. by $n^2$ but cannot be u.b. by $n$

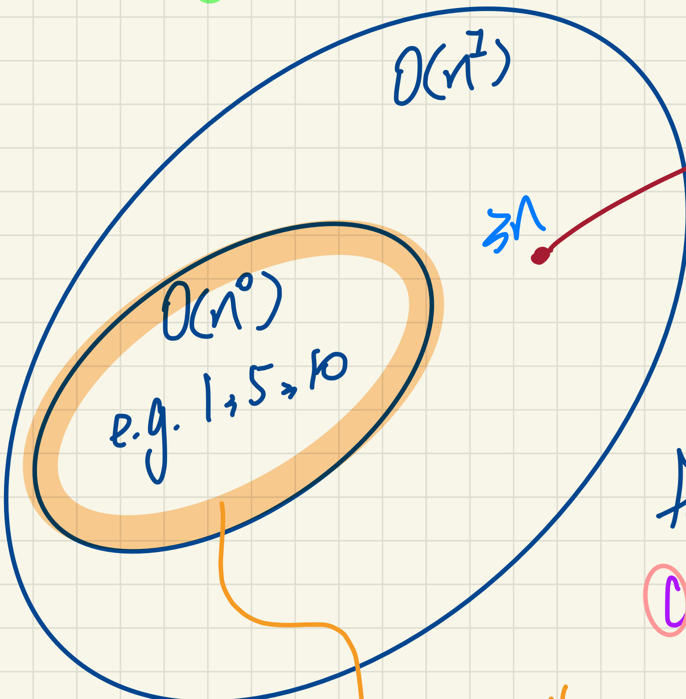
e.g. $n^2$, $2 \cdot n^2$, $n \cdot \log n$

$S \subset T$




a member in T but not in S

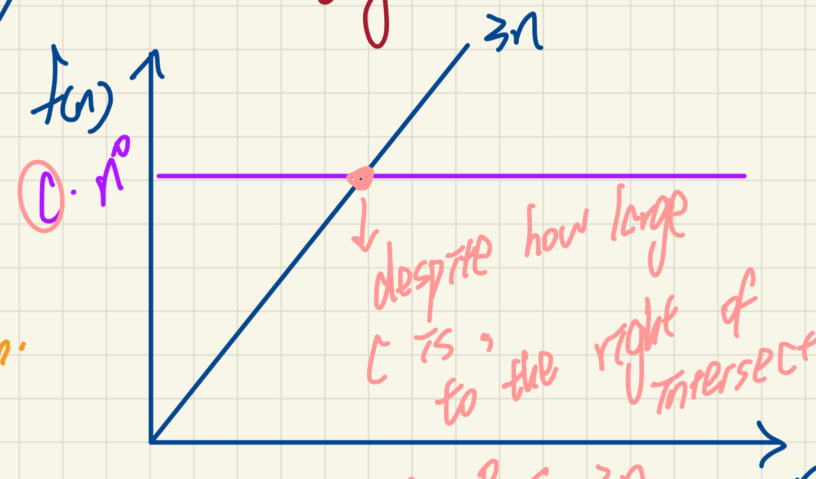
# $O(n^0) \subset O(n)$




at least one function that can be u.b. by $n^1$ but cannot be u.b. by $n^0$

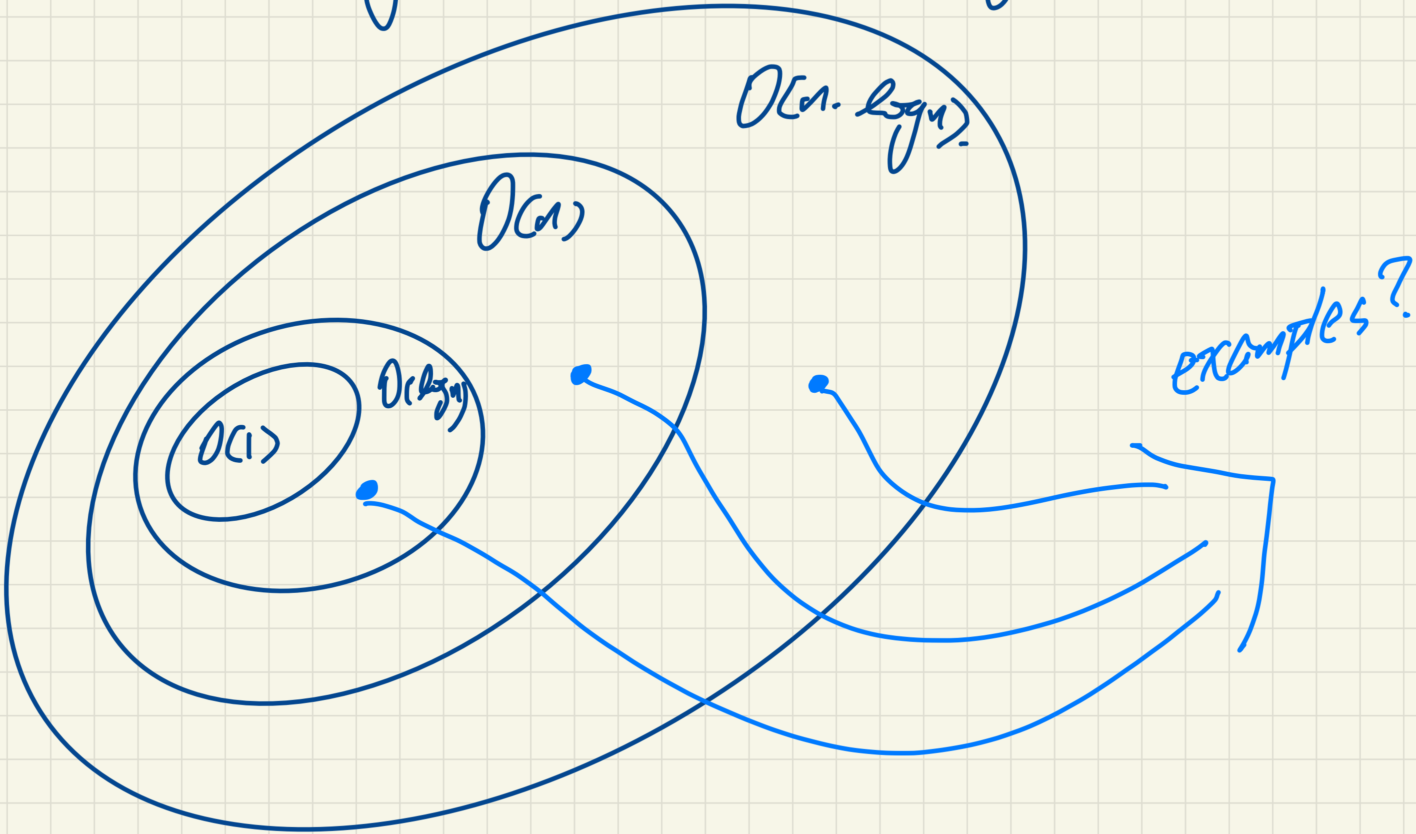
e.g. $f(n) = 3n$

set of functions that can be u.b. by $n^0$




$\underbrace{c \cdot n^0 \leq 3n}_{\text{no}}$ u.b.e.

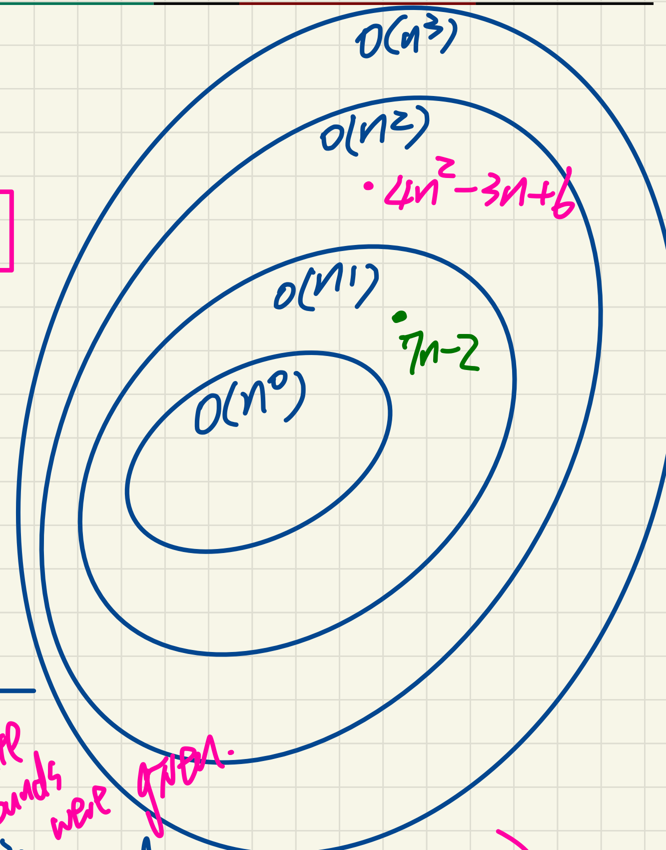
$$O(\underline{n^0}) \subset O(\underline{\log n}) \subset O(\underline{n}) \subset O(\underline{n \cdot \log n}) \subset \cdots$$



# Big-O Properties (3): Deciding Correct & Accurate Bound

e.g. $f_1(n) = \boxed{7n - 2}$

$f_2(n) = \boxed{4n^2 - 3n + 6}$




$7n - 2$ is order of $\boxed{O(n)}$
( $\in$ )

$O(n^2)$

$O(n^3)$

$O(2^n)$

Correct.

smallest power that can upper-bound $7n - 2$ ⇒ most accurate a.u.b.

Misleading!: not accurate bounds were given.

$f_1(n)$ is $O(n^3)$, $f_2(n)$ is $O(n^2)$

$O(7n^2 + 4n - 2)$ ✗ → always exclude lower terms and mul. constants.

$\underline{O(n^2)}$ ✓

## Asymptotic Upper Bounds: Example (1)

Given $f(\cancel{n}^{1}) = 5n^2 + 3n \cdot \log n + 2n + 5$:

(1) What is f(n)'s **most accurate** **asymptotic upper bound**.

(2) **Prove** your claim.

(1) $O(n^2)$

(2) Choose $c: |5| + |3| + |2| + |5| = 15$.

$n_0: 1$

Verify $f(1) \leq 15 \cdot 1^2$

u.b.e. starts at $n_0 = 1$.

# Asymptotic Upper Bounds: Example (2) (Exercise)

Given $f(n) = 20n^3 + 10n \cdot \log n + 5$:

(1) What is $f(n)$'s **most accurate** **asymptotic upper bound.**

(2) **Prove** your claim.

# Asymptotic Upper Bounds: Example (3)

Given $f(n) = 3 \cdot \log n + 2$:

(1) What is f(n)'s **most accurate** **asymptotic upper bound**.

(2) **Prove** your claim.

(1) $O(\log n)$ $g(n)$

(2) Choose: $c = |3| + |2| = 5$.

$n_0 = 1$ × → Verify: $\boxed{f(1)} \leq 5 \cdot \boxed{\log 1}$

$0$ ∵ $2^0 = 1$

$3 \cdot \log 1 + 2$

$\boxed{2} \leq 0$

u.b.e. not there.

what about $n_0 = 2$

↳ $f(2) \leq 5 \cdot \log 2$

u.b.e.?

(Exercise)

## Asymptotic Upper Bounds: Example (4) (Exercise)

Given $f(n) = 2^{n+2}$:

(1) What is f(n)'s most accurate asymptotic upper bound.

(2) Prove your claim.

# Asymptotic Upper Bounds: Example (5) (Exercise)

Given $f(n) = 2n + 100 \cdot \log n$:

(1) What is $f(n)$'s **most accurate** **asymptotic upper bound**.

(2) **Prove** your claim.

# Determining the Asymptotic Upper Bound (1)

```
int maxOf (int x, int y) {
  int max = x;      1
  if (y > x) {      1
    max = y;        1
  }
  return max;       1
}
```

$O(1+1+1+1)$

$= O(4)$

$4n^0$

$O(1)$

# Determining the Asymptotic Upper Bound (2)

```
int findMax (int[] a, int n) {
  currentMax = a[0];            1
  for (int i = 1; i < n; ) {    n
    if (a[i] > currentMax) {    n
      currentMax = a[i]; }      n
    i ++ }                      n
  return currentMax; }          1
```

$O(1 + n + n + n + n + 1)$

$O(4n + 2)$

$O(n)$