Analysis of Algorithms

The Time Complexity of an Algorithm

Specifies how the running time depends on the size of the input.

Purpose

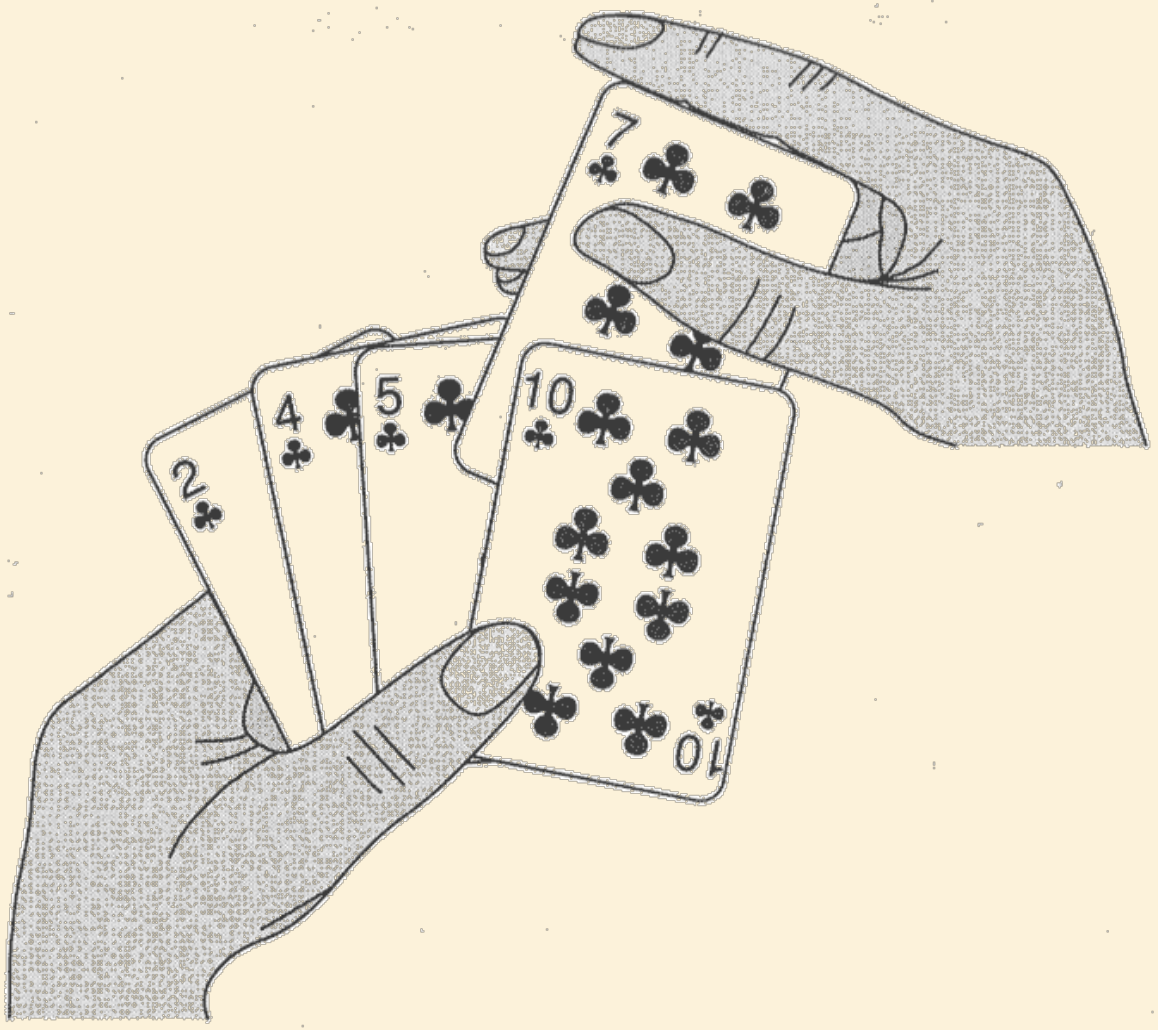
- To estimate how long a program will run.
- To estimate the largest input that can reasonably be given to the program.
- To compare the efficiency of different algorithms.
- To help focus on the parts of code that are executed the largest number of times.
- To choose an algorithm for an application.

Time Complexity Is a Function

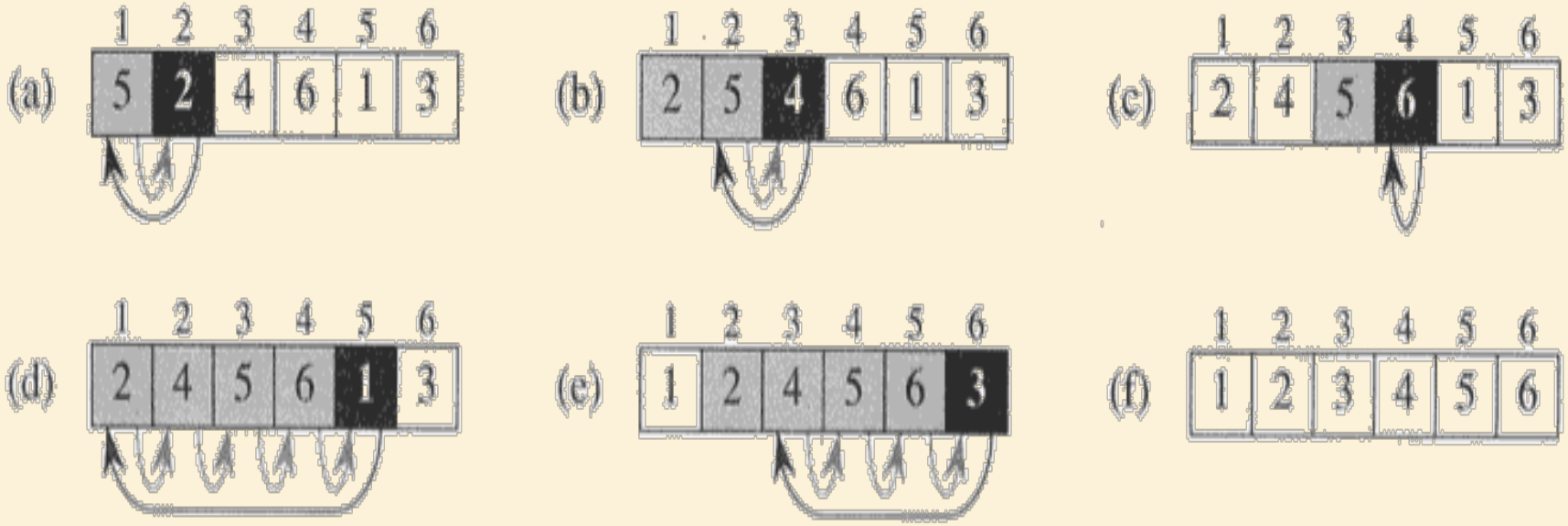
- Specifies how the running time depends on the size of the input.

- A function mapping “size” of input → “time” T(n) executed .

Example: Insertion Sort



Example: Insertion Sort

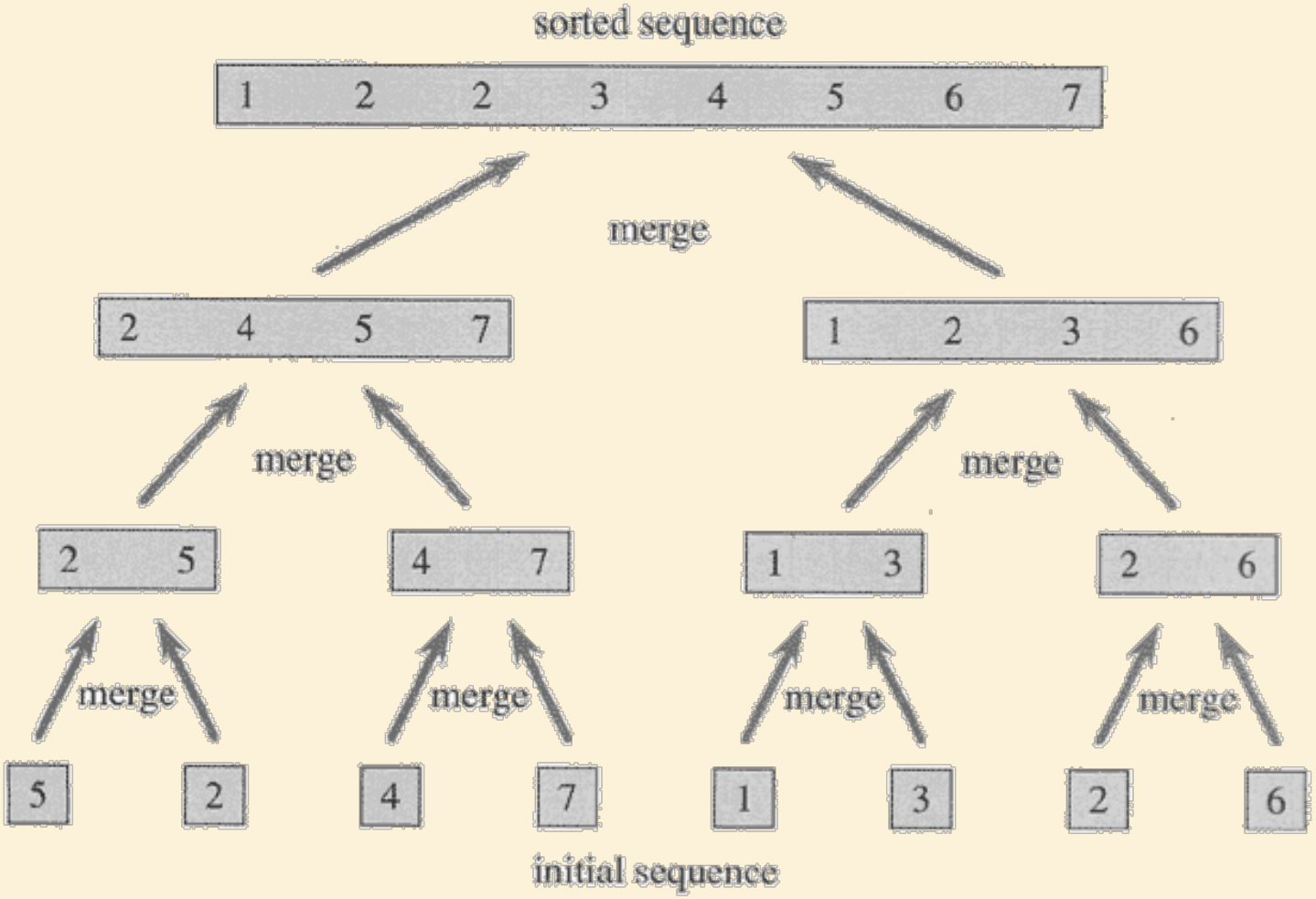


Example: Insertion Sort

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j \leftarrow 2$ **to** *length*$[A]$	c_1	n
2 **do** $key \leftarrow A[j]$	c_2	$n-1$
3 ▷ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
4 $i \leftarrow j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] \leftarrow key$	c_8	$n-1$

Worst case (reverse order): $t_j = j$: $\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1 \rightarrow T(n) \in \theta(n^2)$

Example: Merge Sort

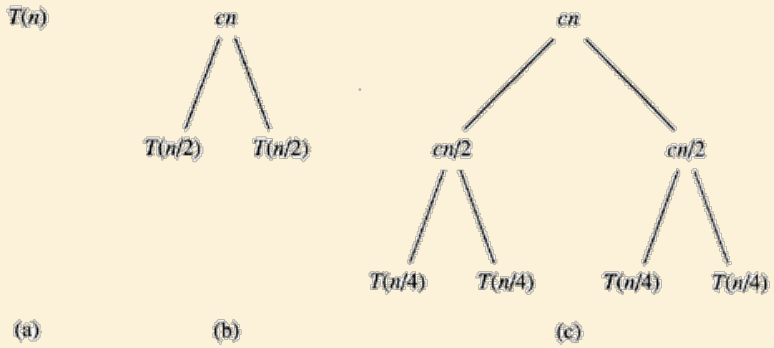


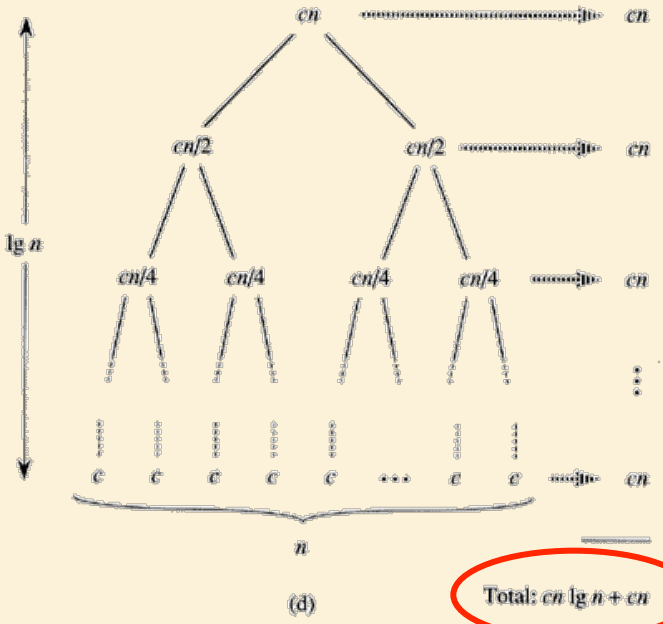
```
MERGE-SORT(A, p, r)
1  if p < r
2     then q ← ⌊(p + r)/2⌋
3          MERGE-SORT(A, p, q)
4          MERGE-SORT(A, q + 1, r)
5          MERGE(A, p, q, r)
```

Example: Merge Sort

```
MERGE(A, p, q, r)
 1  n1 ← q − p + 1
 2  n2 ← r − q
 3  create arrays L[1 .. n1 + 1] and R[1 .. n2 + 1]
 4  for i ← 1 to n1
 5       do L[i] ← A[p + i − 1]
 6  for j ← 1 to n2
 7       do R[j] ← A[q + j]
 8  L[n1 + 1] ← ∞
 9  R[n2 + 1] ← ∞
10  i ← 1
11  j ← 1
12  for k ← p to r
13       do if L[i] ≤ R[j]
14            then A[k] ← L[i]
15                 i ← i + 1
16            else A[k] ← R[j]
17                 j ← j + 1
```

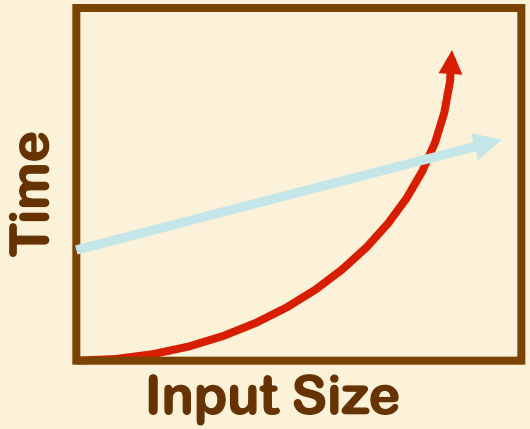
Example: Merge Sort





$T(n) \in \theta(n \log n)$

Relevant Mathematics: Classifying Functions



Assumed Knowledge

Chapter 22. Existential and Universal Quantifiers

$$\exists g \forall b \ \text{Loves}(b, g)$$

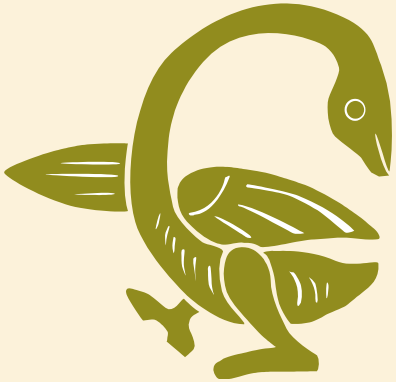
$$\forall g \exists b \ \text{Loves}(b, g)$$

Chapter 24. Logarithms and Exponentials

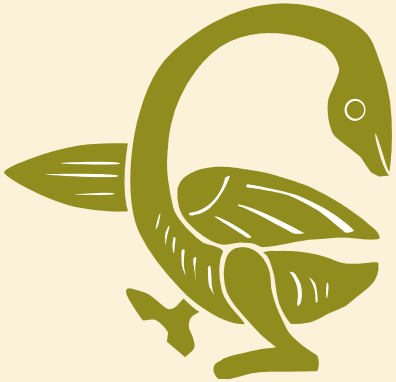
Classifying Functions

Describing how fast a function grows without going into too much detail.

Which are more alike?

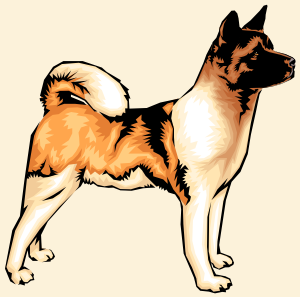


Which are more alike?

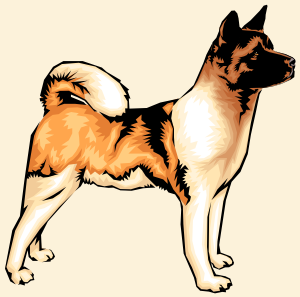


Mammals

Which are more alike?

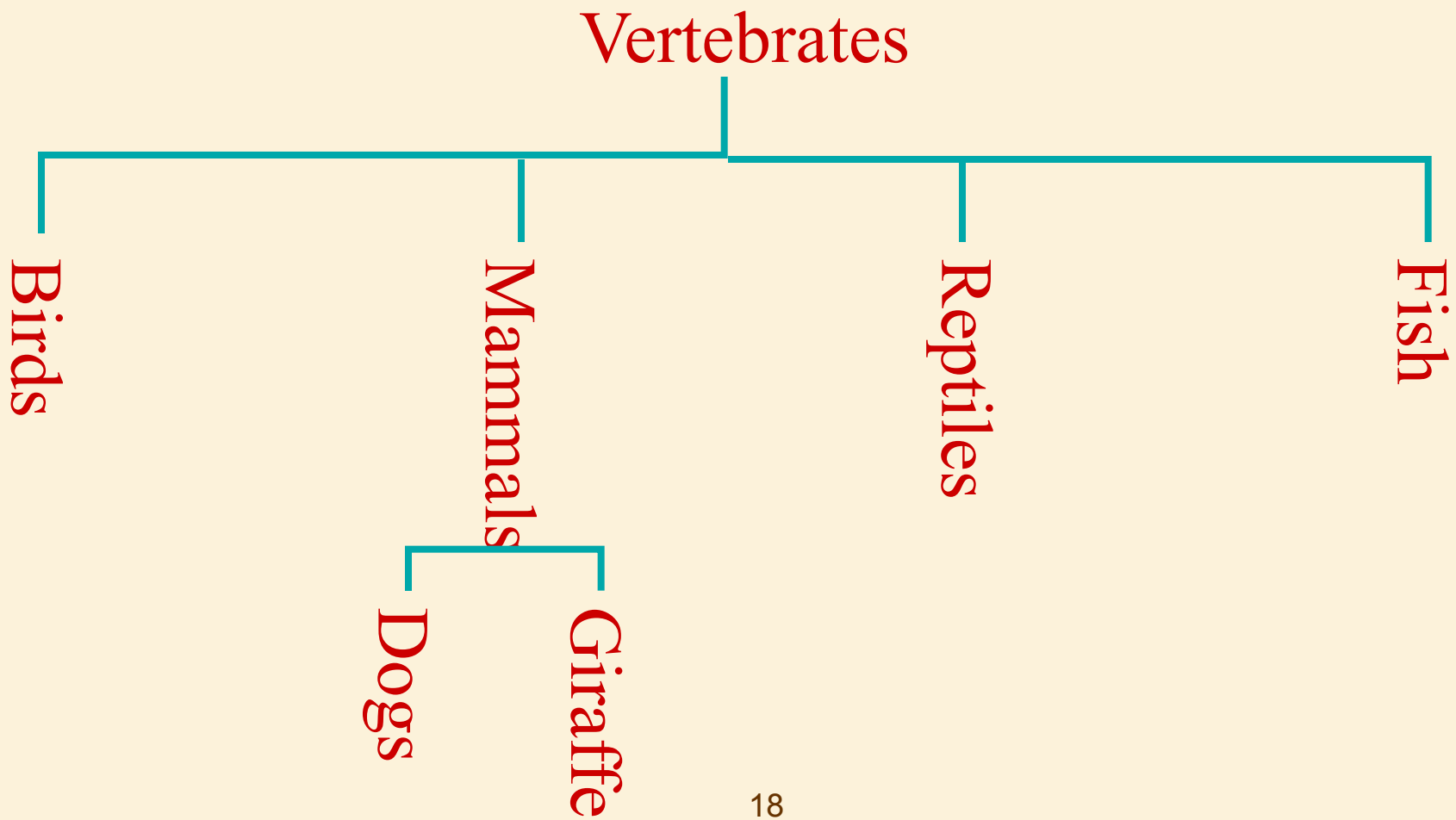


Which are more alike?



Dogs

Classifying Animals



Classifying Functions

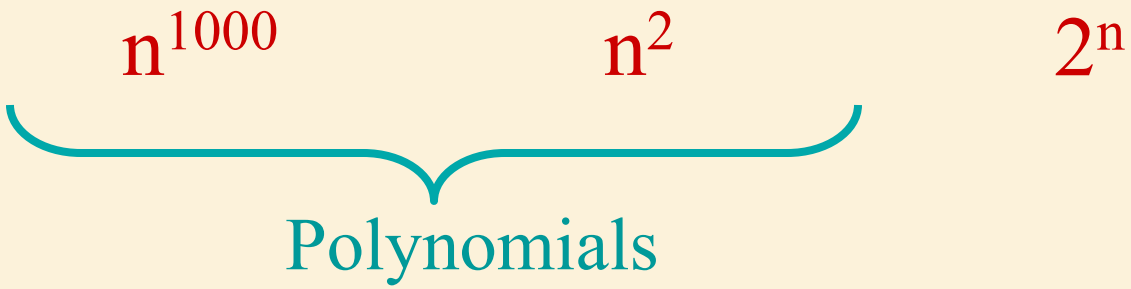
	n			
$T(n)$	10	100	1,000	10,000
$\log n$	3	6	9	13
$n^{1/2}$	3	10	31	100
n	10	100	1,000	10,000
$n \log n$	30	600	9,000	130,000
n^2	100	10,000	10^6	10^8
n^3	1,000	10^6	10^9	10^{12}
2^n	1,024	10^{30}	10^{300}	10^{3000}

Note: The universe is estimated to contain ~10^{80} particles.

Which are more alike?

n^{1000} n^2 2^n

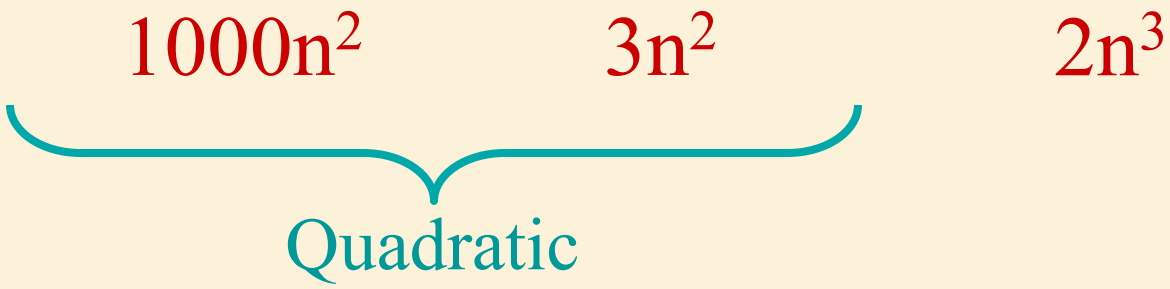
Which are more alike?



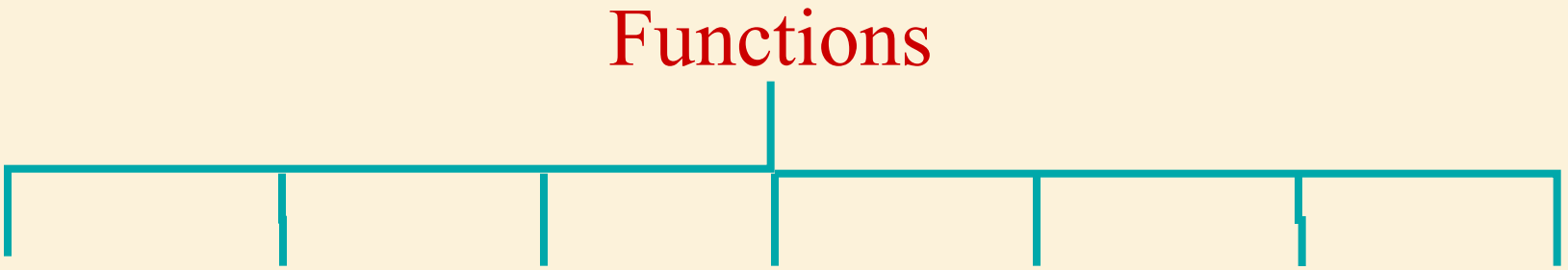
Which are more alike?

$1000n^2$ $3n^2$ $2n^3$

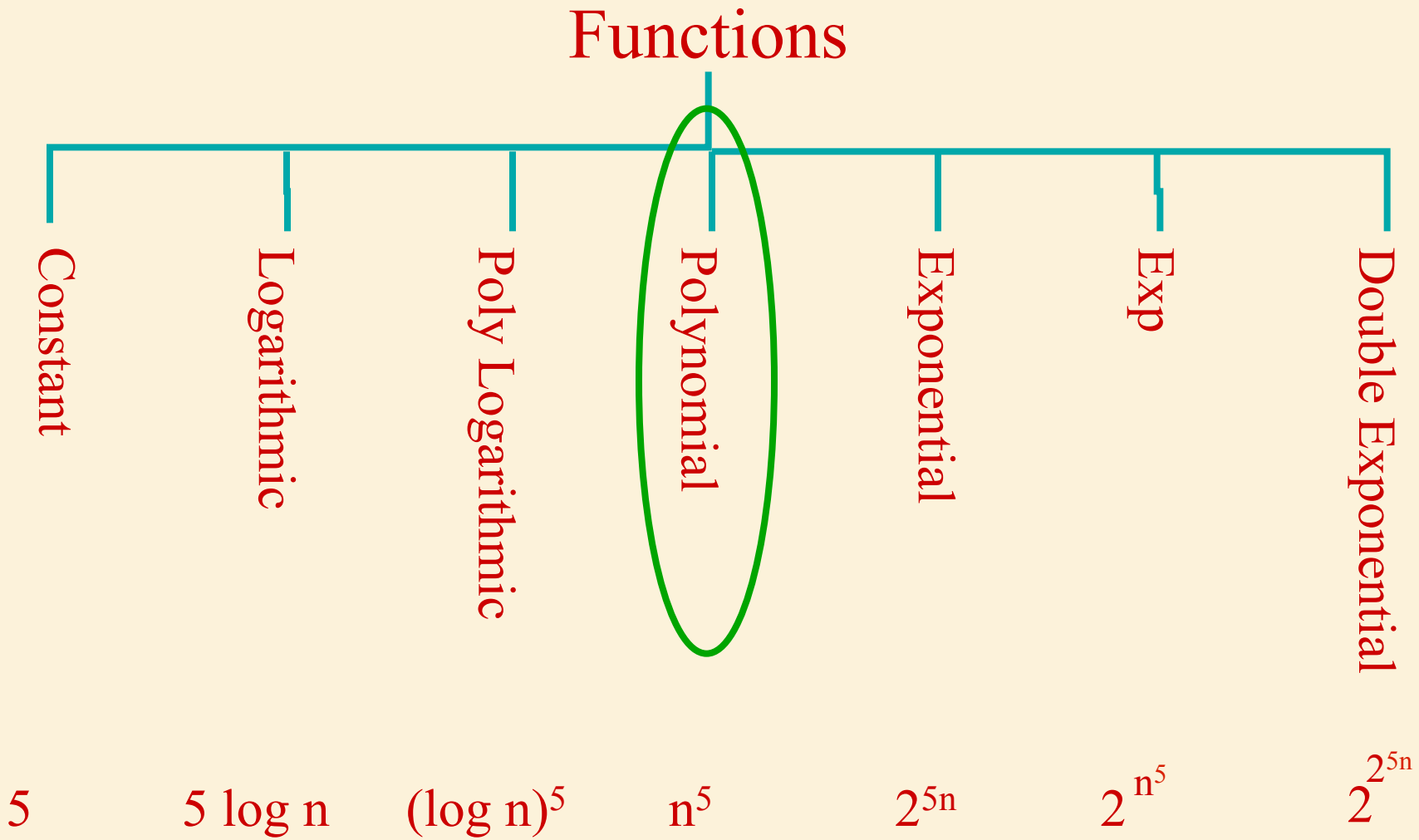
Which are more alike?



Classifying Functions?



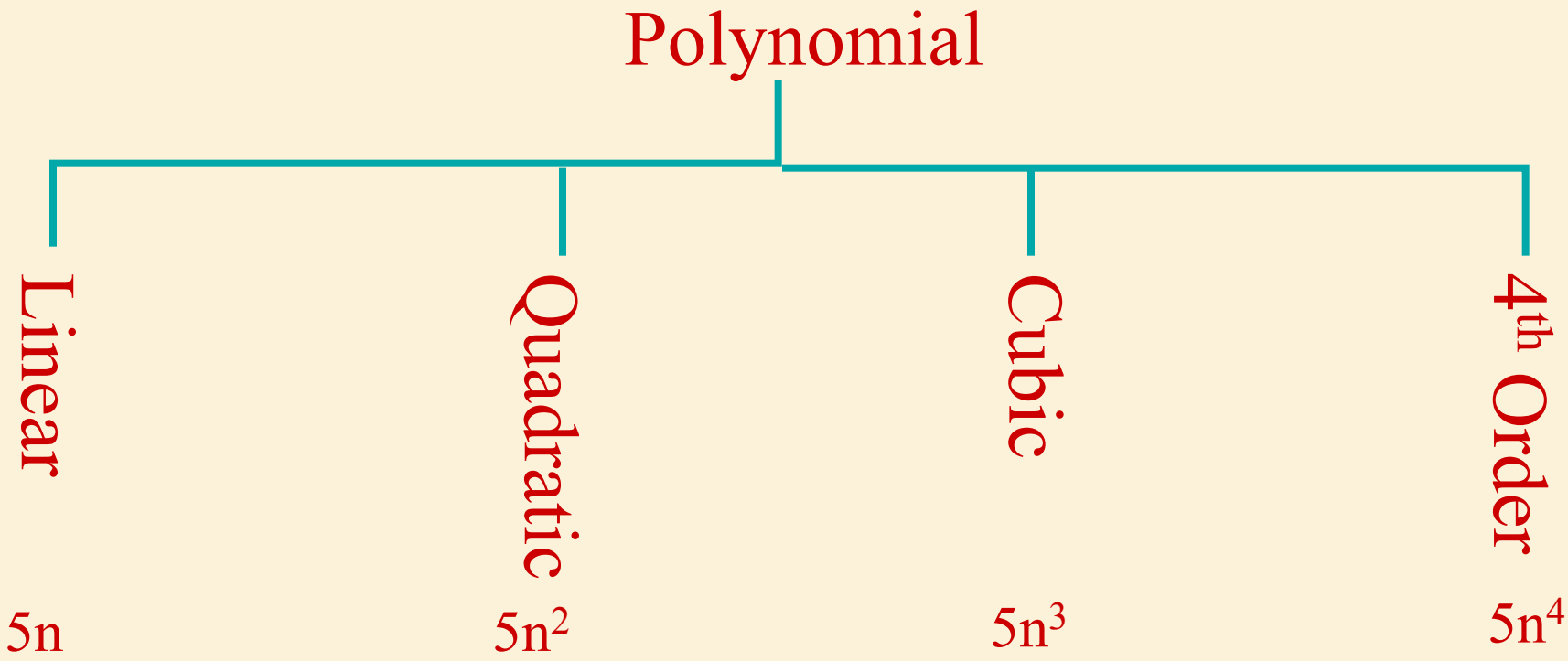
Classifying Functions



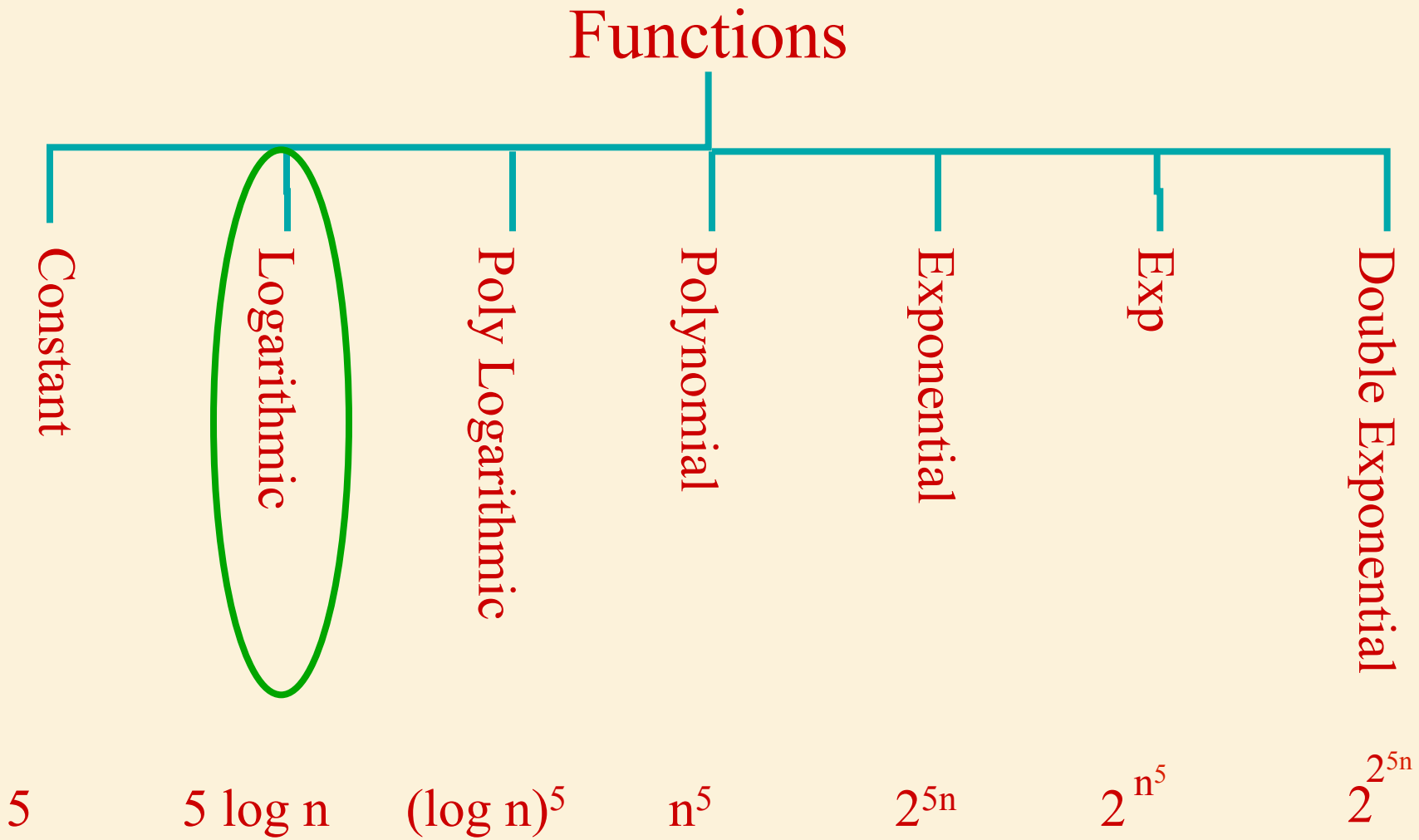
Classifying Functions?

Polynomial

Classifying Functions



Classifying Functions



Properties of the Logarithm

Changing bases: $\log_a n = \frac{1}{\log_b a} \times \log_b n$

Swapping base and argument: $\log_a b = \frac{1}{\log_b a}$

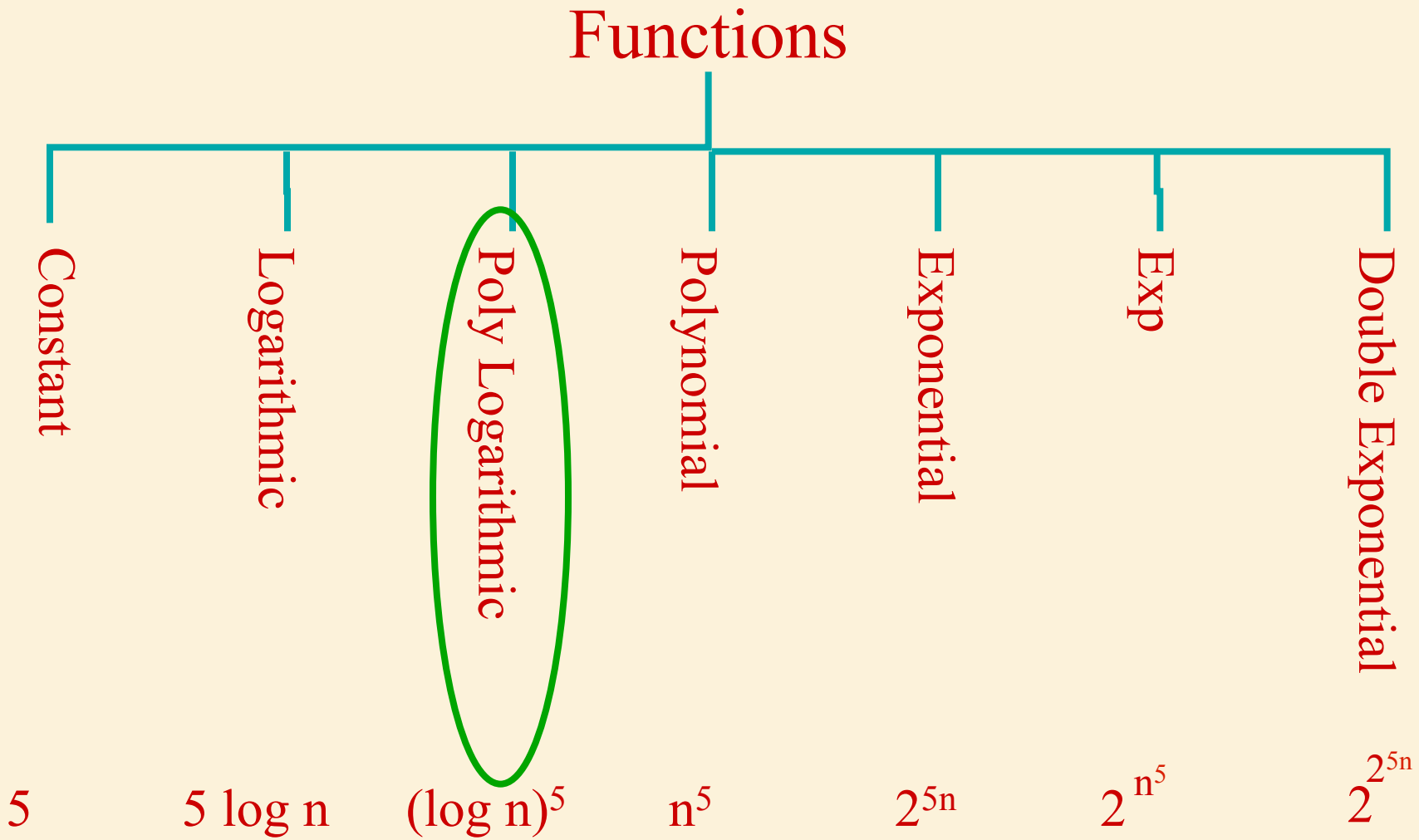
Logarithmic

- $\log_{10} n$ = # digits to write n
- $\log_2 n$ = # bits to write n
  = $3.32 \log_{10} n$
- $\log(n^{1000}) = 1000 \log(n)$

Differ only by a multiplicative constant.

Changing bases: $\log_a n = \frac{1}{\log_b a} \times \log_b n$

Classifying Functions



Poly Logarithmic

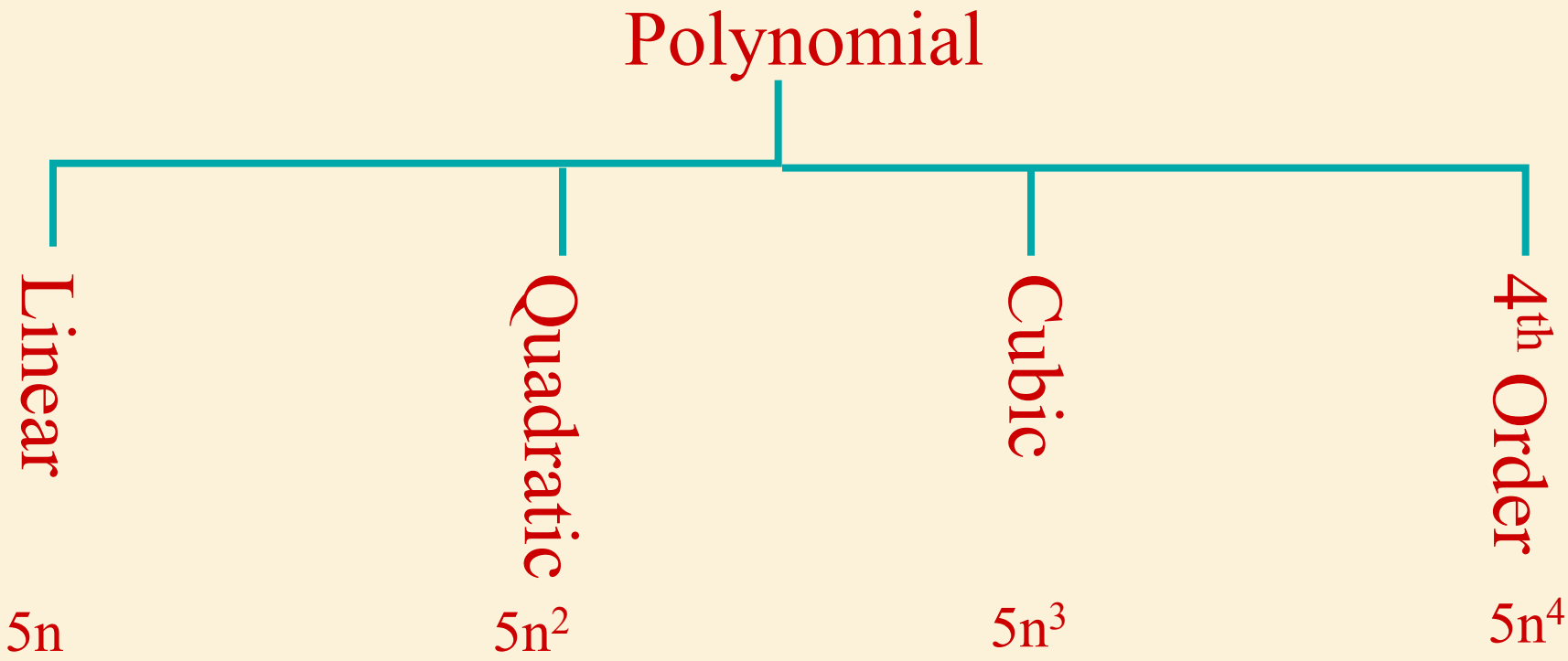
$$(\log n)^5 = \log^5 n$$

(Poly)Logarithmic << Polynomial

$$\log^{1000} n << n^{0.001}$$

For sufficiently large n

Classifying Functions

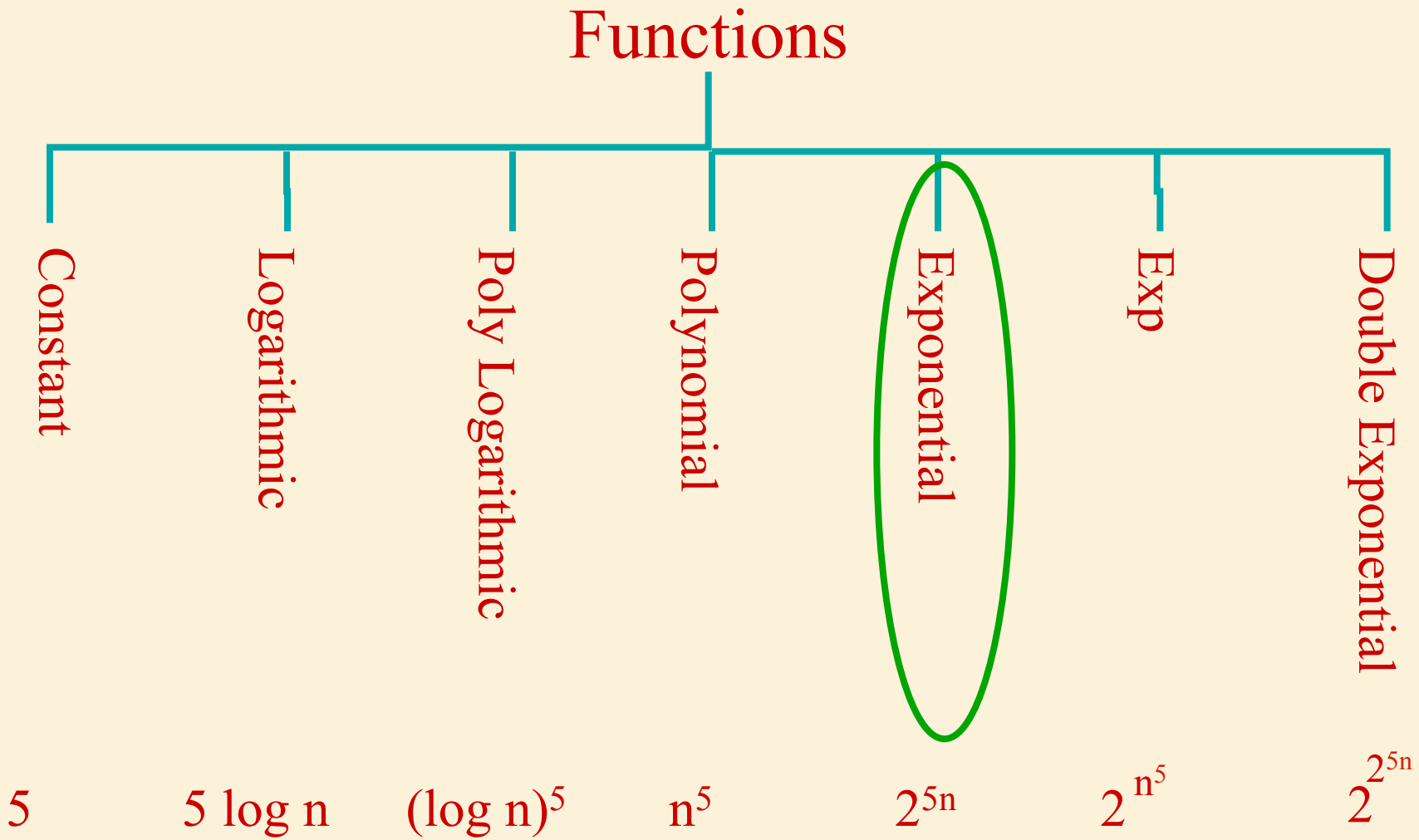


Linear << Quadratic

$$10000\ n \ll 0.0001\ n^2$$

For sufficiently large n

Classifying Functions

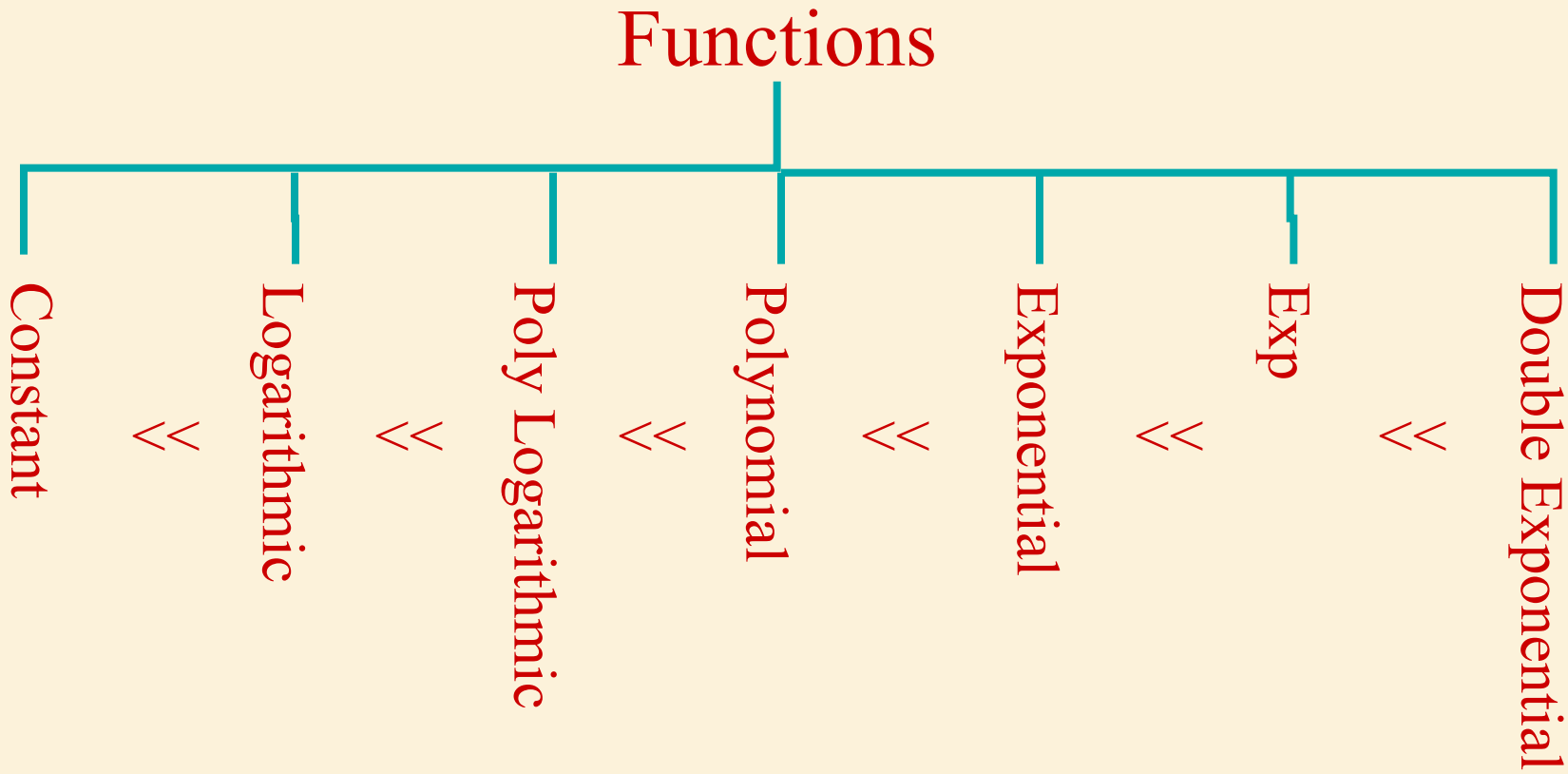


Polynomial << Exponential

$$n^{1000} << 2^{0.001\,n}$$

For sufficiently large n

Ordering Functions



$5 << 5 \log n << (\log n)^5 << n^5 << 2^{5n} << 2^{n^5} << 2^{2^{5n}}$

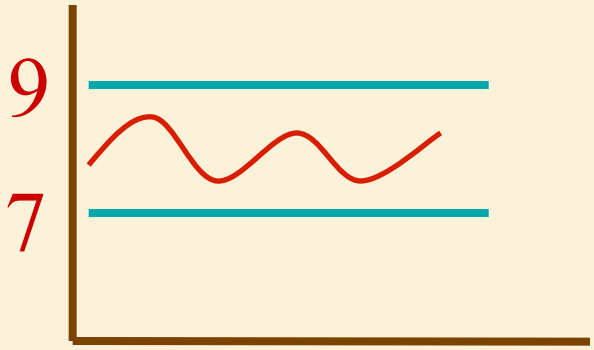
Which Functions are Constant?

Yes	• 5
Yes	• 1,000,000,000,000
Yes	• 0.0000000000001
Yes	• -5
Yes	• 0
No	• 8 + sin(n)

Which Functions are "Constant"?

The running time of the algorithm is a "Constant"
It does not depend significantly
on the size of the input.

- Yes • 5
- Yes • 1,000,000,000,000
- Yes • 0.0000000000001
- No • -5
- No • 0
- Yes • 8 + sin(n)



Lies in between

Which Functions are Quadratic?

- n^2
- … ?

Which Functions are Quadratic?

- n^2
- $0.001\ n^2$
- $1000\ n^2$

Some constant times n^2.

Which Functions are Quadratic?

- n^2
- $0.001\ n^2$
- $1000\ n^2$
- $5n^2 + 3n + 2\log n$

Which Functions are Quadratic?

- n^2
- $0.001\ n^2$
- $1000\ n^2$
- $5n^2 + 3n + 2\log n$

Lies in between

Which Functions are Quadratic?

- n^2
- $0.001\ n^2$
- $1000\ n^2$
- $5n^2 + 3n + 2\log n$

Ignore low-order terms
Ignore multiplicative constants.
Ignore "small" values of n.
Write $\theta(n^2)$.

Which Functions are Polynomial?

- n^5
- … ?

Which Functions are Polynomial?

- n^c
- $n^{0.0001}$
- n^{10000}

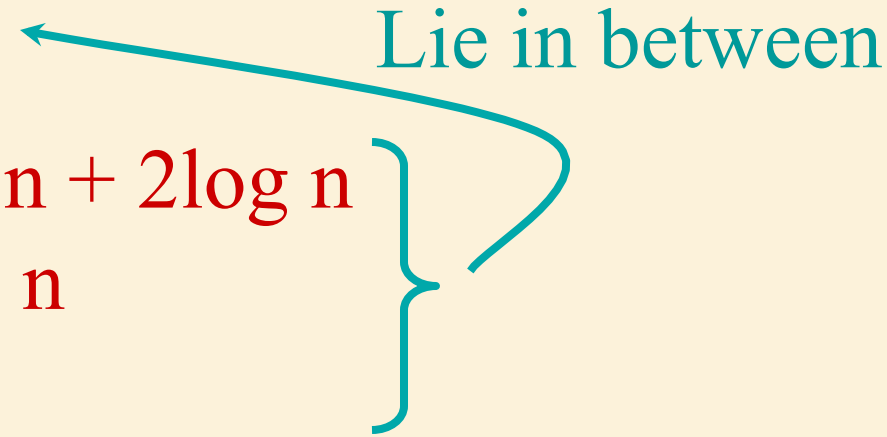
n to some constant power.

Which Functions are Polynomial?

- n^c
- $n^{0.0001}$
- n^{10000}
- $5n^2 + 8n + 2\log n$
- $5n^2 \log n$
- $5n^{2.5}$

Which Functions are Polynomial?

- n^c
- $n^{0.0001}$
- n^{10000}
- $5n^2 + 8n + 2\log n$
- $5n^2 \log n$
- $5n^{2.5}$



Which Functions are Polynomials?

- n^c
- $n^{0.0001}$
- n^{10000}
- $5n^2 + 8n + 2\log n$
- $5n^2 \log n$
- $5n^{2.5}$

Ignore low-order terms

Ignore power constant.

Ignore "small" values of n.

Write $n^{\theta(1)}$

Which Functions are Exponential?

- 2^n
- … ?

Which Functions are Exponential?

- 2^n
- $2^{0.0001\ n}$
- $2^{10000\ n}$

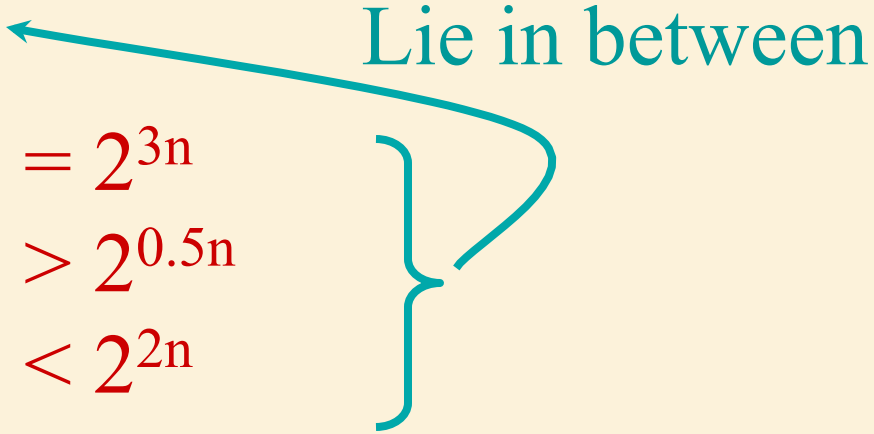
2 raised to a linear function of n.

Which Functions are Exponential?

- 2^n
- $2^{0.0001\ n}$
- $2^{10000\ n}$
- 8^n
- $2^n / n^{100}$ too small?
- $2^n \cdot n^{100}$ too big?

Which Functions are Exponential?

- 2^n
- $2^{0.0001\ n}$
- $2^{10000\ n}$
- $8^n = 2^{3n}$
- $2^n / n^{100} > 2^{0.5n}$
- $2^n \cdot n^{100} < 2^{2n}$



Which Functions are Exponential?

- 2^n
- $2^{0.0001\,n}$
- $2^{10000\,n}$
- $8^n = 2^{3n}$
- $2^n / n^{100} > 2^{0.5n}$
- $2^n \cdot n^{100} < 2^{2n}$

$$2^{0.5n} > n^{100}$$

$$2^n = 2^{0.5n} \cdot 2^{0.5n} > n^{100} \cdot 2^{0.5n}$$

$$2^n / n^{100} > 2^{0.5n}$$

Which Functions are Exponential?

- 2^n
- $2^{0.0001\ n}$
- $2^{10000\ n}$
- 8^n
- $2^n / n^{100}$
- $2^n \cdot n^{100}$

- Ignore low-order terms
- Ignore base.
- Ignore "small" values of n.
- Ignore polynomial factors.
- Write $2^{\theta(n)}$

Classifying Functions

$f(n) = 8 \cdot 2^{4n} / n^{100} + 5 \cdot n^3$

Tell me the most significant thing about your function

Rank constants in significance.

Classifying Functions

$f(n) = 8 \cdot 2^{4n} / n^{100} + 5 \cdot n^3$

Tell me the most significant thing about your function

$2^{\theta(n)}$

because $2^{3n} < f(n) < 2^{4n}$

Classifying Functions

$f(n) = 8 \cdot 2^{4n} / n^{100} + 5 \cdot n^3$

Tell me the most significant thing about your function

$2^{\theta(n)}$

$2^{4n}/n^{\theta(1)}$

$\theta(2^{4n}/n^{100})$

$8 \cdot 2^{4n} / n^{100} + n^{\theta(1)}$

$8 \cdot 2^{4n} / n^{100} + \theta(n^3)$

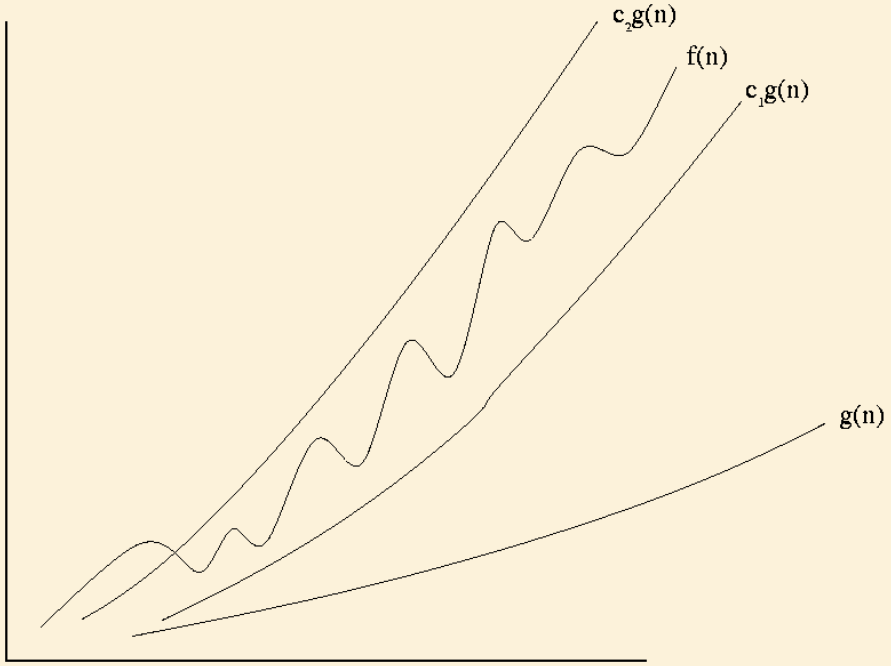
$8 \cdot 2^{4n} / n^{100} + 5 \cdot n^3$

Notations

Theta	$f(n) = \theta(g(n))$	$f(n) \approx c\ g(n)$
Big Oh	$f(n) = O(g(n))$	$f(n) \leq c\ g(n)$
Big Omega	$f(n) = \Omega(g(n))$	$f(n) \geq c\ g(n)$
Little Oh	$f(n) = o(g(n))$	$f(n) << c\ g(n)$
Little Omega	$f(n) = \omega(g(n))$	$f(n) >> c\ g(n)$

Definition of Theta

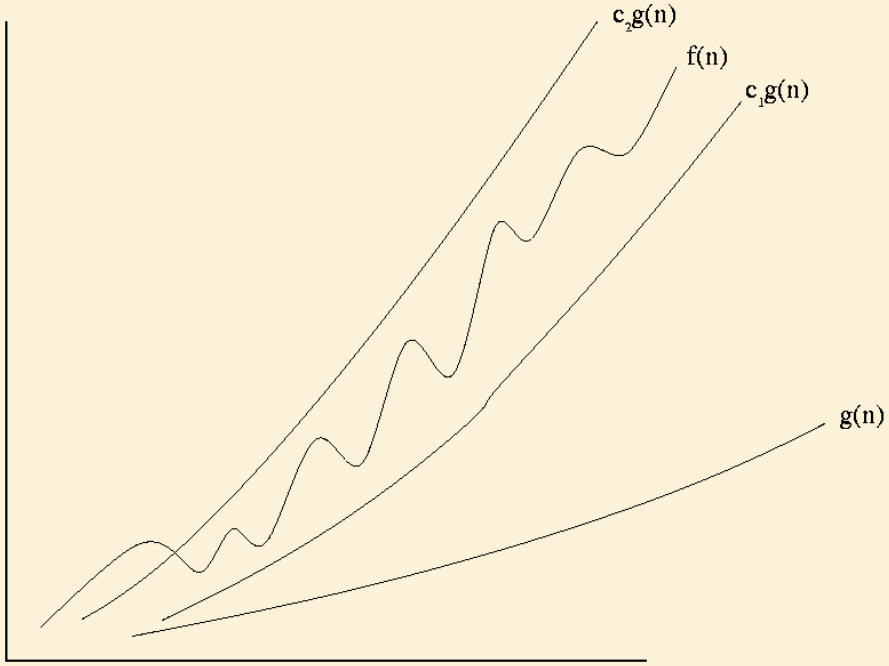
$$f(n) = \theta(g(n))$$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Definition of Theta

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

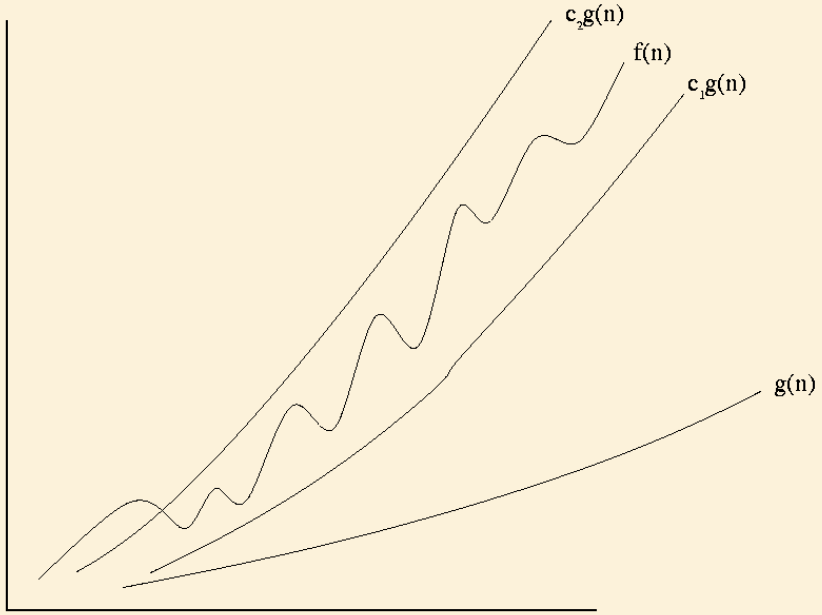


$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

f(n) is sandwiched between $c_1 g(n)$ and $c_2 g(n)$

Definition of Theta

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



$$\exists \underbrace{c_1, c_2}, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

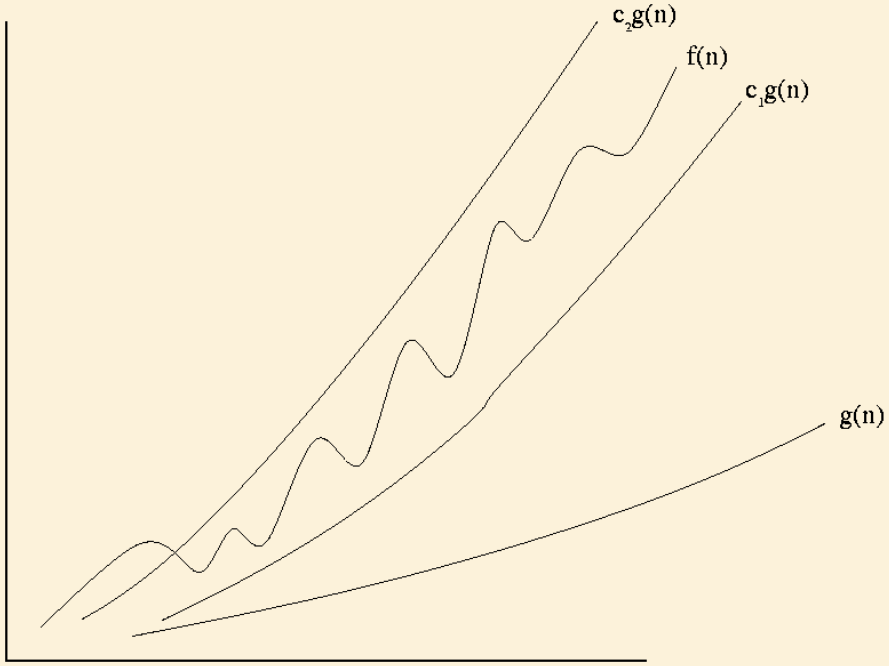
f(n) is sandwiched between $c_1 g(n)$ and $c_2 g(n)$

for some sufficiently small c_1 (= 0.0001)

for some sufficiently large c_2 (= 1000)

Definition of Theta

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

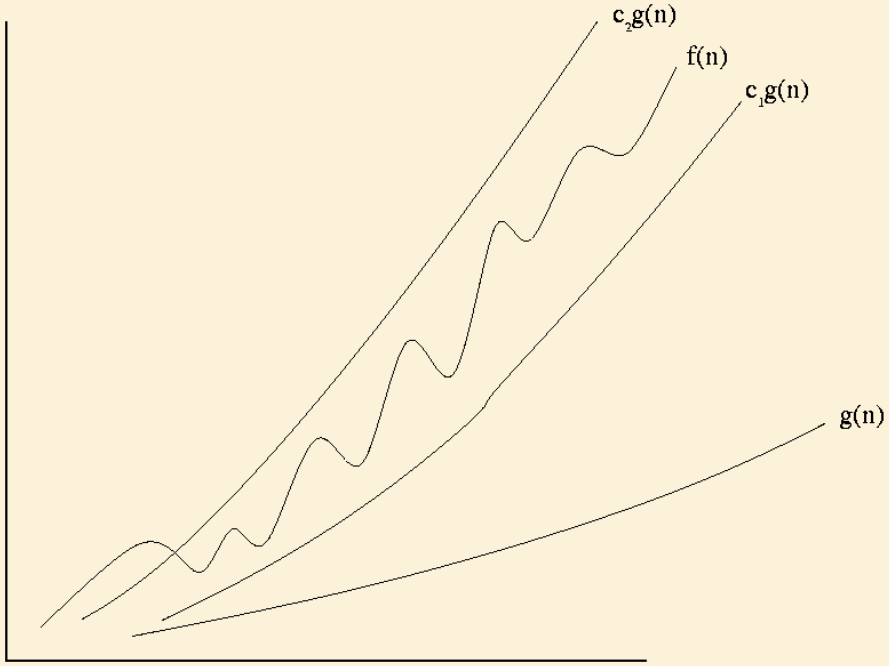


$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

For all sufficiently large n

Definition of Theta

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



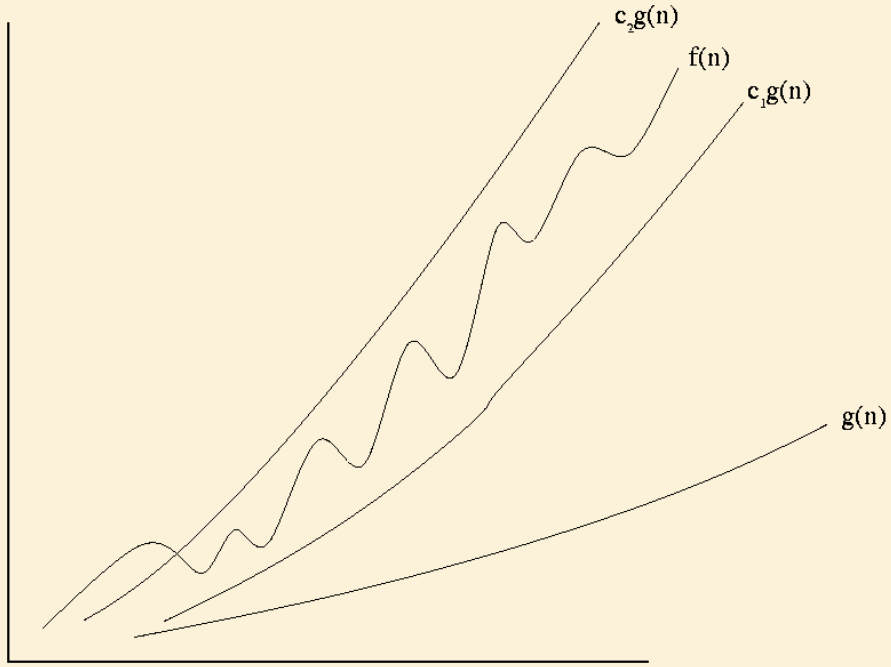
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

For all sufficiently large n

For some definition of "sufficiently large"

Definition of Theta

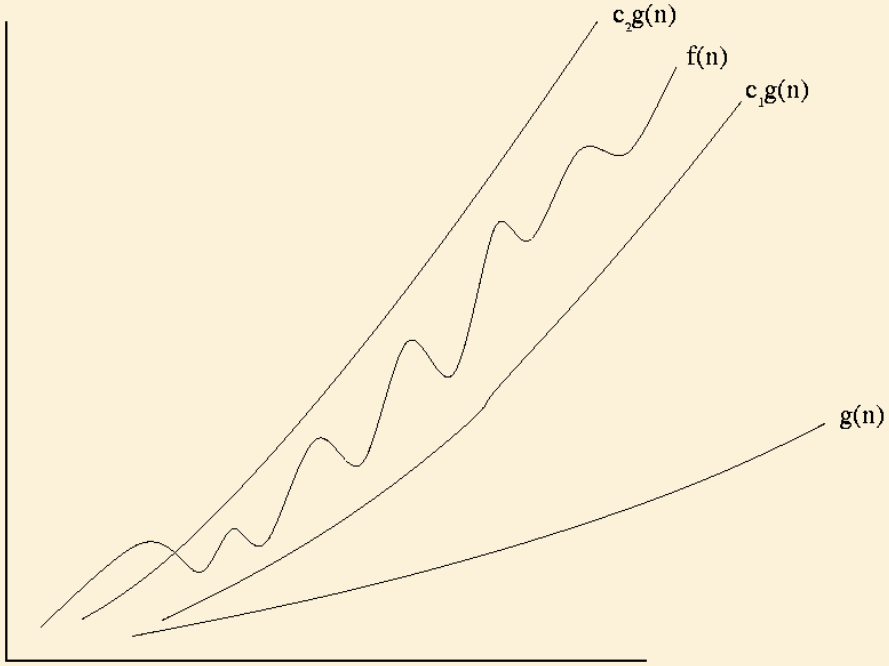
$3n^2 + 7n + 8 = \theta(n^2)$



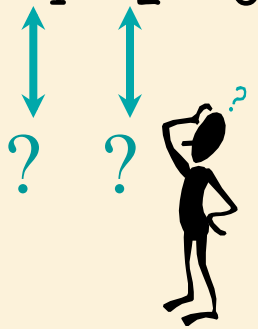
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



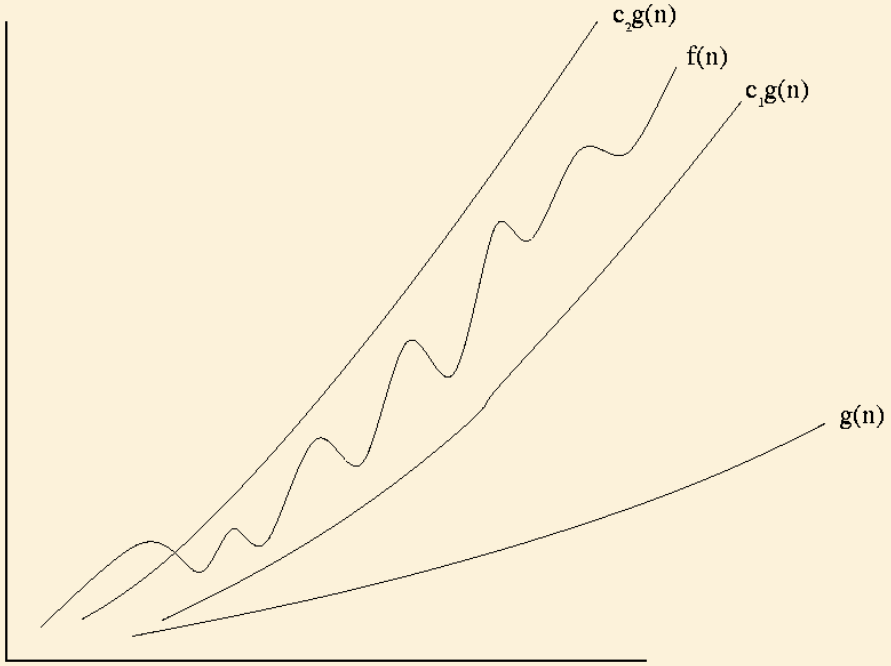
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$



$$c_1 \cdot n^2 \leq 3n^2 + 7n + 8 \leq c_2 \cdot n^2$$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



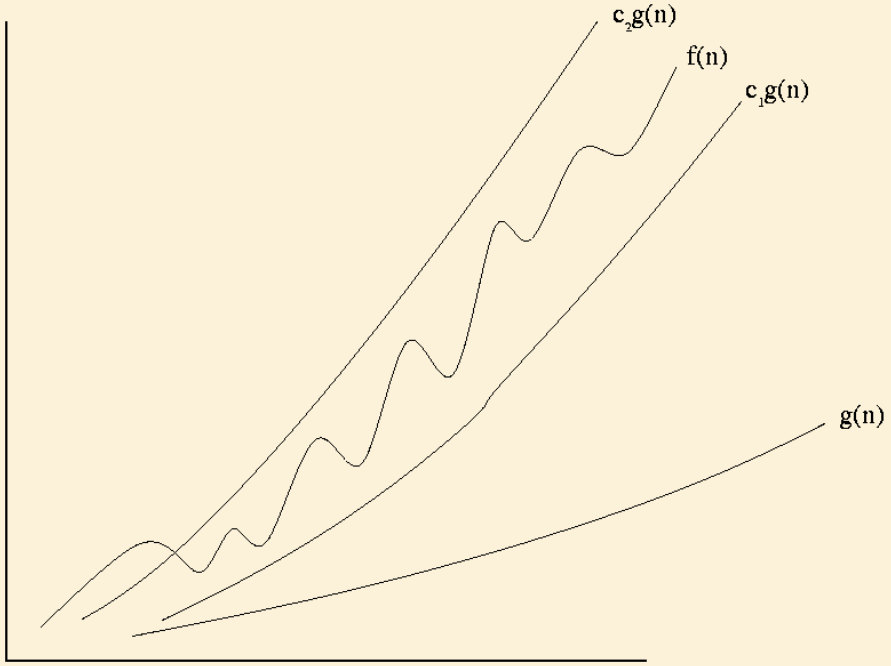
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$c_1 = 3 \quad c_2 = 4$

$3 \cdot n^2 \leq 3n^2 + 7n + 8 \leq 4 \cdot n^2$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

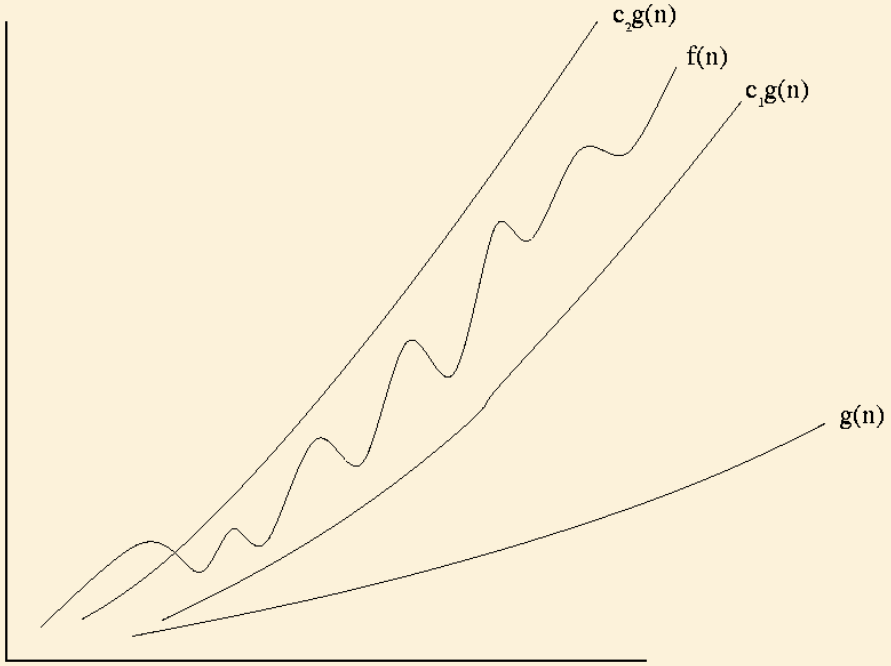
1

$$3 \cdot 1^2 \leq 3 \cdot 1^2 + 7 \cdot 1 + 8 \leq 4 \cdot 1^2$$

False

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

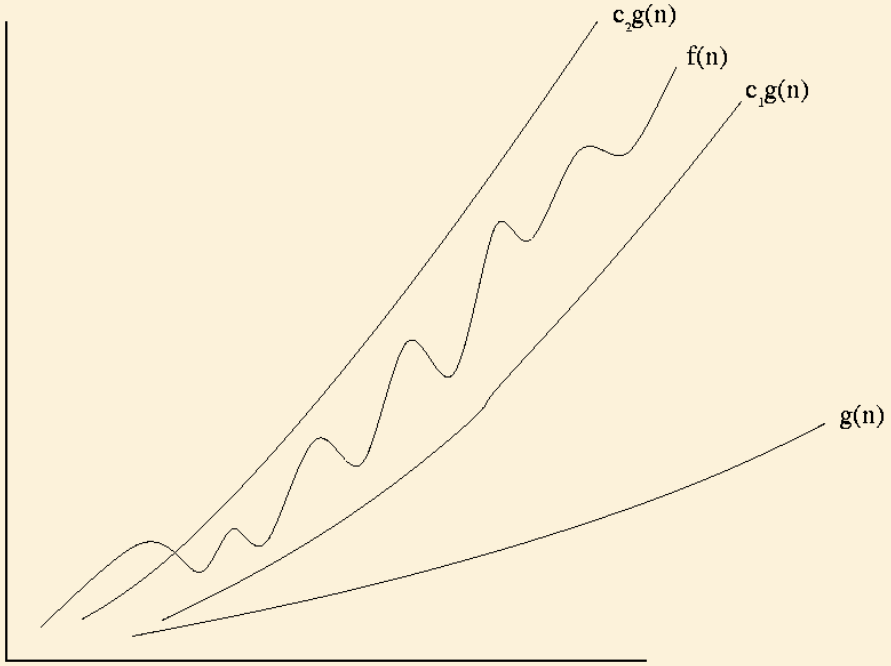
7

$3 \cdot 7^2 \leq 3 \cdot 7^2 + 7 \cdot 7 + 8 \leq 4 \cdot 7^2$

False

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

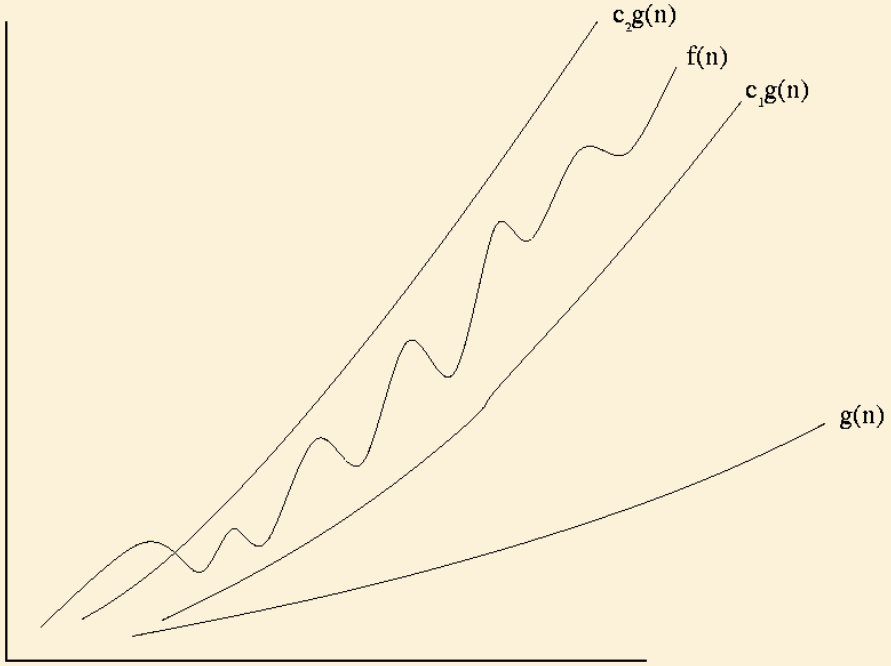
$n_0 = 8$

$$3 \cdot 8^2 \leq 3 \cdot 8^2 + 7 \cdot 8 + 8 \leq 4 \cdot 8^2$$

True

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

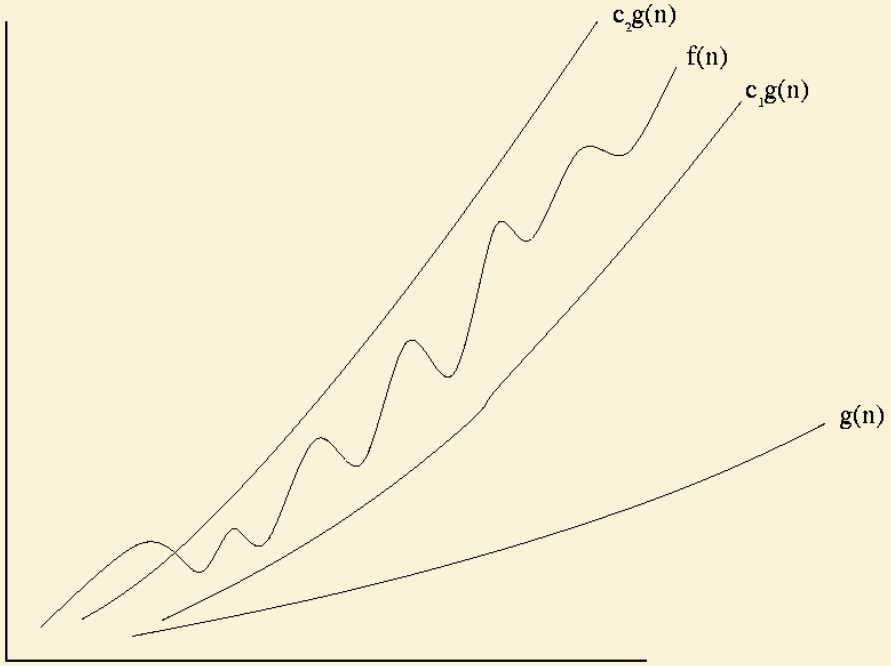
9

$3 \cdot 9^2 \leq 3 \cdot 9^2 + 7 \cdot 9 + 8 \leq 4 \cdot 9^2$

True

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

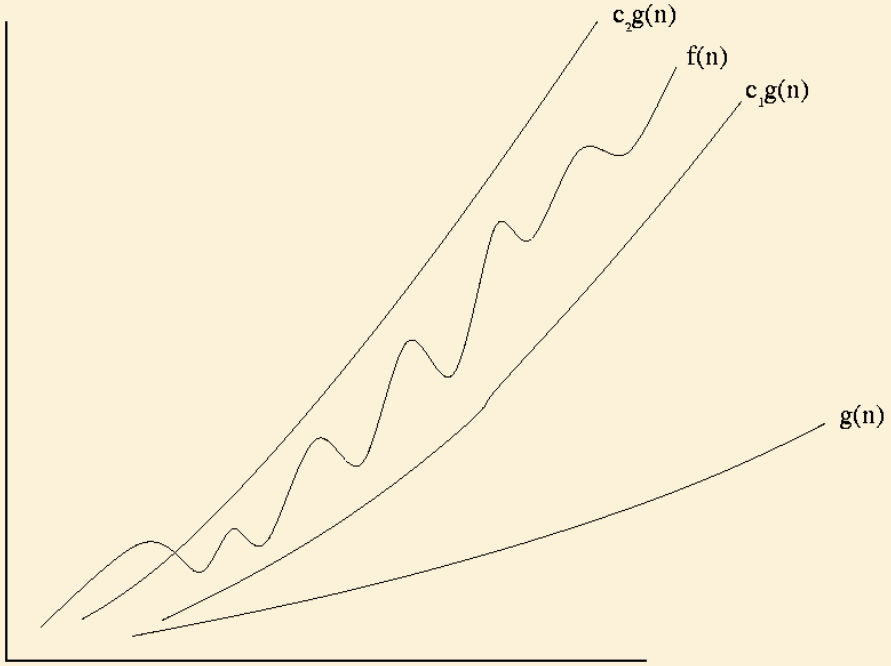
$n_0 = 10$

$3 \cdot 10^2 \leq 3 \cdot 10^2 + 7 \cdot 10 + 8 \leq 4 \cdot 10^2$

True

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$

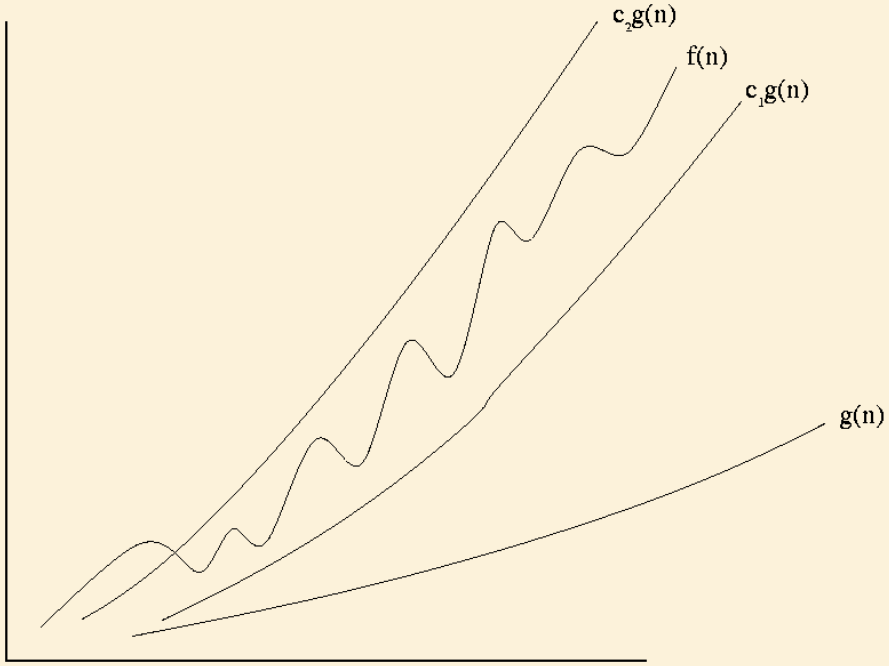


$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$3 \cdot n^2 \leq 3n^2 + 7n + 8 \leq 4 \cdot n^2$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$

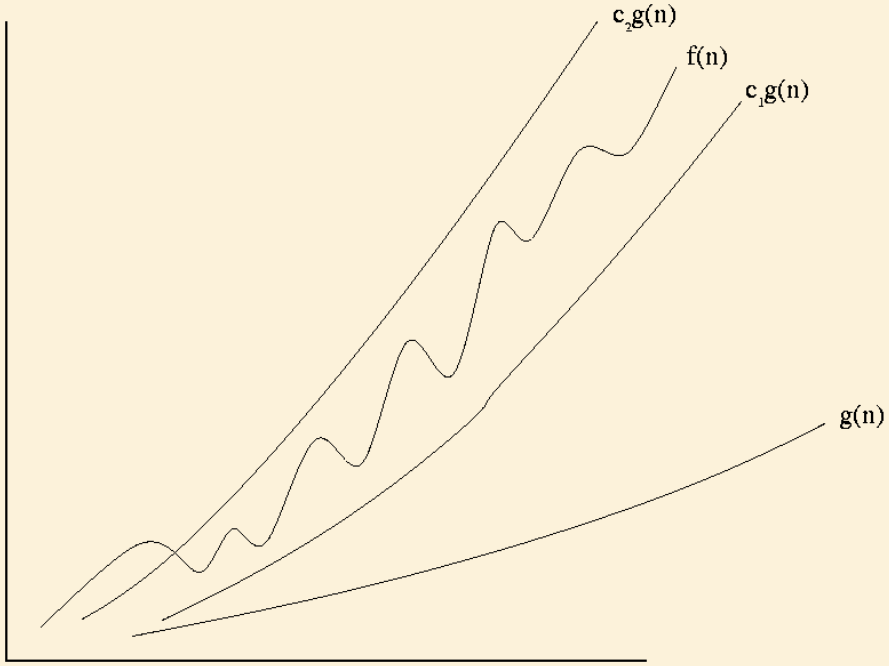
8

$n \geq 8$

$3 \cdot n^2 \leq 3n^2 + 7n + 8 \leq 4 \cdot n^2$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

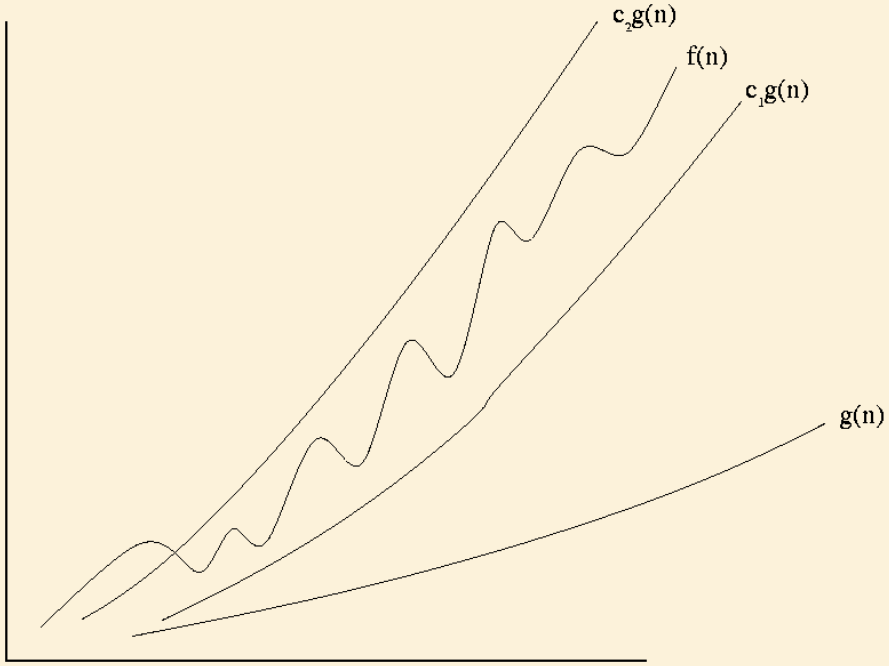
$n \geq 8$

$3 \cdot n^2 \leq 3n^2 + 7n + 8 \leq 4 \cdot n^2$

True

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$3 \cdot n^2 \leq 3n^2 + 7n + 8 \leq 4 \cdot n^2$$

$n \geq 8$

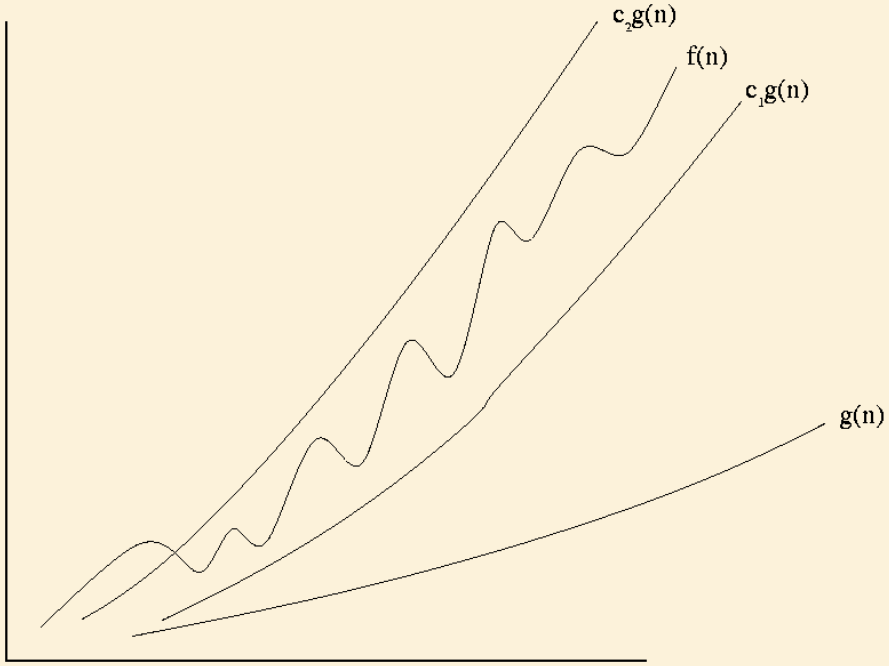
$7n + 8 \leq 1 \cdot n^2$

$7 + {}^8/_n \leq 1 \cdot n$

True

Definition of Theta

$3n^2 + 7n + 8 = \theta(n^2)$



True

$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$

3 4 8

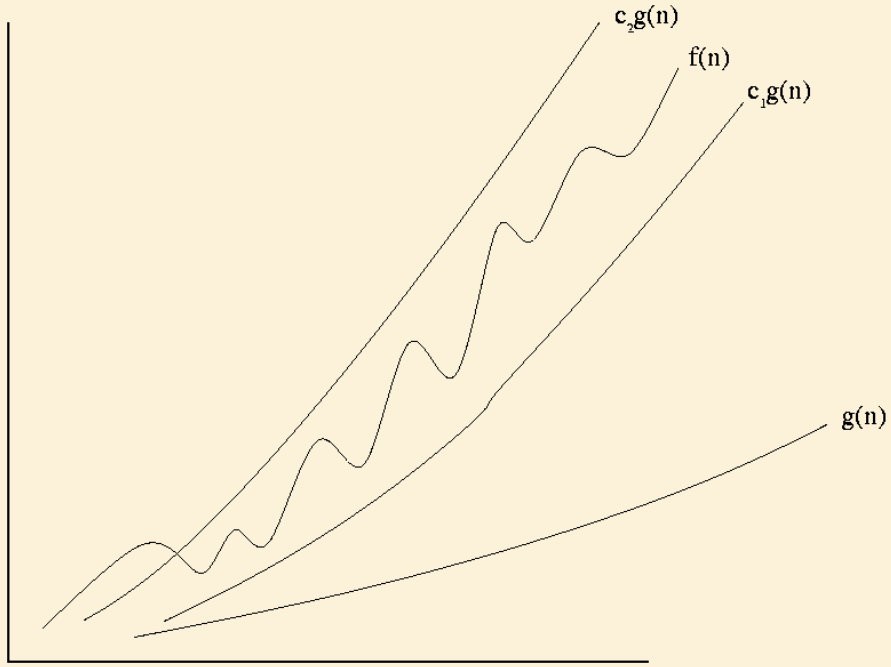
$n \geq 8 \quad 3 \cdot n^2 \leq 3n^2 + 7n + 8 \leq 4 \cdot n^2$

Asymptotic Notation

Theta	$f(n) = \theta(g(n))$	$f(n) \approx c\ g(n)$
BigOh	$f(n) = O(g(n))$	$f(n) \leq c\ g(n)$
Omega	$f(n) = \Omega(g(n))$	$f(n) \geq c\ g(n)$
Little Oh	$f(n) = o(g(n))$	$f(n) << c\ g(n)$
Little Omega	$f(n) = \omega(g(n))$	$f(n) >> c\ g(n)$

Definition of Theta

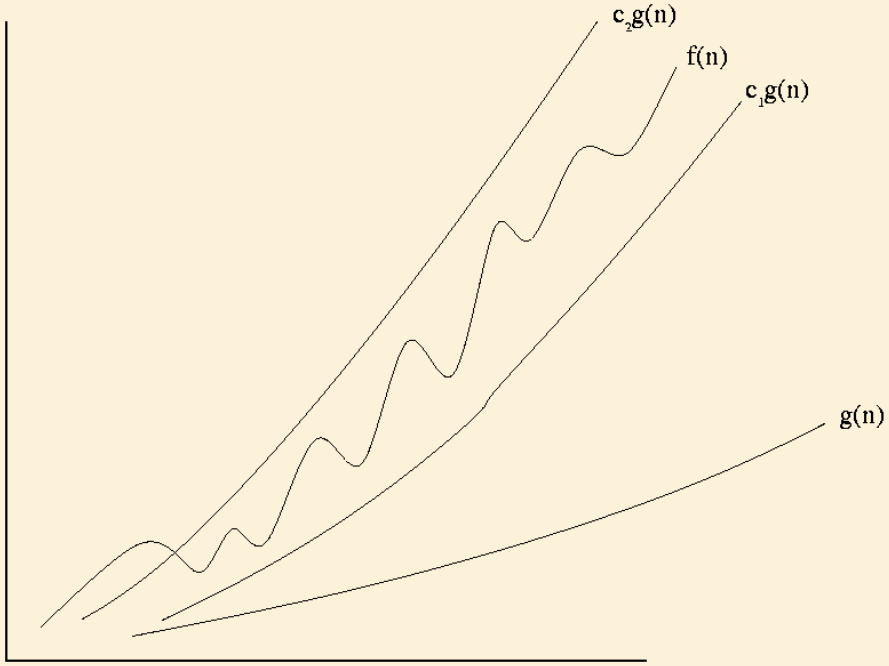
$3n^2 + 7n + 8 = \theta(n)$



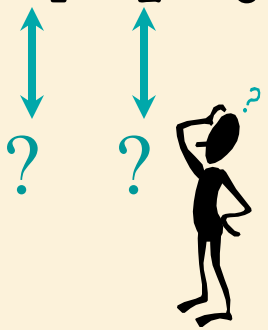
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n)$



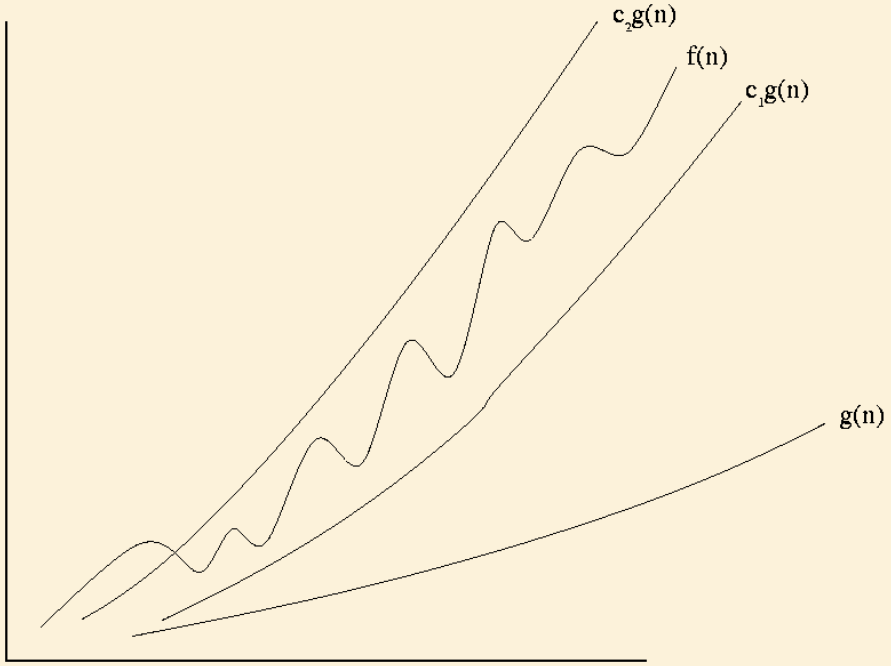
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$



$$c_1 \cdot n \leq 3n^2 + 7n + 8 \leq c_2 \cdot n$$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n)$



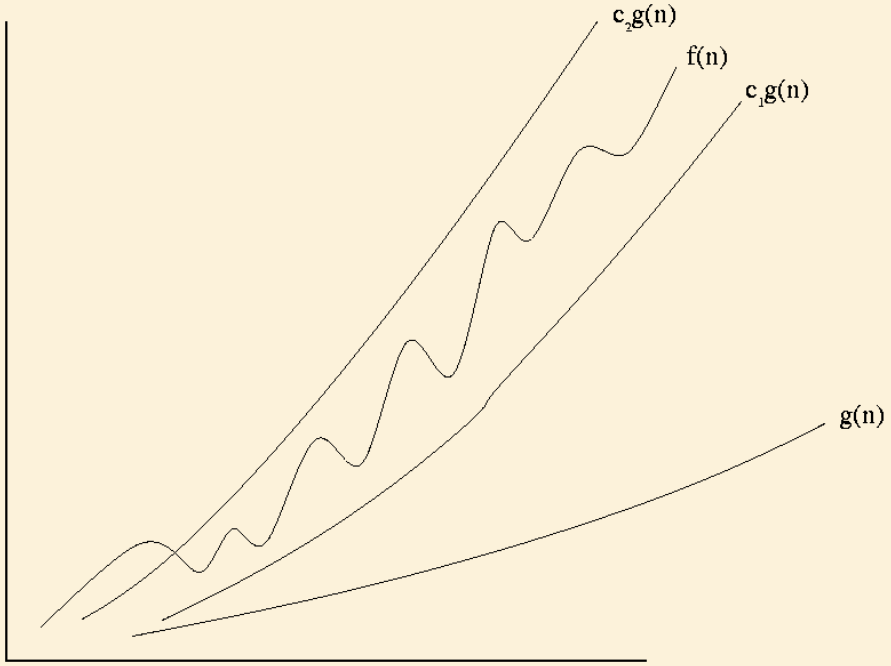
$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

c_1 = 3, c_2 = 100

$3 \cdot n \leq 3n^2 + 7n + 8 \leq 100 \cdot n$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n)$

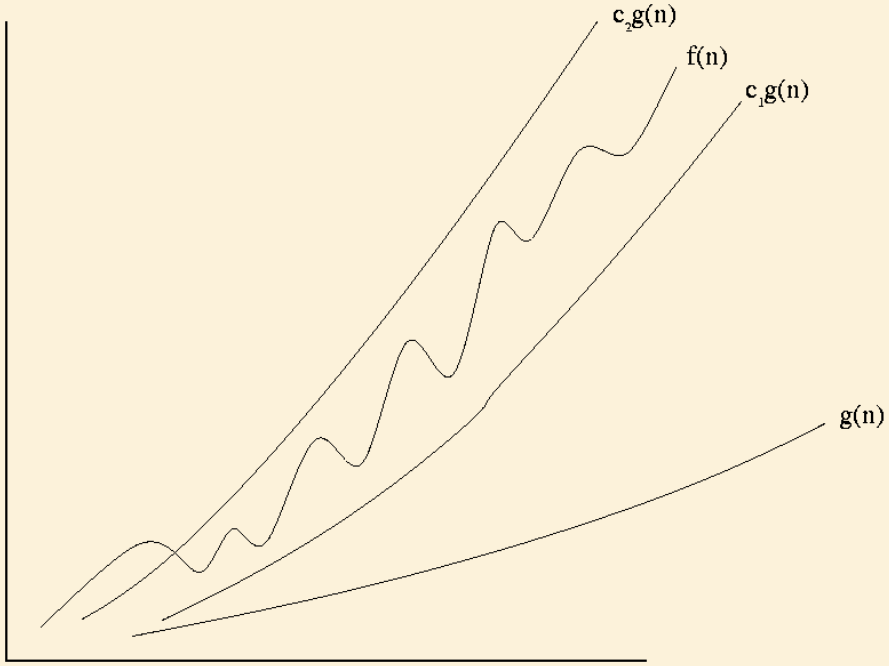


$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$3 \cdot n \leq 3n^2 + 7n + 8 \leq 100 \cdot n$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

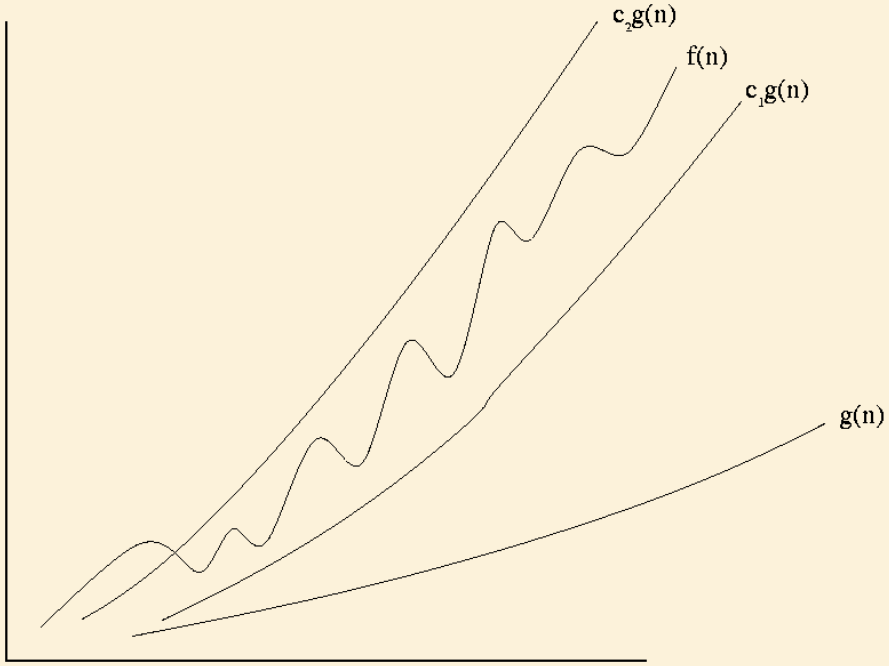
$n_0 = 100$

$3 \cdot 100 \leq 3 \cdot 100^2 + 7 \cdot 100 + 8 \leq 100 \cdot 100$

False

Definition of Theta

$3n^2 + 7n + 8 = \theta(n)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

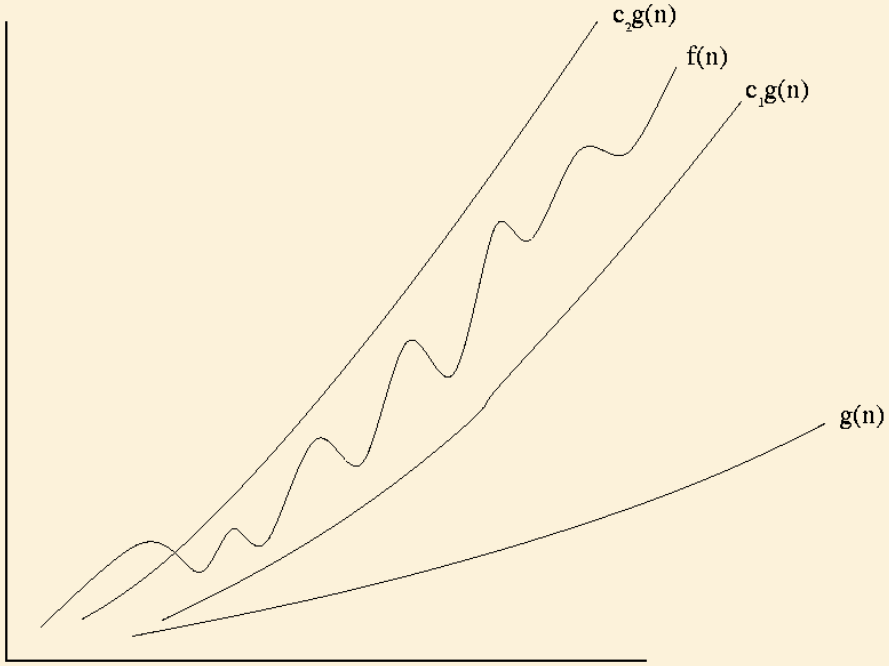
c_1 c_2

3 10,000

$3 \cdot n \leq 3n^2 + 7n + 8 \leq 10{,}000 \cdot n$

Definition of Theta

$3n^2 + 7n + 8 = \theta(n)$



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

10,000

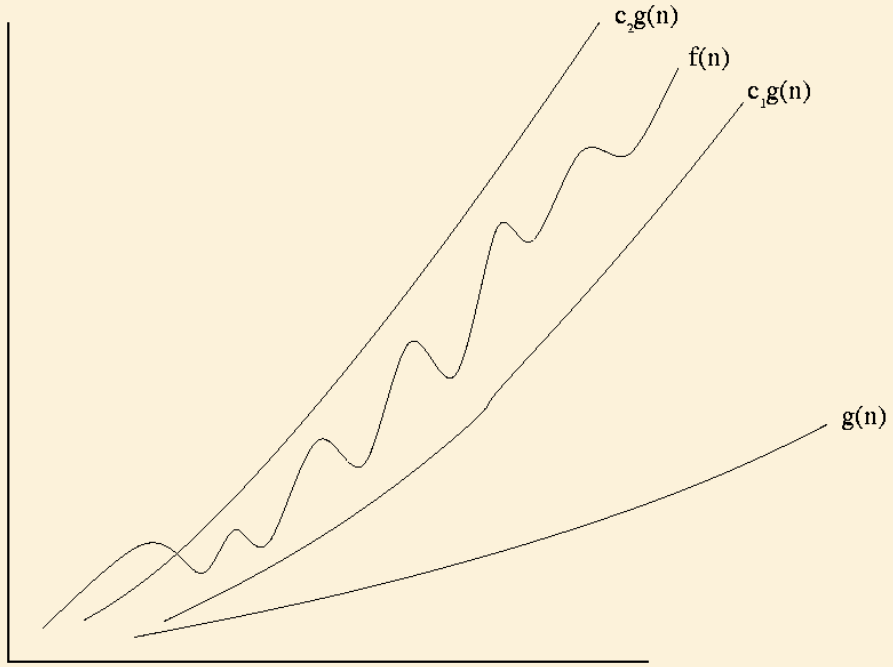
$3 \cdot 10{,}000 \leq 3 \cdot 10{,}000^2 + 7 \cdot 10{,}000 + 8 \leq 10{,}000 \cdot 10{,}000$

False

Definition of Theta

$3n^2 + 7n + 8 \neq \theta(n)$

What is the
reverse statement?



$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

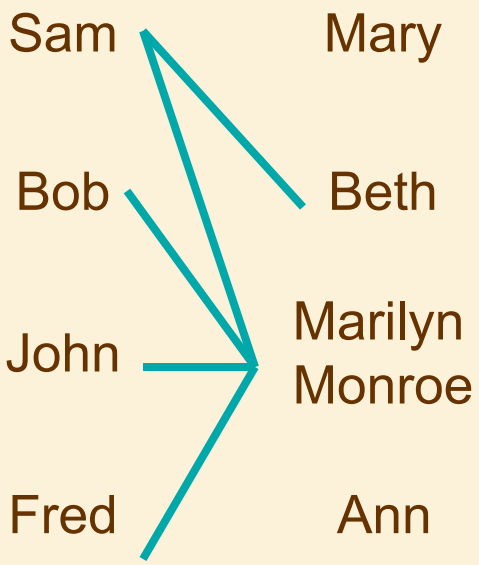
Understand Quantifiers!!!

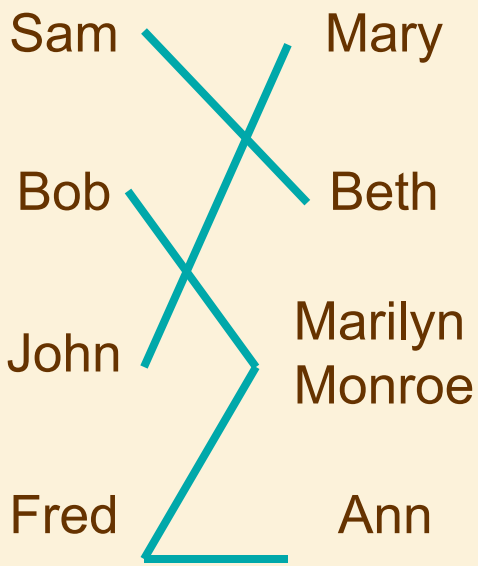
$\forall$ b, Loves(b, MM)

$\neg[\exists$ b, $\neg$Loves(b, MM)$]$

$\neg[\forall$ b, Loves(b, MM)$]$

$\exists$ b, $\neg$Loves(b, MM)

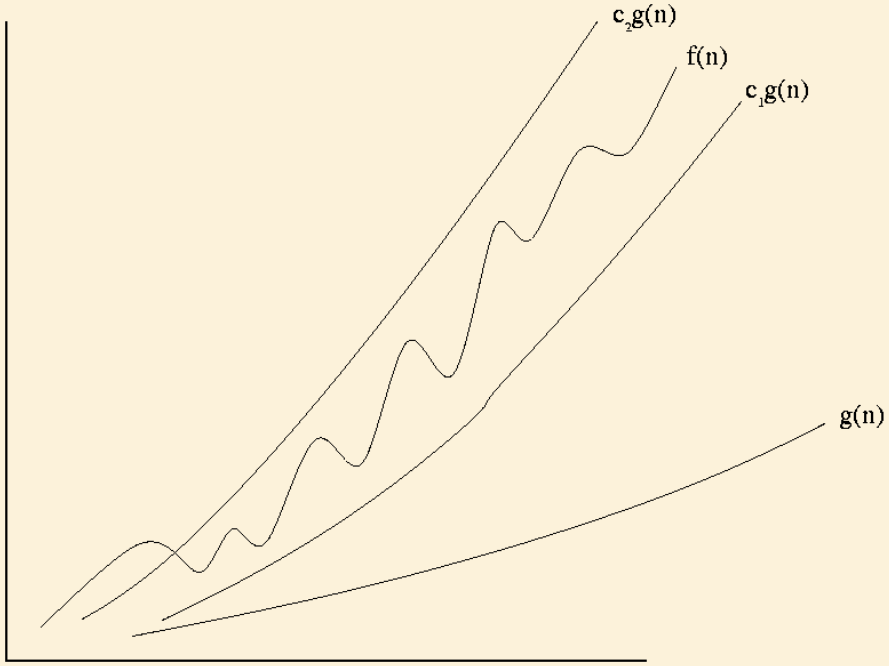




Definition of Theta

$3n^2 + 7n + 8 \neq \theta(n)$

The reverse statement

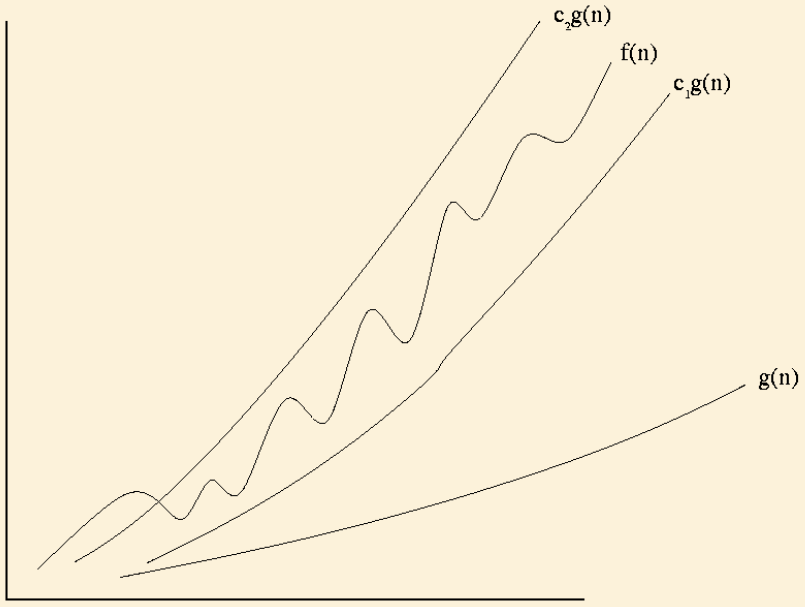


$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$\forall c_1, c_2, n_0 > 0 : \exists n \geq n_0 : f(n) < c_1 g(n) \text{ or } f(n) > c_2 g(n)$$

Definition of Theta

$3n^2 + 7n + 8 \neq \theta(n)$



$$\forall c_1, c_2, n_0 > 0 : \exists n \geq n_0 : f(n) < c_1 g(n) \text{ or } f(n) > c_2 g(n)$$

$$3n^2 + 7n + 8 > c_2 n$$

$$\leftrightarrow 3 + \frac{7}{n} + \frac{8}{n^2} > \frac{c_2}{n}$$

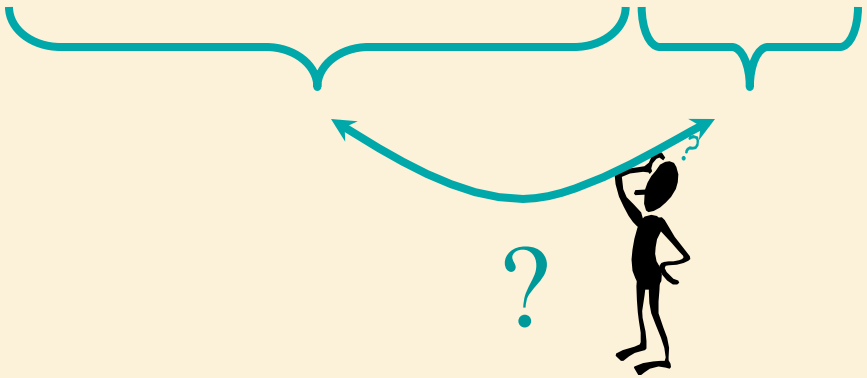
Satisfied for $n = \max(n_0, c_2)$

Order of Quantifiers

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

$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

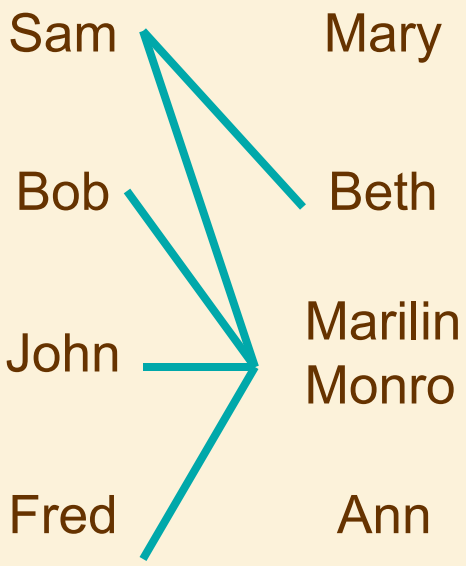
$$\exists n_0 > 0 : \forall n \geq n_0, \exists c_1, c_2 : c_1 g(n) \leq f(n) \leq c_2 g(n)$$



Understand Quantifiers!!!

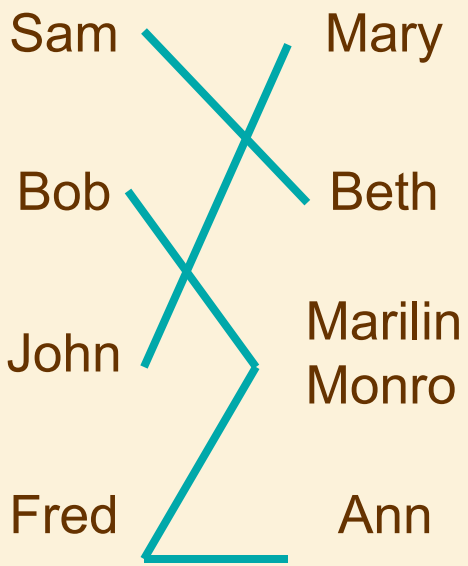
$$\exists g, \forall b, loves(b, g)$$

One girl



$$\forall b, \exists g, loves(b, g)$$

Could be a separate girl for each boy.

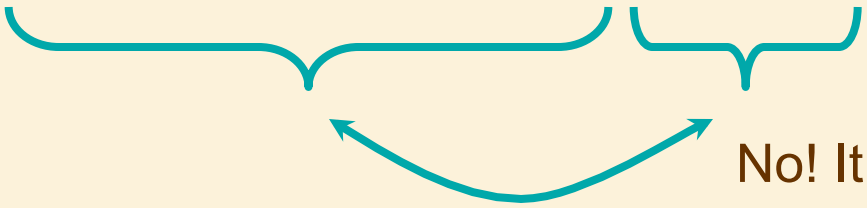


Order of Quantifiers

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

$$\exists c_1, c_2, n_0 > 0 : \forall n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$\exists n_0 > 0 : \forall n \geq n_0, \exists c_1, c_2 : c_1 g(n) \leq f(n) \leq c_2 g(n)$$



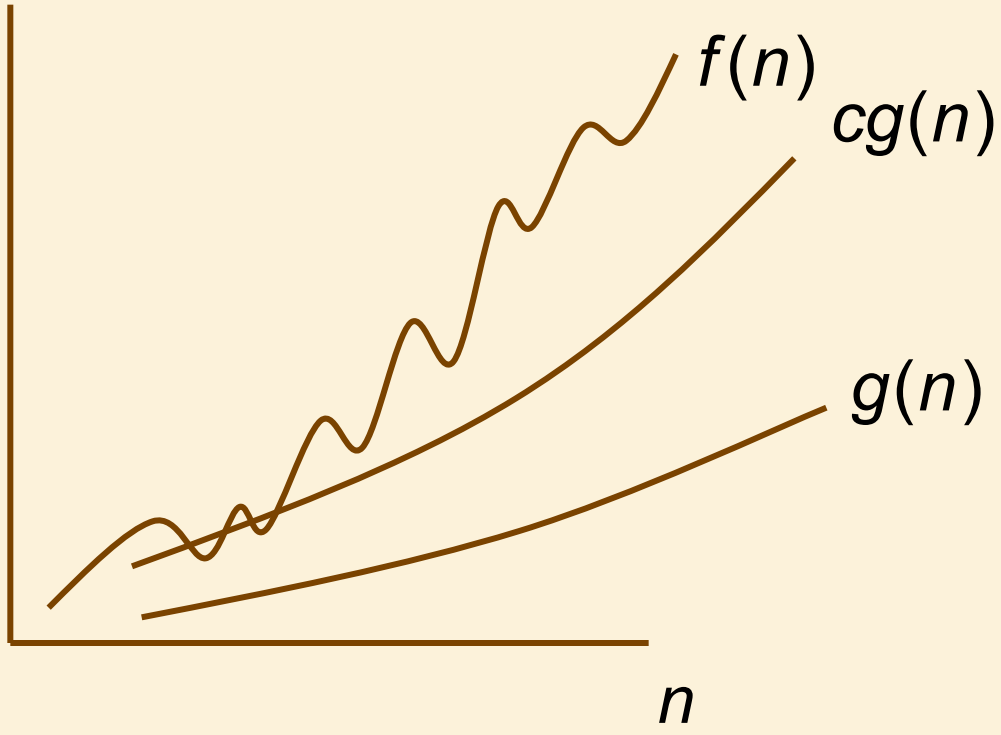
No! It cannot be a different c_1 and c_2 for each n.

Other Notations

Theta	$f(n) = \theta(g(n))$	$f(n) \approx c\ g(n)$
BigOh	$f(n) = O(g(n))$	$f(n) \leq c\ g(n)$
Omega	$f(n) = \Omega(g(n))$	$f(n) \geq c\ g(n)$
Little Oh	$f(n) = o(g(n))$	$f(n) << c\ g(n)$
Little Omega	$f(n) = \omega(g(n))$	$f(n) >> c\ g(n)$

Definition of "Big Omega"

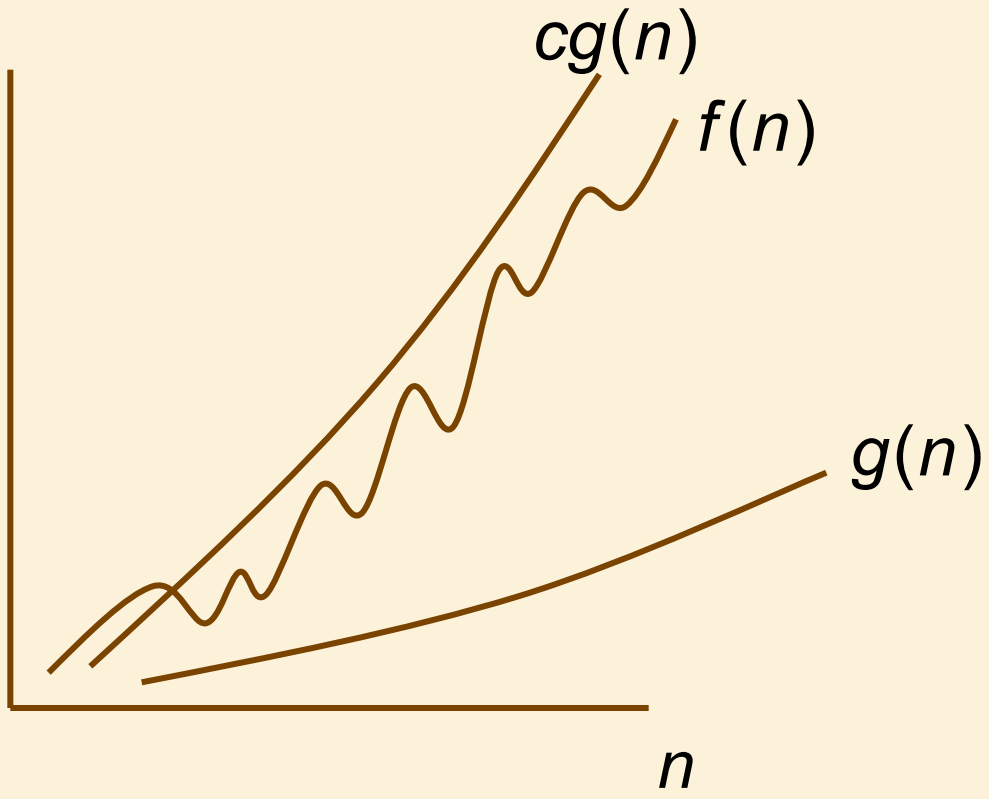
$f(n) \in \Omega(g(n))$



$$\exists c, n_0 > 0 : \forall n \geq n_0, f(n) \geq cg(n)$$

Definition of "Big Oh"

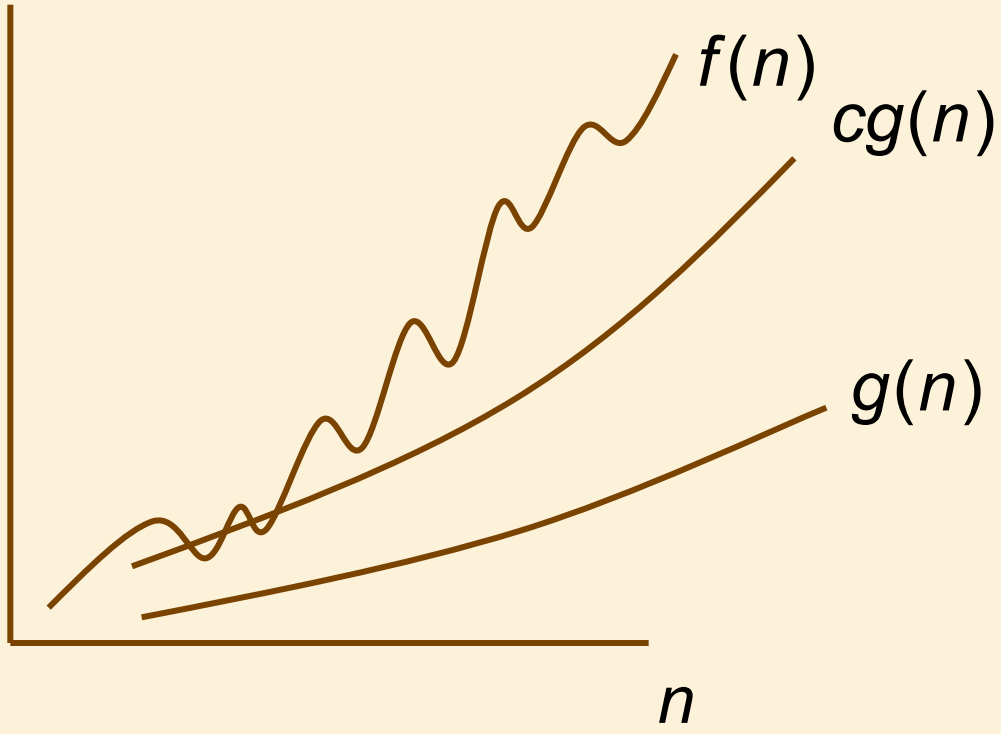
$f(n) \in O(g(n))$



$$\exists c, n_0 > 0 : \forall n \geq n_0, f(n) \leq cg(n)$$

Definition of "Little Omega"

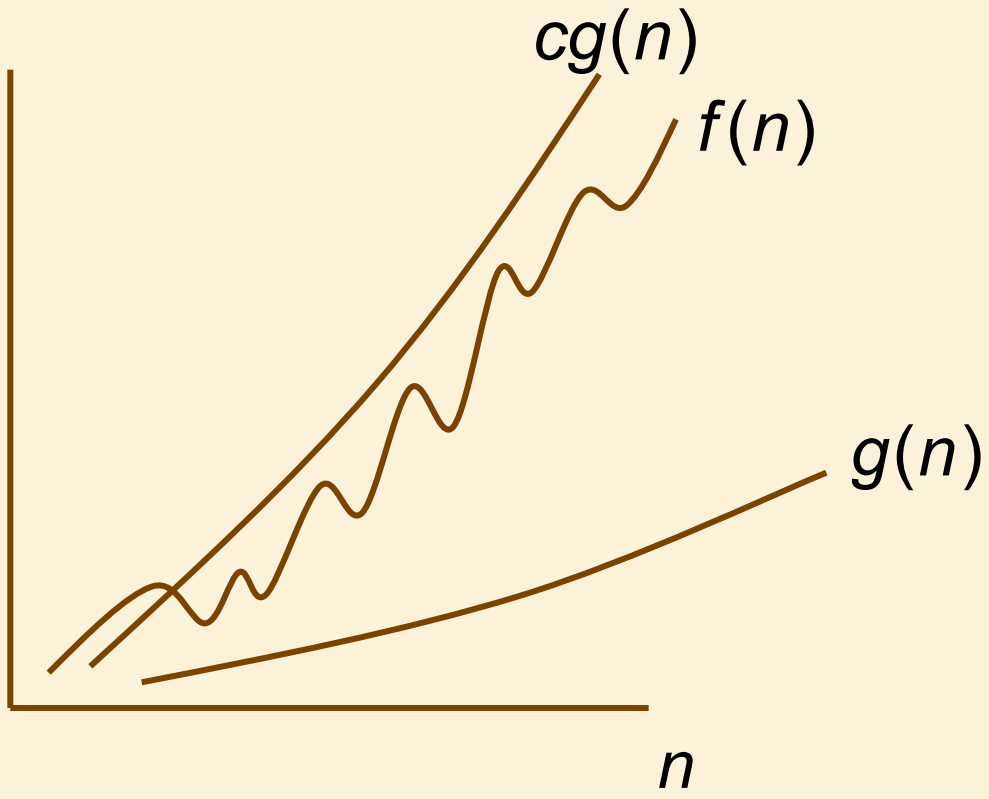
$$f(n) \in \omega(g(n))$$



$$\forall c > 0, \exists n_0 > 0 : \forall n \geq n_0, f(n) > cg(n)$$

Definition of "Little Oh"

$f(n) \in o(g(n))$



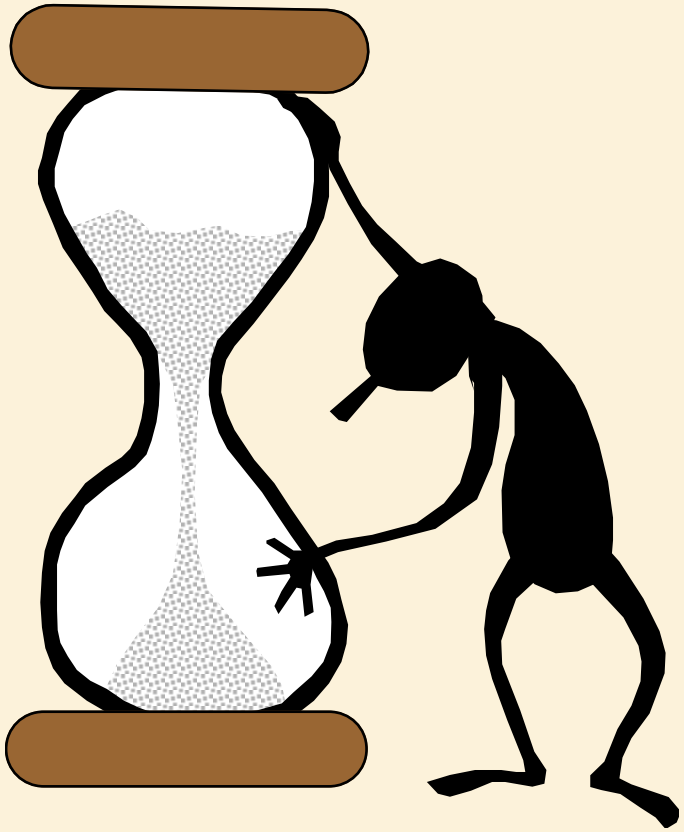
$$\forall c > 0, \exists n_0 > 0 : \forall n \geq n_0, f(n) < cg(n)$$

Time Complexity Is a Function

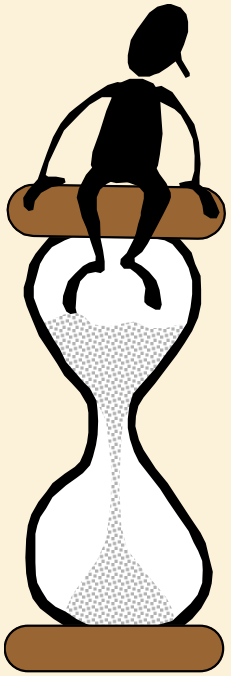
- Specifies how the running time depends on the size of the input.

- A function mapping “size” of input → “time” T(n) executed .

Definition of Time?



Definition of Time



- # of seconds (machine dependent).
- # lines of code executed.
- # of times a specific operation is performed (e.g., addition).

- These are all reasonable definitions of time, because they are within a constant factor of each other.

Definition of Input Size

- A good definition: #bits
- Examples:

1. Sorting an array A of k-bit integers.

   e.g., k=16.

   Input size $n = k \times \text{length}(A)$

   Note that, since $n \propto \text{length}(A)$, sufficient to define $n = \text{length}(A)$.

2. Multiplying $A, B \in \bullet$.

   Let n_A = #bits used to represent A

   Let n_B = #bits used to represent B

   Then input size $n = n_A + n_B$

   Note that $n_A \; ; \; \log_2 A$ and $n_B \; ; \; \log_2 B$

   Thus input size $n \; ; \; \log_2 A + \log_2 B$

Time Complexity Is a Function

- Specifies how the running time depends on the size of the input.

- A function mapping:
  - “size” of input → “time” T(n) executed .

- Or more precisely:
  - # of bits n needed to represent the input → # of operations T(n) executed .

Time Complexity of an Algorithm

The time complexity of an algorithm is the *largest* time required on *any* input of size n. (Worst case analysis.)

- $O(n^2)$: For any input size $n>n_0$, the algorithm takes no more than cn^2 time on every input.
- $\Omega(n^2)$: For any input size $n>n_0$, the algorithm takes at least cn^2 time on at least one input.
- $\theta(n^2)$: Do both.

What is the height of tallest person in the class?

Bigger than this?

Need to find only one person who is taller

Smaller than this?

Need to look at every person

Time Complexity of a Problem

The time complexity of a problem is the time complexity of the *fastest* algorithm that solves the problem.

- $O(n^2)$: Provide an algorithm that solves the problem in no more than this time.
  - Remember: for every input, i.e. worst case analysis!
- $\Omega(n^2)$: Prove that no algorithm can solve it faster.
  - Remember: only need one input that takes at least this long!
- $\theta(n^2)$: Do both.