ADSII3ILV

Algorithms and Data Structures II

Asymptotic Notations

Dr. Alessio Gambi

IMC Krems

Agenda

- Recap on Complexity
- Asymptotic Notations

Recap: Slow vs. Fast Fibonacci

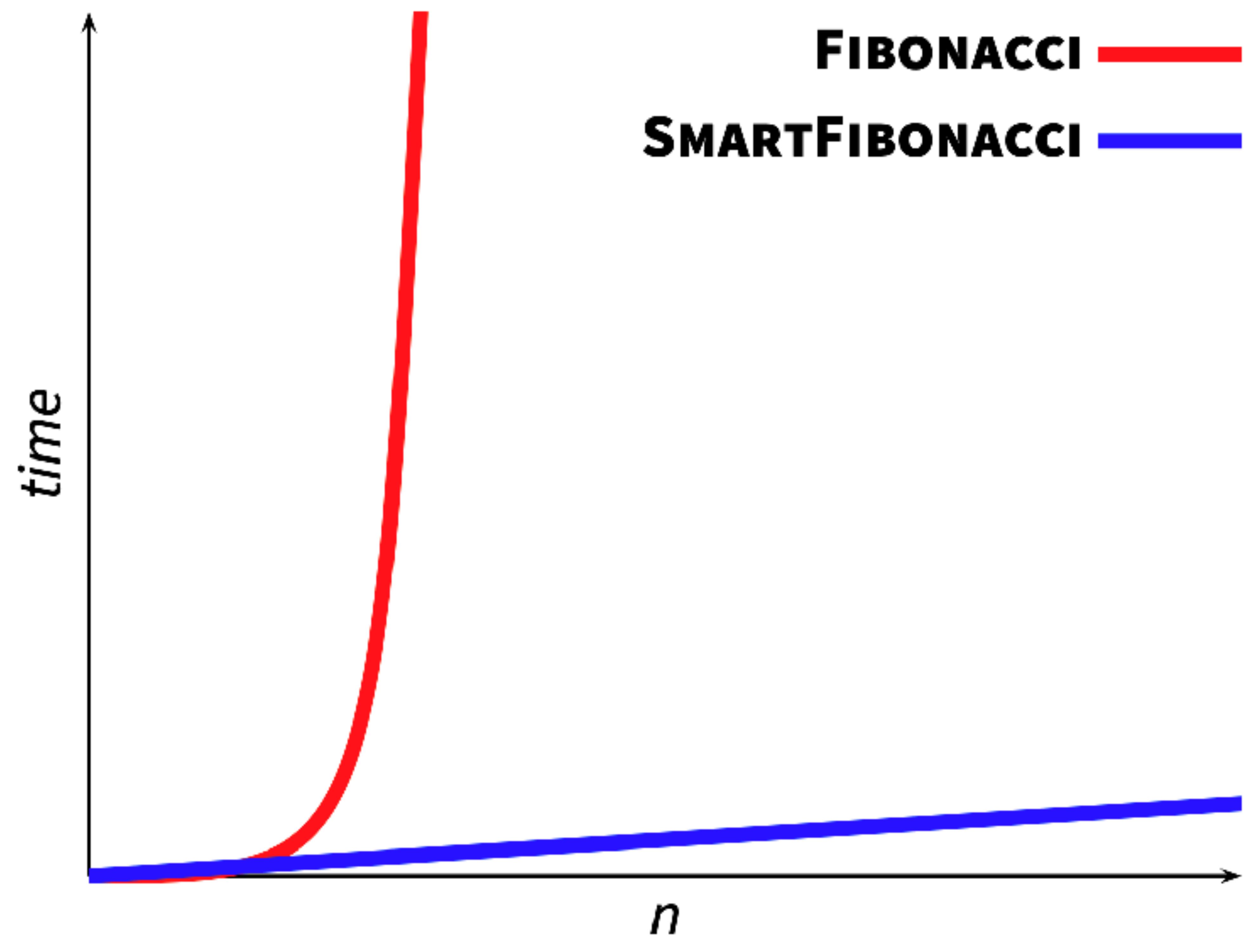
- We informally characterized our two Fibonacci algorithms

```
FIBONACCI(n)
if n == 0
        return 0
elseif n == 1
        return 1
else return FIBONACCI(n − 1) + FIBONACCI(n − 2)
```

```
SMARTFIBONACCI(n)
if n == 0
        return 0
elseif n == 1
        return 1
else pprev = 0
        prev = 1
        for i = 2 to n
                f = prev + pprev
                pprev = prev
                prev = f
return f
```

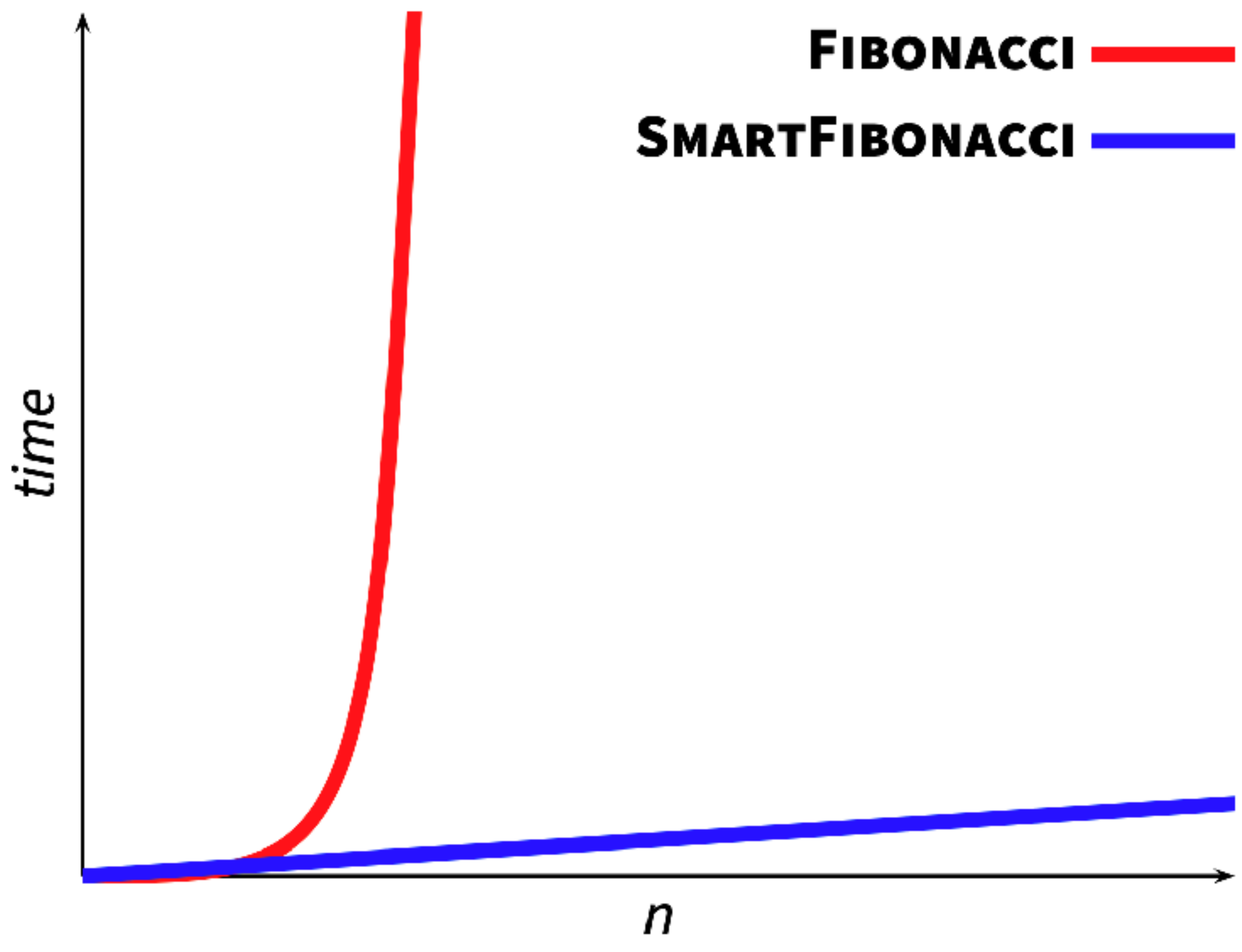
Recap: Slow vs. Fast Fibonacci

- We informally characterized our two Fibonacci algorithms



Recap: Slow vs. Fast Fibonacci

- We informally characterized our two Fibonacci algorithms as
  - **exponential** in n (FIBONACCI)
  - **linear** in n (SMARTFIBONACCI)



Recap: The Random-Access Machine Model

- A model of the computer that **characterizes the complexity of algorithms**
  - in general, and
  - in a way that is specific to the algorithms, but
  - independent of implementation details
- The model has only **basic types** and can perform only **basic steps**
- Each type of basic steps takes **a constant time**


IGNORE EVERY CONSTANT COST FACTOR
imgflip.com

Recap: Order of Growth

- We measure the **complexity** of an algorithm as a function of the **size of the input** (a **problem-dependent** variable or set of variables)
- In general we are interested in the **worst case complexity** to provide (possibly conservative) **guarantees** on how much time it takes for an algorithm to finish while fed with input of increasing size
- Other types of analysis exist:
  - Best case
  - Average case

Recap: Order of Growth

- Higher-order terms **dominate**, i.e., they grow at faster pace, lower-order terms
- Thus, we **ignore** lower-order terms while characterizing the asymptotic worst case behavior of the algorithms
- For instance,
  - given $T(n) = an^2 + bn + c$
  - we consider only the dominant factor $\mathbf{n^2}$
  - we conclude that $T(n)$ is a **quadratic function** in n
  - we write $T(n) = \mathbf{\Theta(n)^2}$

Recap: "Better" Algorithms

- We study the **asymptotic efficiency** of algorithms and consider the size of the input **in the limit**
- We consider one algorithm A_1 to be more efficient than another A_2 if A_1's **worst-case** running time **has a lower order of growth** than A_2's **<u>asymptotically</u>**
- This does NOT imply that A_1 is **always** more efficient than A_2; however, for **large enough inputs**, A_1 will be quicker than A_2 in the worst case

Asymptotic Notations

- We use asymptotic notations **primarily** to describe the running times of algorithms. But asymptotic notation applies to **functions**; thus, we can use them to characterize other aspects of algorithms (e.g., the amount of space they use, aka Space Complexity)
- Standard notations include the Θ-notation, the O- and o-notation, and the Ω- and ω-notation
- Each notation captures an asymptotic notion that characterizes a target function bounds:
  - Is it **above, below** or **between** other functions?
  - Is this bound **tight**?
- Note: the function under analysis might **not be completely known**

$\mathbb{A}$ *absolutely at most c*

- Let $\mathbb{A}(c)$ indicate a quantity that is absolutely at most c
    - e.g., $x = \mathbb{A}(2)$ means that $|x| \leq 2$
- When $x = \mathbb{A}(y)$ we say that "*x is absolutely at most y*"

- **$x = \mathbb{A}(y)$ <u>does not mean</u> that x equals $\mathbb{A}(y)$**
    - $\mathbb{A}(y)$ denotes a set of values
    - $x = \mathbb{A}(y)$ really means $x \in \mathbb{A}(y)$

$\mathbb{A}$ *absolutely at most c*

- $\mathbb{A}(x) + \mathbb{A}(y) = \mathbb{A}(x+y)$
- $\mathtt{k}\mathbb{A}(x) = \mathbb{A}(\mathtt{k}y)$
- $\mathbb{A}(x)\mathbb{A}(y) = \mathbb{A}(xy)$
- $X = \mathbb{A}(3) \Rightarrow X = \mathbb{A}(4)$, but $x = \mathbb{A}(4) \not\Rightarrow x = \mathbb{A}(3)$
- $X = \mathbb{A}(n-1) \Rightarrow X = \mathbb{A}(n^2)$, *for all* n

From 𝔸 to O

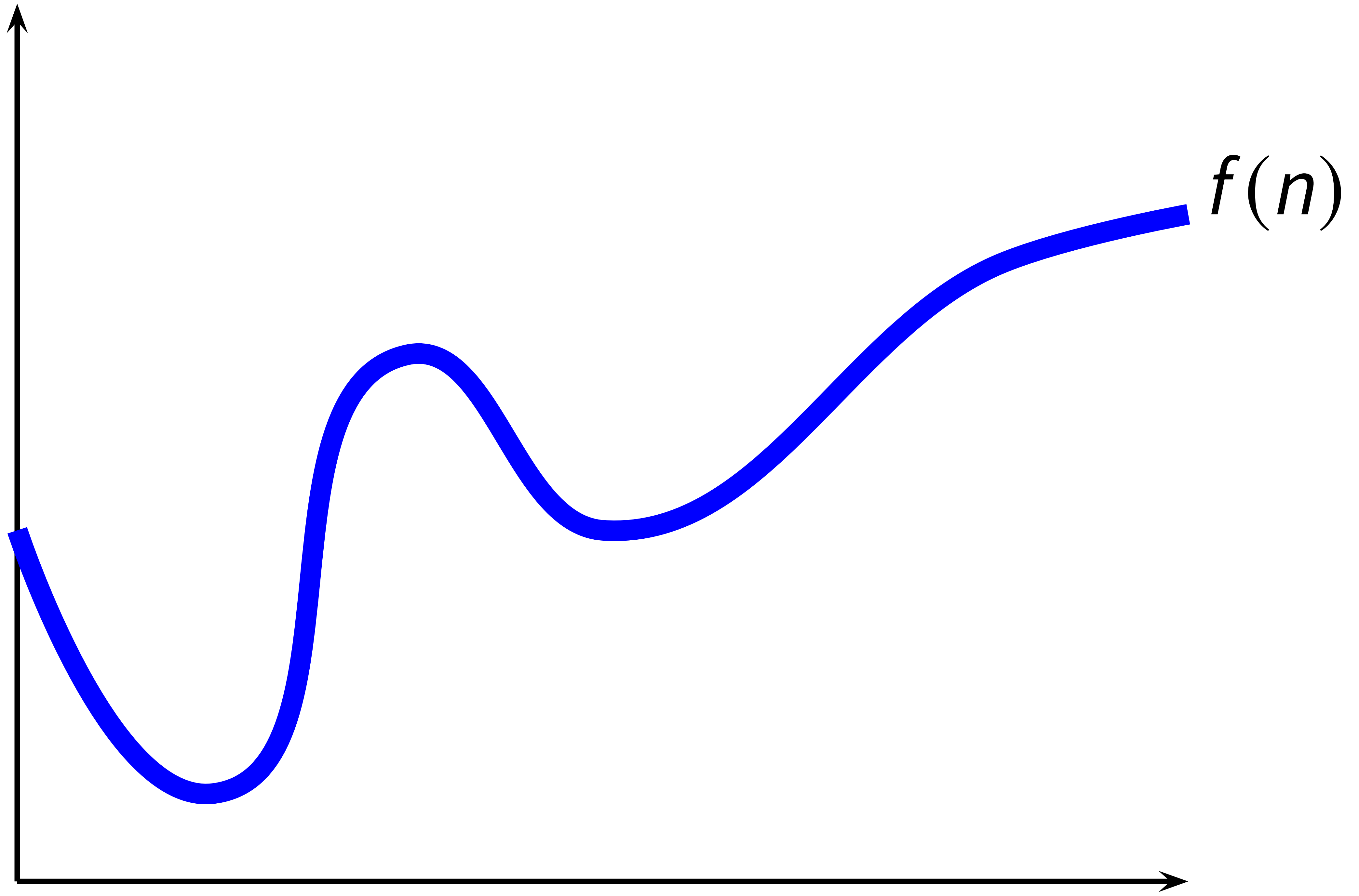
- If f(n) is such that **f(n) = c𝔸(g(n))** for all n **sufficiently large** and for some **positive constant** c (c>0), then we say that **f(n) = O(g(n))**
    - *read "f(n) is big-oh of g(n)" or simply "f(n) is oh of g(n)"*

- O(g(n)) denotes a **family of functions**

- **Warning: f(n) = O(g(n)) <u>does not mean</u> that f(n) equals O(g(n))**

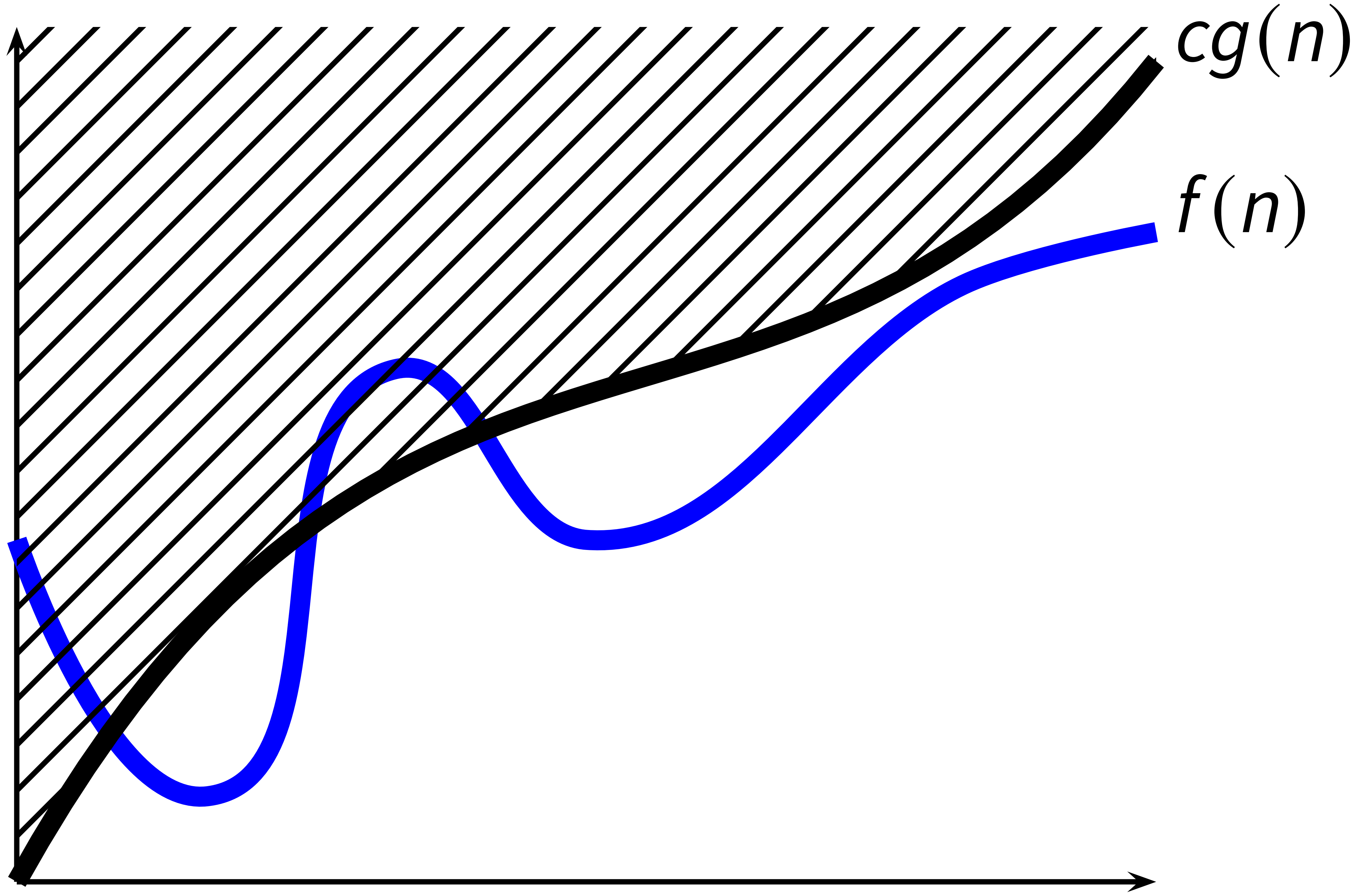
O-Notation

- Given a function g(n), we define the **family of functions** O(g(n))



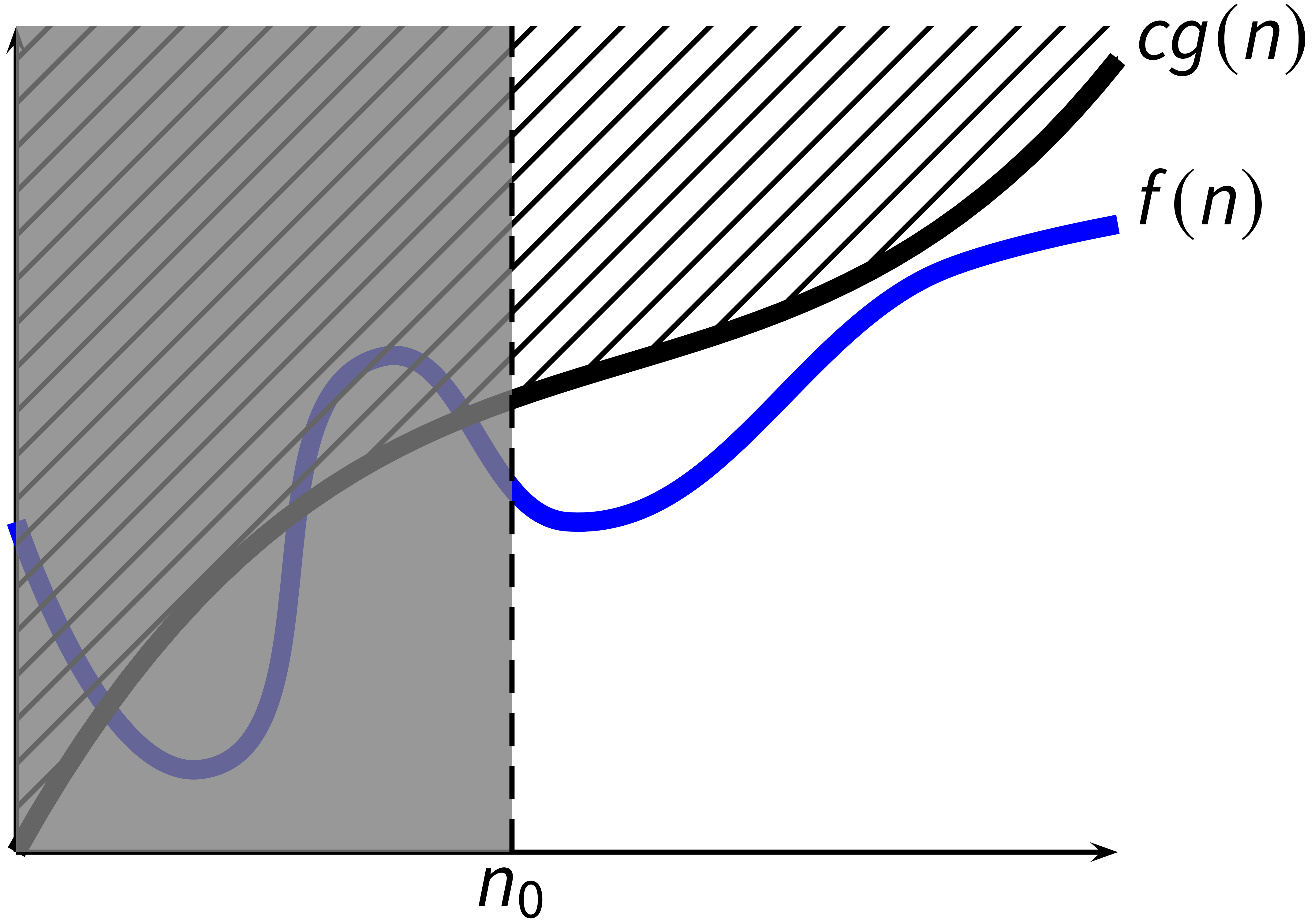
O-Notation

- Given a function g(n), we define the **family of functions** O(g(n))



O-Notation

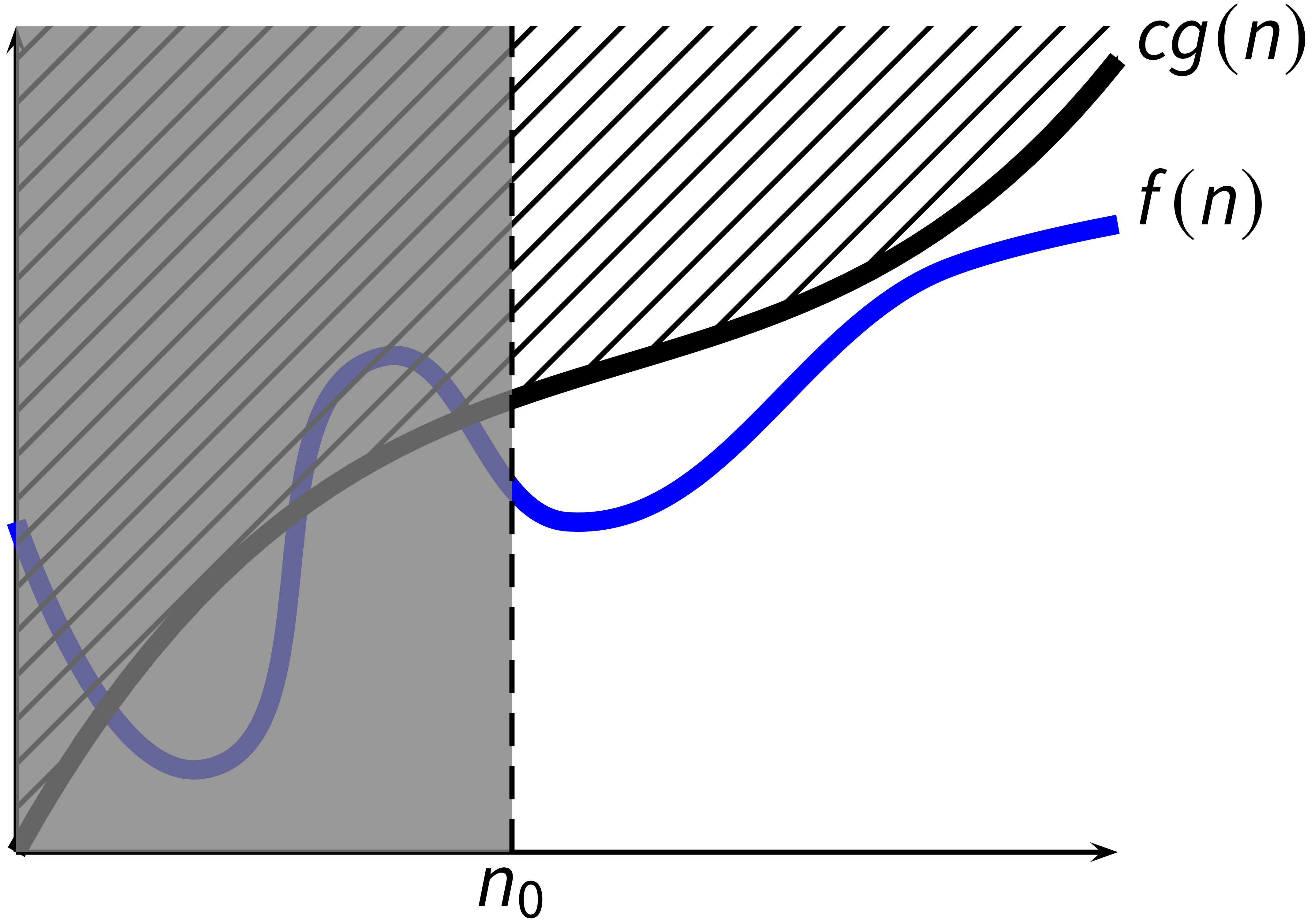
- Given a function g(n), we define the **family of functions** O(g(n))



$$O(g(n)) = \{f(n) : \exists c>0, \exists n_0 >0 : 0 \leq f(n) \leq cg(n) \text{ for } n \geq n_0\}$$

O-Notation

- Given a function g(n), we define the **family of functions** O(g(n))



f(n) = O(g(n)), i.e., f(n) ∈ O(g(n)) which reads "*f(n) is big-oh of g(n)*"

Exercise

- Given T(n) find a suitable O(n)

$n^2 + 10n + 100$	
$n+10\log(n)$	
$n \log(n) + n \operatorname{sqrt}(n)$	
$2^{n/6} + n^7$	
$(10+n)/(n^2)$	

Exercise

- Given T(n) find a suitable O(n)

$n^2 + 10n + 100$	$O(n^2)$, $O(n^3)$
$n+10\log(n)$	$O(2^n)$
$n \log(n) + n\ \mathrm{sqrt}(n)$	$O(n^2)$
$2^{n/6} + n^7$	$O((1.5)^n)$
$(10+n)/(n^2)$	$O(1)$

From O to Ω

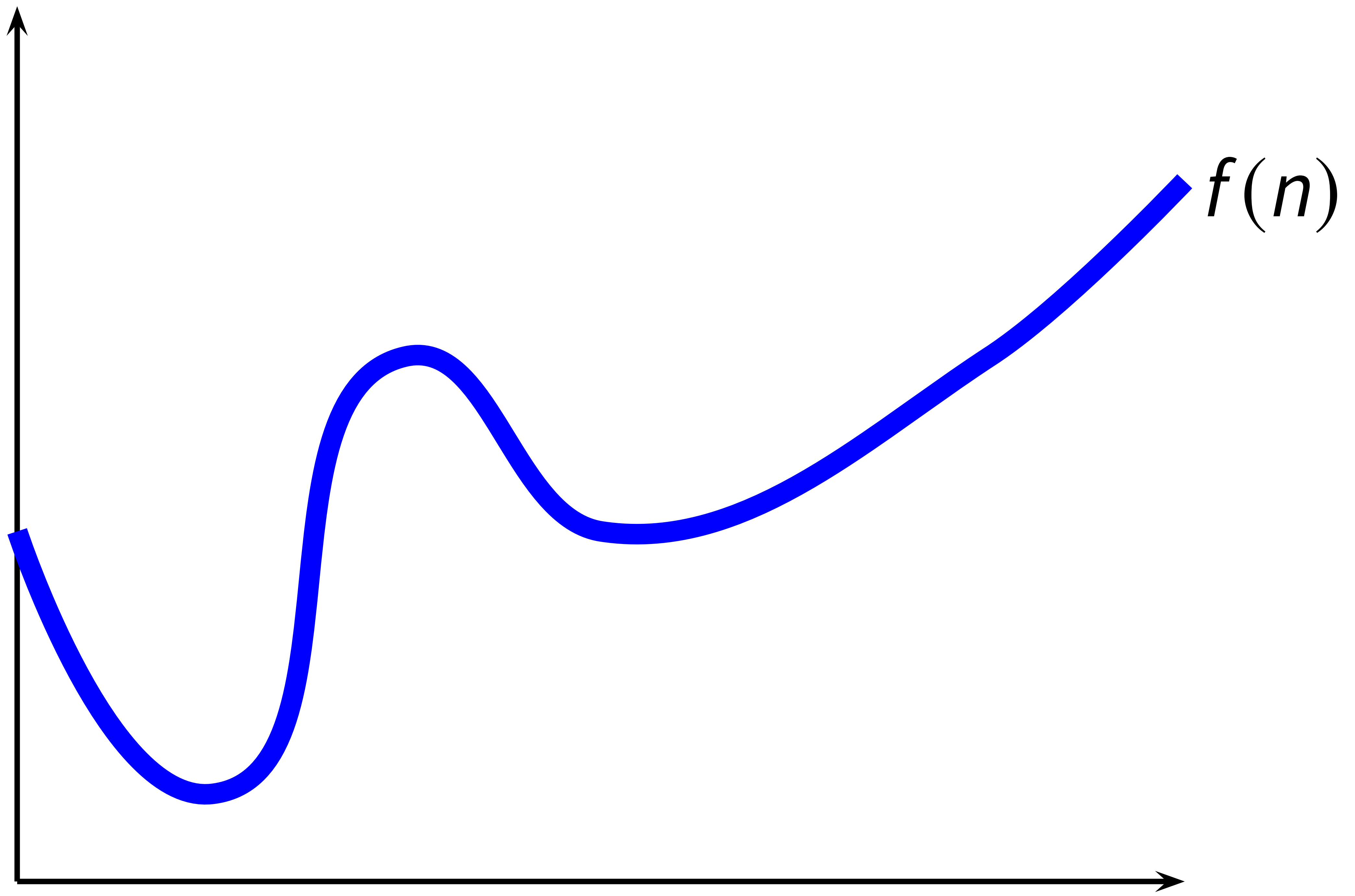
- If g(n) = O(f(n)) then we can **also** say that g(n) **asymptotically dominates** f(n), which we can also write as **f(n) = Ω(g(n))**
    - *read "f(n) is big-omega of g(n)" or simply "f(n) is omega of g(n)"*

- Ω(g(n)) denotes a **family of functions**

- **Warning: f(n) = Ω(g(n)) does not mean that f(n) equals Ω(g(n))**

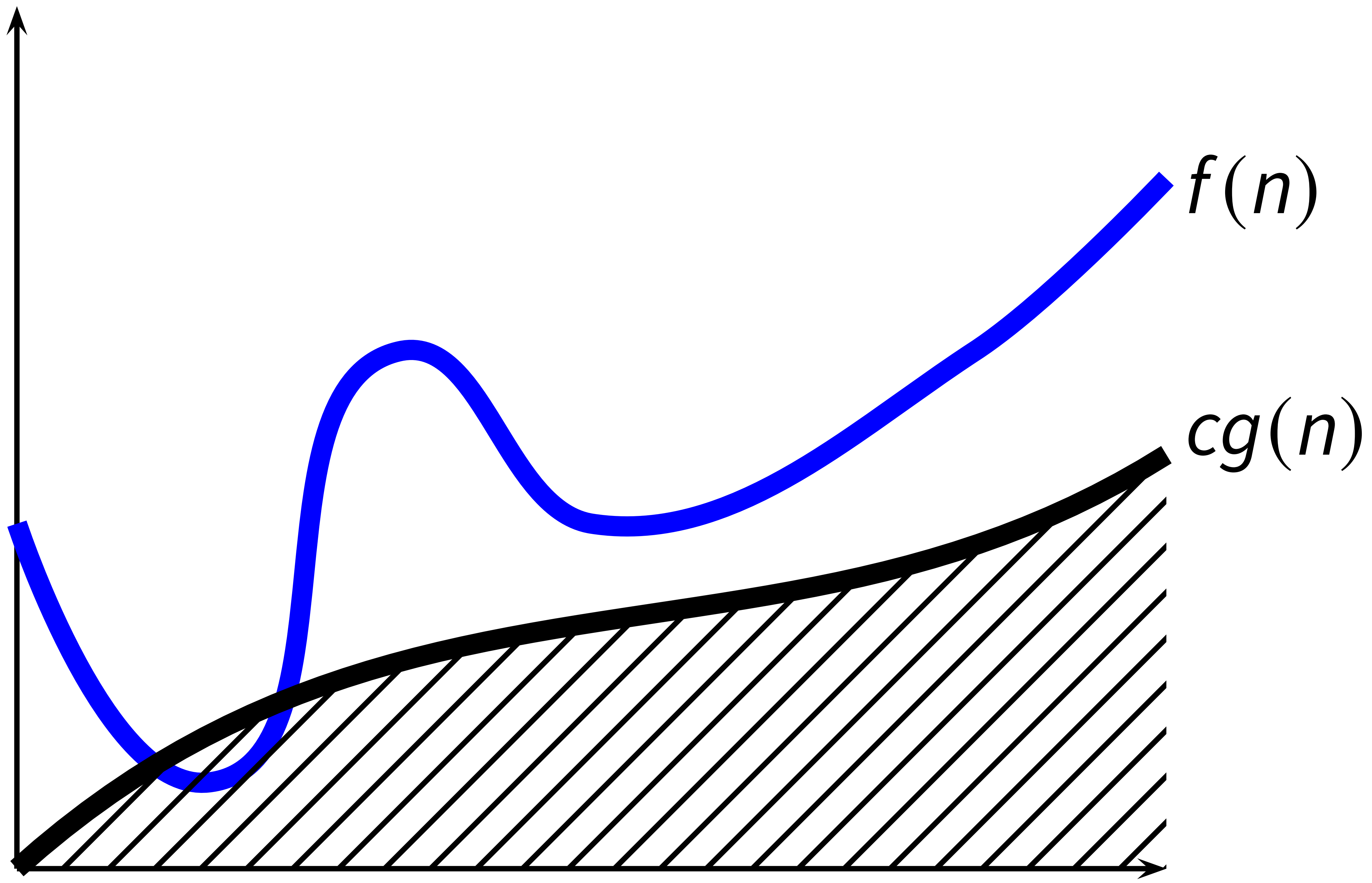
Ω-Notation

- Given a function g(n), we define the **family of functions** Ω(g(n))



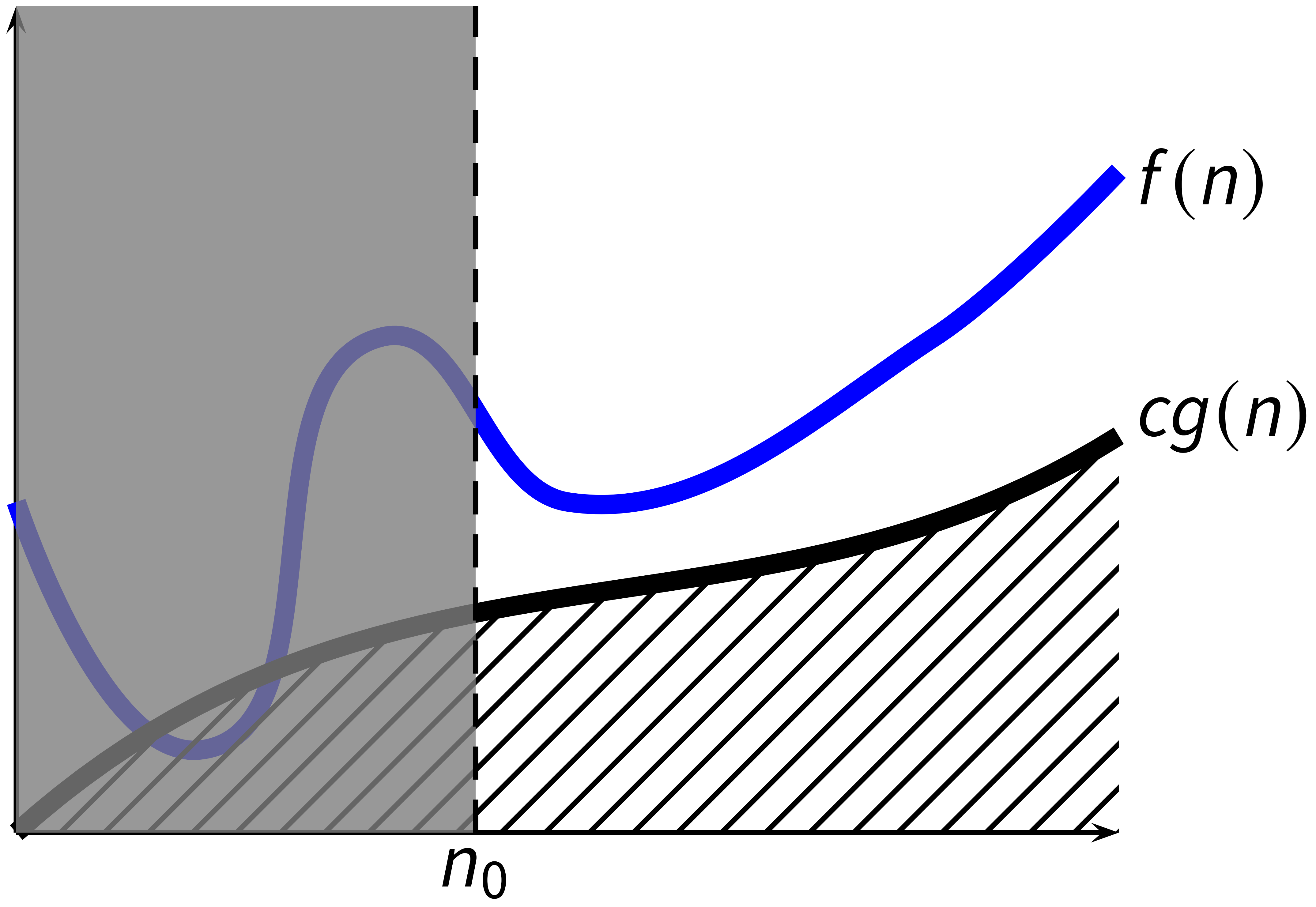
Ω-Notation

- Given a function g(n), we define the **family of functions** Ω(g(n))



Ω-Notation

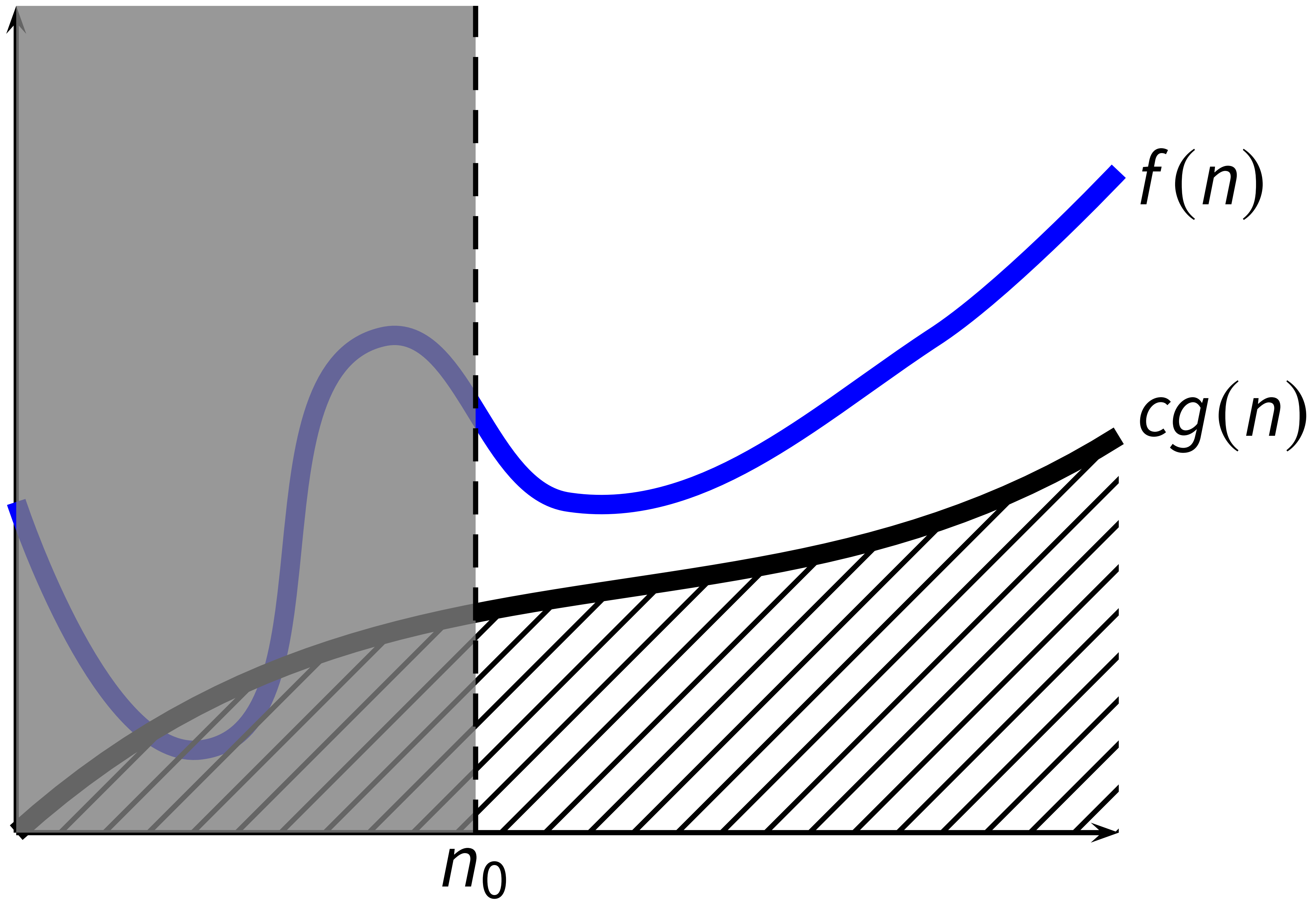
- Given a function g(n), we define the **family of functions** Ω(g(n))



$\Omega(g(n))=\{f(n) :\ \exists\, c>0, \exists\ \exists\, n_0>0 : 0 \leq cg(n) \leq f(n) \text{ for } n \geq n_0\}$

Ω-Notation

- Given a function g(n), we define the **family of functions** Ω(g(n))



f(n) = Ω(g(n)), i.e., f(n) ∈ Ω(g(n)) "*f(n) is omega of g(n)*"

Exercise

- Given T(n) find a suitable Ω(n)

$n^2 + 10n + 100$	
$n+10\log(n)$	
$n \log(n) + n\ \mathrm{sqrt}(n)$	
$2^{n/6} + n^7$	
$(10+n)/(n^2)$	

Exercise

- Given T(n) find a suitable Ω(n)

$n^2 + 10n + 100$	$\Omega(n^2)$, $\Omega(n)$
$n+10\log(n)$	$\Omega(n)$, $\Omega(\log(n))$
$n \log(n) + n\,\mathrm{sqrt}(n)$	$\Omega(n)$
$2^{n/6} + n^7$	$\Omega(2^{n/6})$, $\Omega(n^7)$
$(10+n)/(n^2)$	$\Omega(1/n)$

From O and Ω to Θ

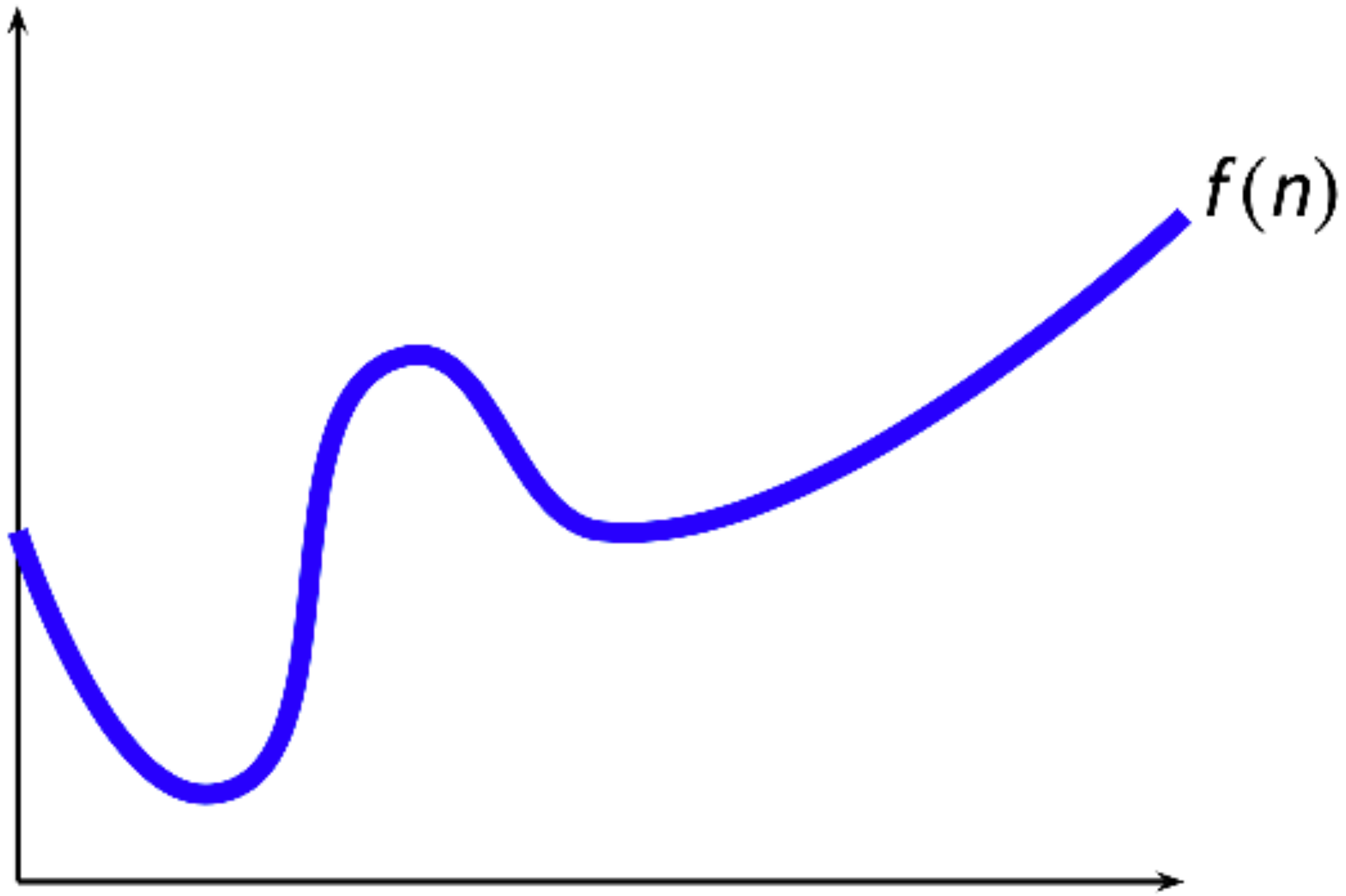
- When $f(n) = O(g(n))$ **and** $f(n) = \Omega(g(n))$, then $\mathbf{f(n) = \Theta(g(n))}$
    - *read "f(n) is theta of g(n)"*

- $\Theta(g(n))$ denotes a **family of functions**

- **Warning: $f(n) = \Theta(g(n))$ does not mean that $f(n)$ equals $\Theta(g(n))$**

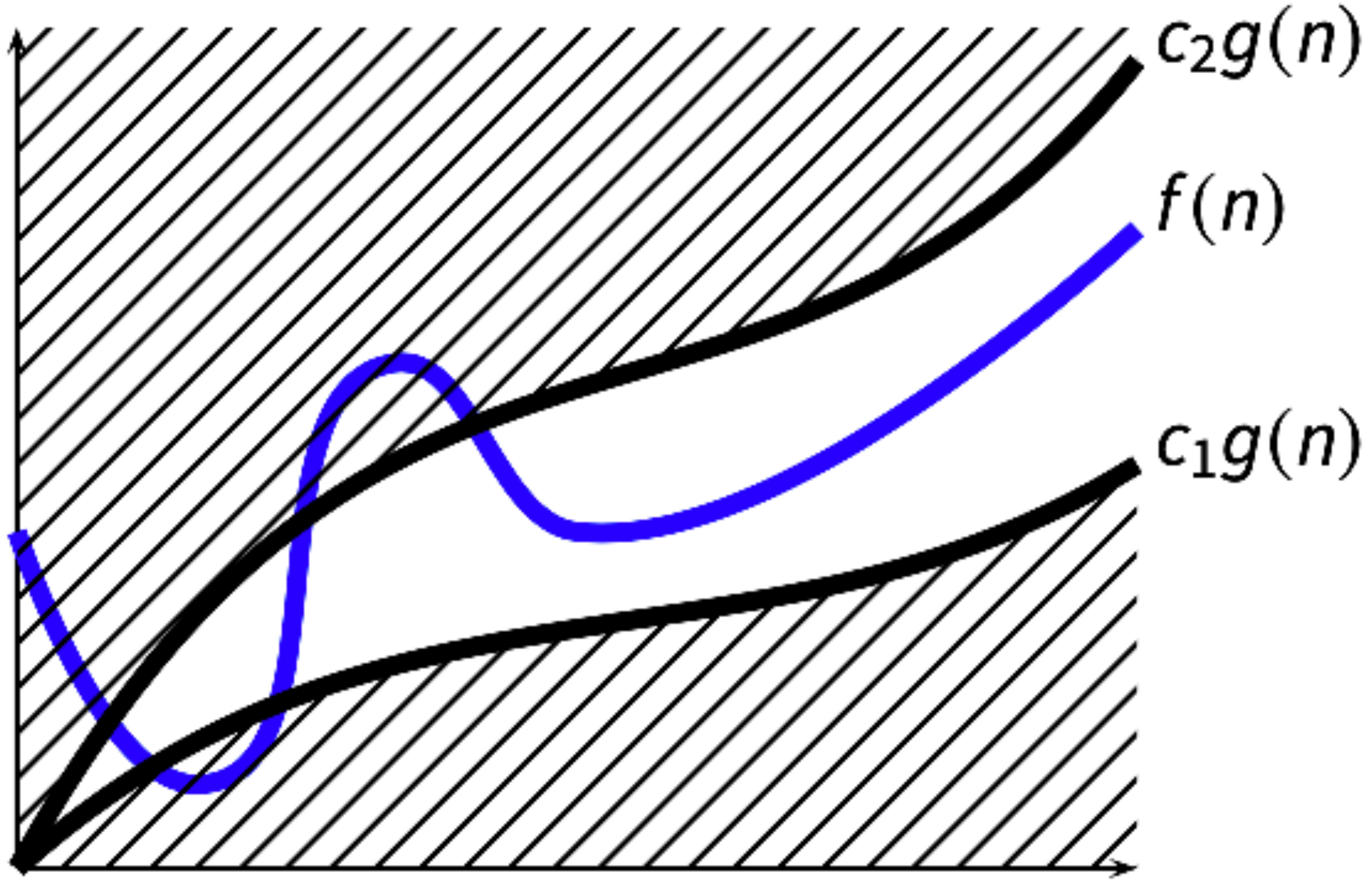
Θ-Notation

- Given a function g(n), we define the **family of functions** Θ(g(n))



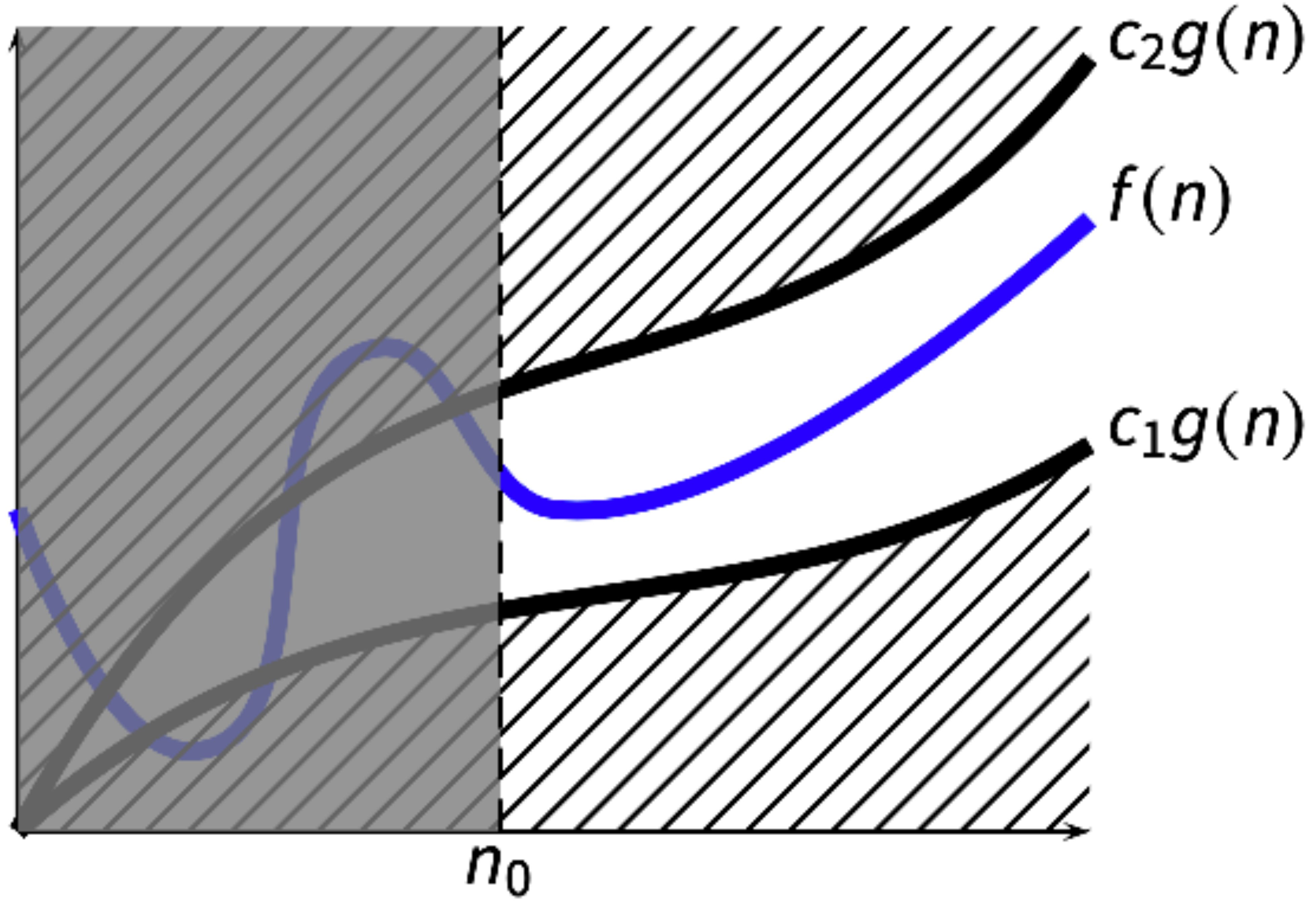
Θ-Notation

- Given a function g(n), we define the **family of functions** Θ(g(n))



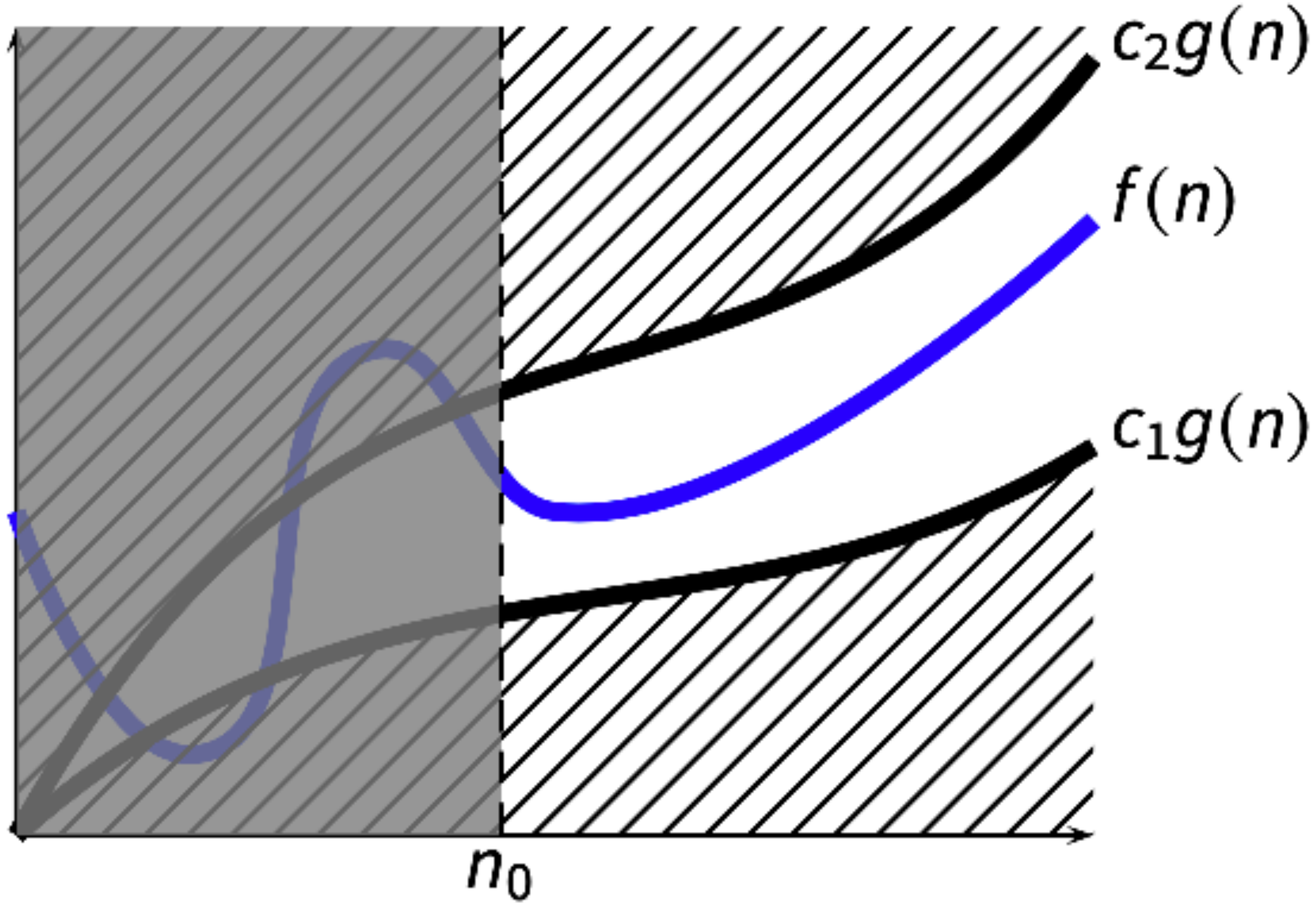
Θ-Notation

- Given a function g(n), we define the **family of functions** Θ(g(n))



Θ-Notation

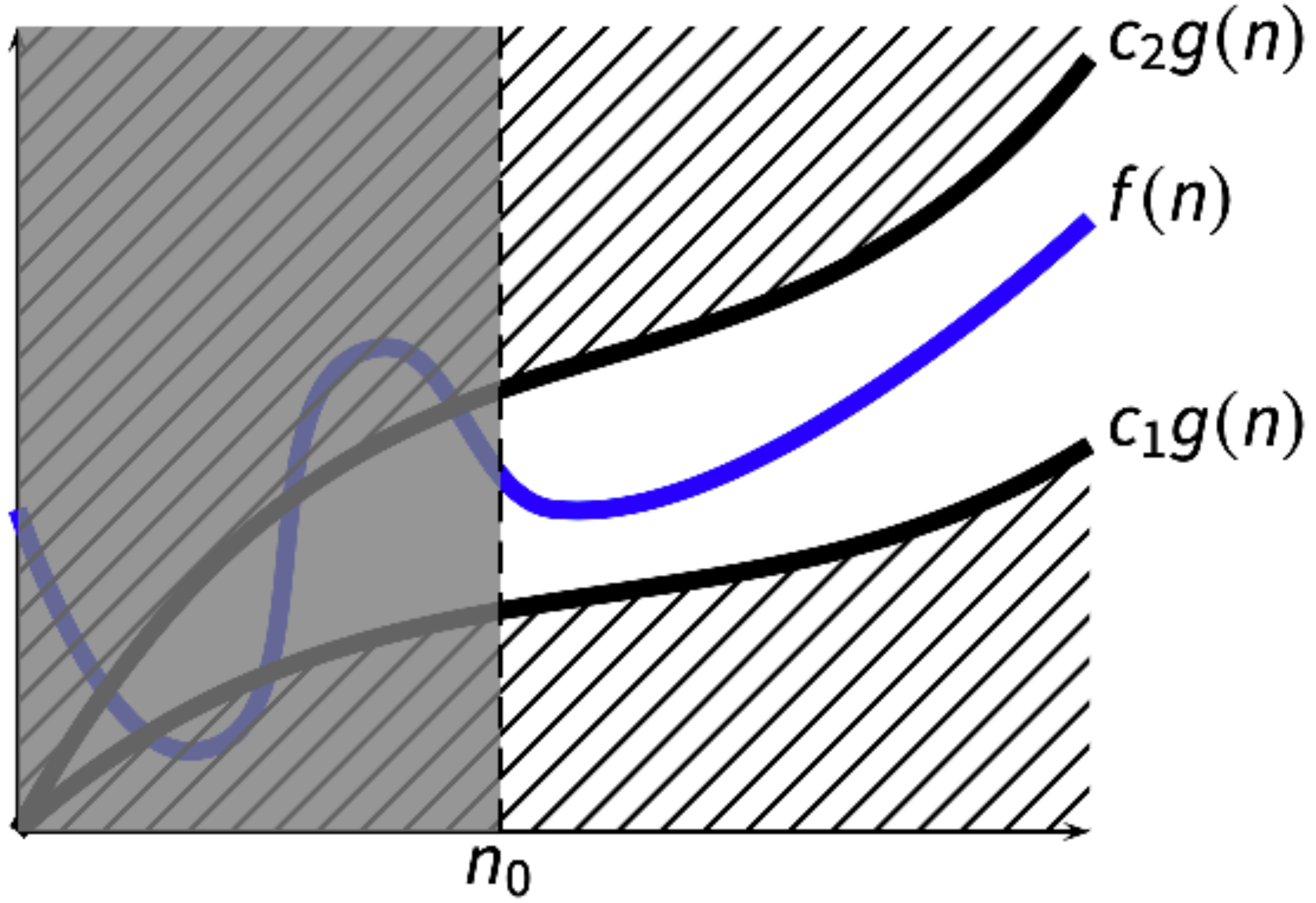
- Given a function g(n), we define the **family of functions** Θ(g(n))



$$\Theta(g(n)) = \{f(n) : \exists\, c_1 > 0, \exists\, c_2 > 0, \exists\, n_0 > 0 \mid 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for } n \ge n_0\}$$

Θ-Notation

- Given a function g(n), we define the **family of functions** Θ(g(n))



f(n) = Θ(g(n)), i.e., f(n) ∈ Θ(g(n)), which reads as "*f(n) is theta of g(n)*"

Exercise

- Given T(n) find a suitable Θ(n)

$n^2 + 10n + 100$	
$n+10\log(n)$	
$n \log(n) + n\, \mathrm{sqrt}(n)$	
$2^{n/6} + n^7$	
$(10+n)/(n^2)$	

Exercise

- Given T(n) find a suitable Θ(n)

$n^2 + 10n + 100$	$\Theta(n^2)$
$n+10\log(n)$	$\Theta(n)$
$n \log(n) + n\ \mathrm{sqrt}(n)$	$\Theta(n\ \mathrm{sqrt}(n))$
$2^{n/6} + n^7$	$\Theta(2^{n/6})$
$(10+n)/(n^2)$	$\Theta(1/n)$

Θ, O and Ω as Relations

- The Θ-, Ω-, and O-notation can be viewed as the **asymptotic version** of the "=", "≥", and "≤" **relations for functions**
- For two functions f(n) and g(n), $f(n) = \Omega(g(n))$ *and* $f(n) = O(g(n))$ *iff* $f(n) = \Theta(g(n))$ can be interpreted as saying $f \geq g \wedge f \leq g \Leftrightarrow f = g$
- When $f(n) = O(g(n))$ we say that g(n) is an **upper bound** for f(n)
  - g(n) **dominates** f (n)
- When $f(n) = \Omega(g(n))$ we say that g(n) is a **lower bound** for f(n)

Θ, O and Ω as Anonymous Functions

- We can use the Θ-, Ω-, and O-notation to represent **unknown or unspecified functions**

- For instance, $f(n) = 10n^2 + O(n)$ means that $f(n)$ is equals to $10n^2$ plus a(ny) function that we do not know (or care) which is **at most linear** in n ... **asymptotically** speaking

Exercise

- Are the following equalities true or false? And why?
  - $n^2 + 4n - 1 = n^2 + \Theta(n)$?
  - $n^2 + \Omega(n) - 1 = O(n^2)$?
  - $n^2 + O(n) - 1 = O(n^2)$?
  - $n \log(n) + \Theta(\mathrm{sqrt}(n)) = O(n\ \mathrm{sqrt}(n))$?

Exercise

- Are the following equalities true or false? And why?
  - $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES
  - $n^2 + \Omega(n) - 1 = O(n^2)$? NO
  - $n^2 + O(n) - 1 = O(n^2)$? YES
  - $n \log(n) + \Theta(\text{sqrt}(n)) = O(n\ \text{sqrt}(n))$? YES

Exercise

Are the following formulae true or false?

- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$?
- $f(n) = \Theta(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$?
- $f(n) = O(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = O(h(n))$?
- $n \log(n) + \Theta(\mathrm{sqrt}(n)) = O(n\ \mathrm{sqrt}(n))$?
- $n^2 + 10n + 100 = O(n \log(n))$?
- $n^2 + (1.5)^n = O(2^{n/2})$?

Exercise

Are the following formulae true or false?

- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$? Yes
- $f(n) = \Theta(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$? Yes
- $f(n) = O(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = O(h(n))$? Yes
- $n \log(n) + \Theta(\mathrm{sqrt}(n)) = O(n\ \mathrm{sqrt}(n))$? Yes
- $n^2 + 10n + 100 = O(n \log(n))$? No
- $n^2 + (1.5)^n = O(2^{n/2})$? No

Exercise @ Home

Are the following formulae true or false?

- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$?
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$?
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$?
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2 \log 2(n)})$?
- $\mathrm{sqrt}(n) = O(\log^2 n)$?

o-Notation

- The O-notation defines an upper bound that **might not** be asymptotically tight
  - $n \log(n) = O(n^2)$ is not asymptotically tight
  - $n^2 - n + 10 = O(n^2)$ is asymptotically tight
- The o-notation denotes upper bounds that **are not** asymptotically tight
- Given a function g(n), we define the families of functions o(g(n))as:

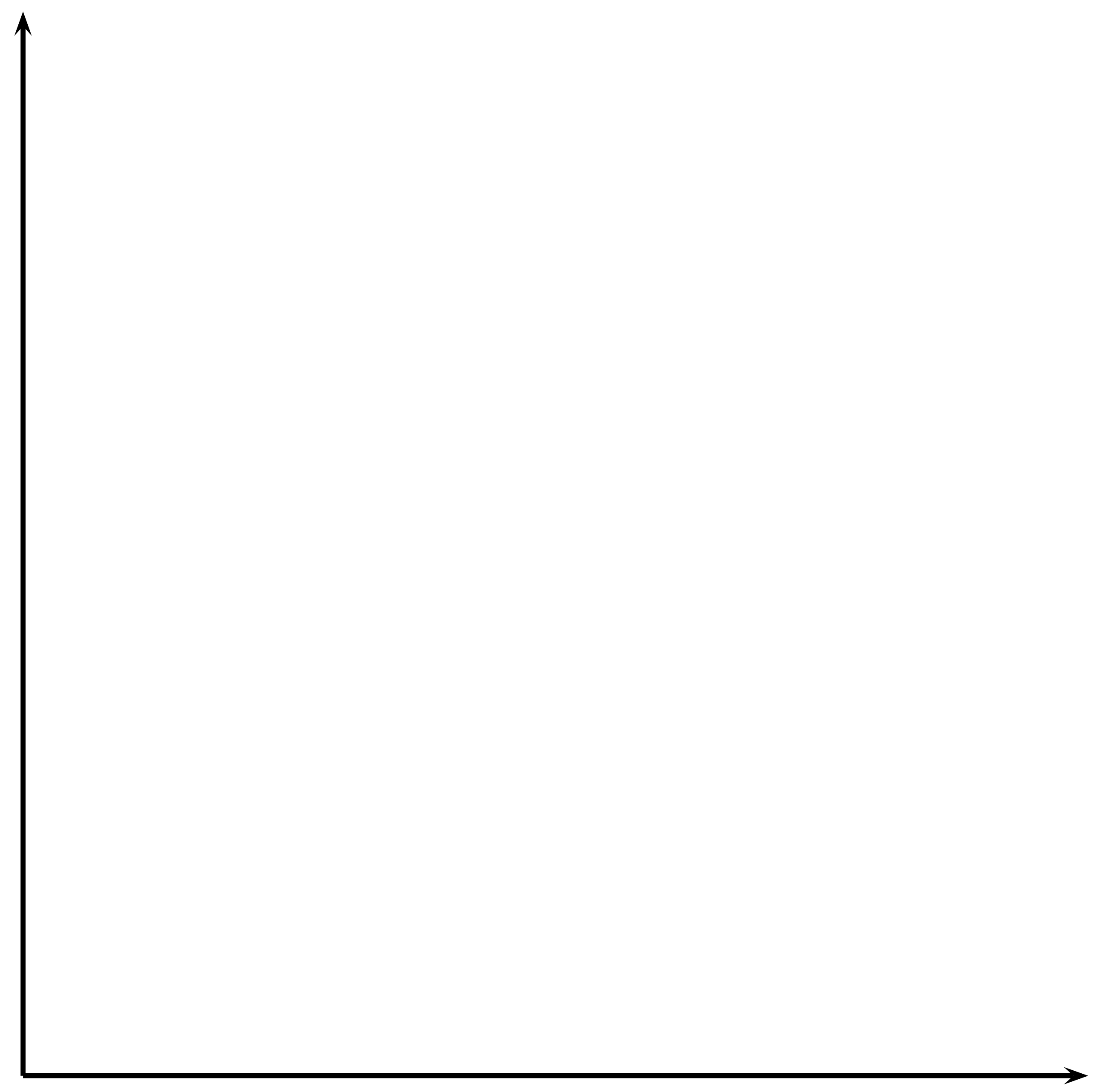
$$o(g(n)) = \{f(n): \forall c>0, \exists n_0 >0 : 0 \leq f(n) < c\, g(n) \text{ for } n \geq n_0\}$$

ω-Notation

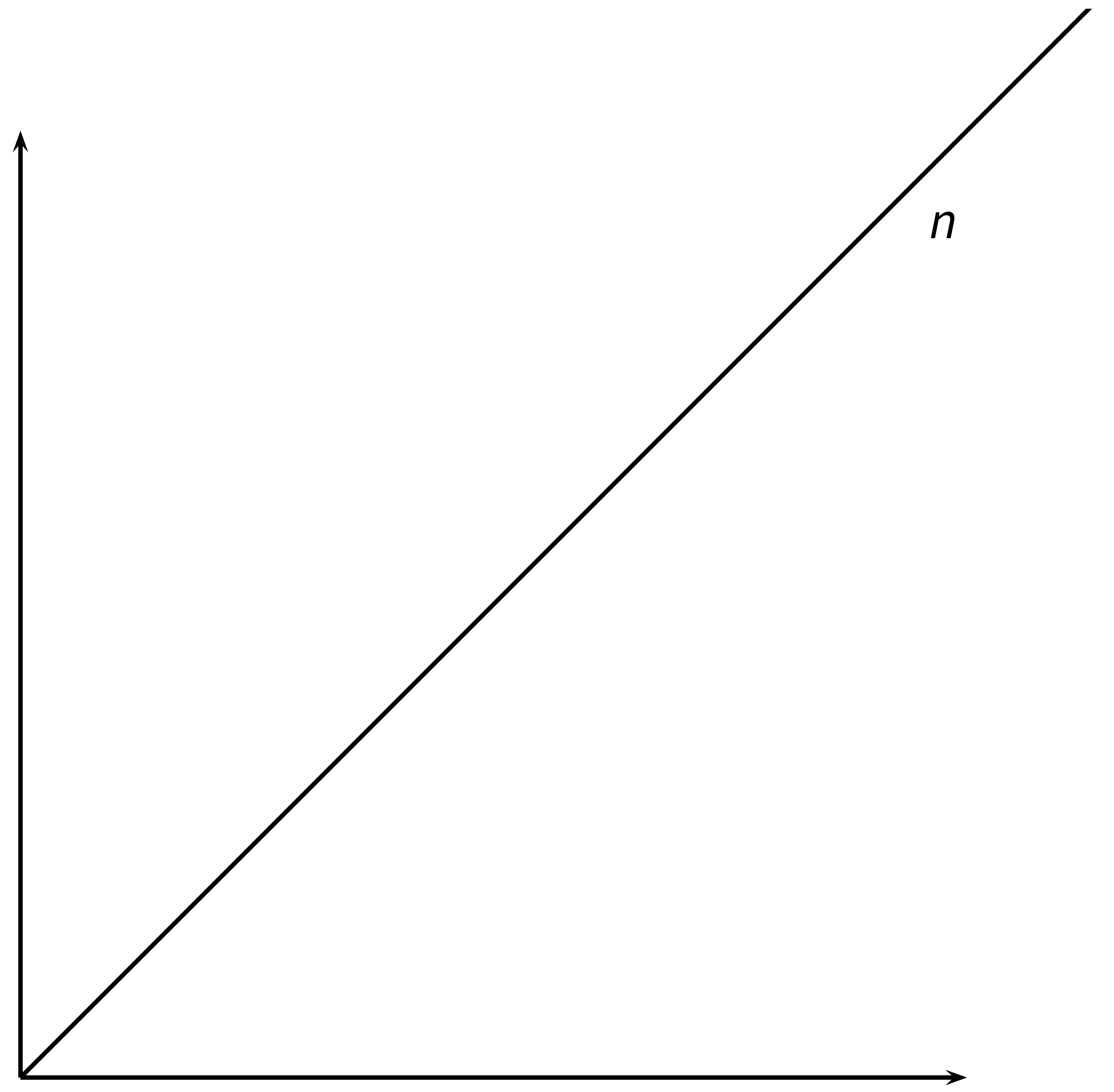
- The Ω-notation defines a lower bound that **might not** be asymptotically tight
  - $2^n = \Omega(n \log(n))$ is not asymptotically tight
  - $n + 4n \log(n) = \Omega(n \log(n))$ is asymptotically tight
- The ω-notation define lower bounds that **are not** asymptotically tight
- Given a function g(n), we define the families of functions ω(g(n)) as:

$$\omega(g(n)) = \{f(n): \forall\, c>0, \exists\, n_0 > 0 : 0 \leq cg(n) < f(n) \text{ for } n \geq n_0\}$$

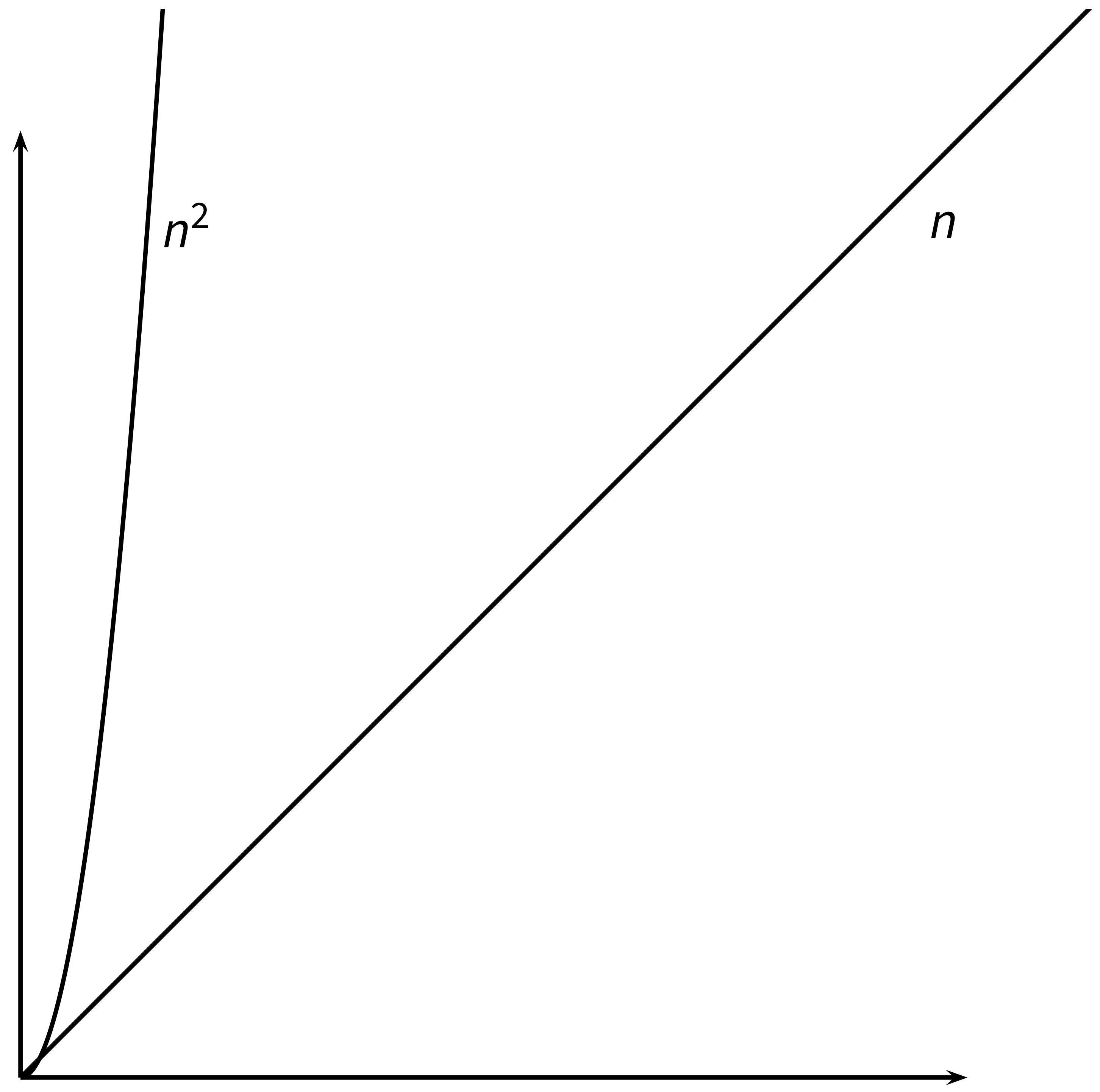
Visualizing the Relation between Functions



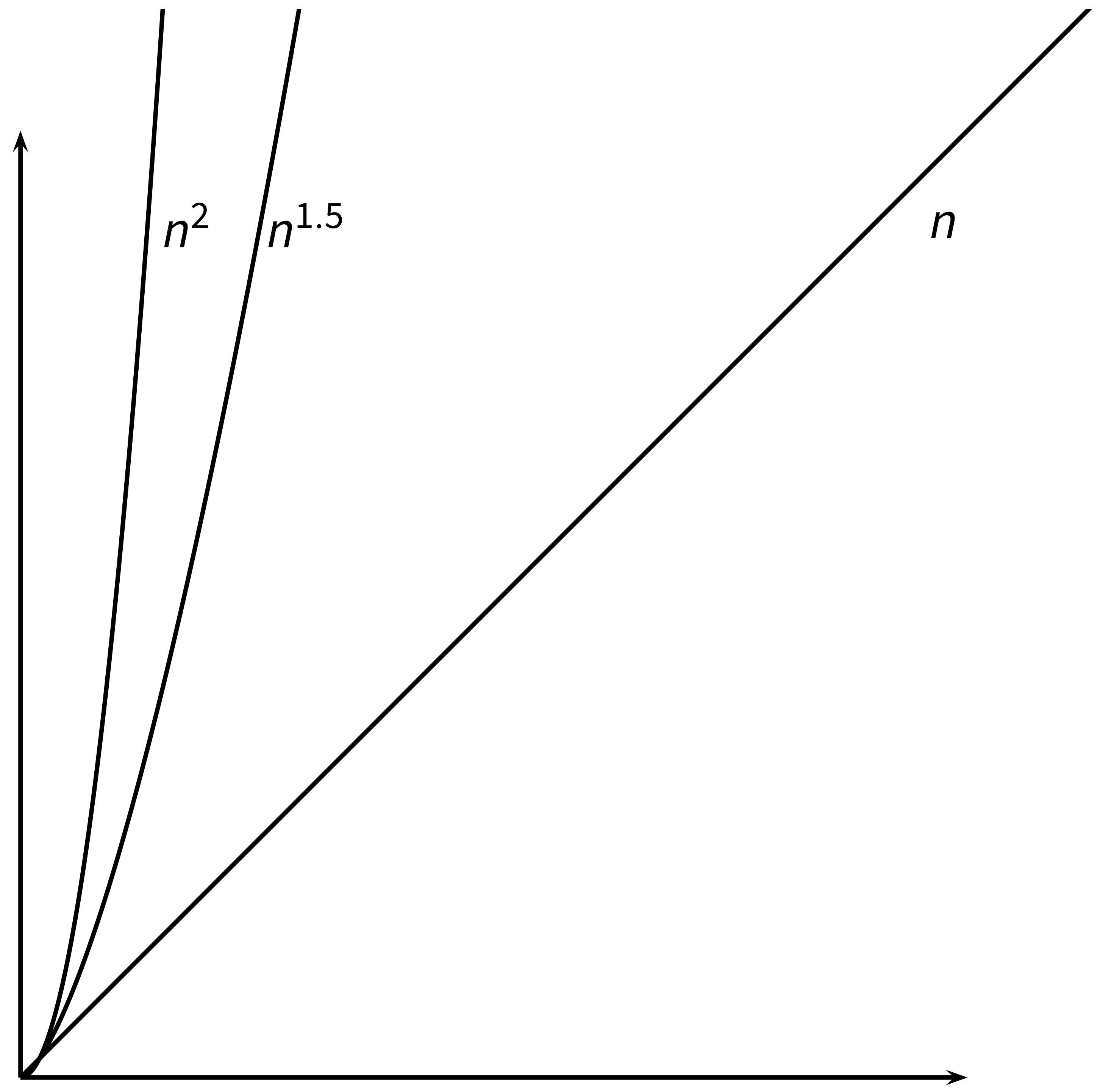
Visualizing the Relation between Functions



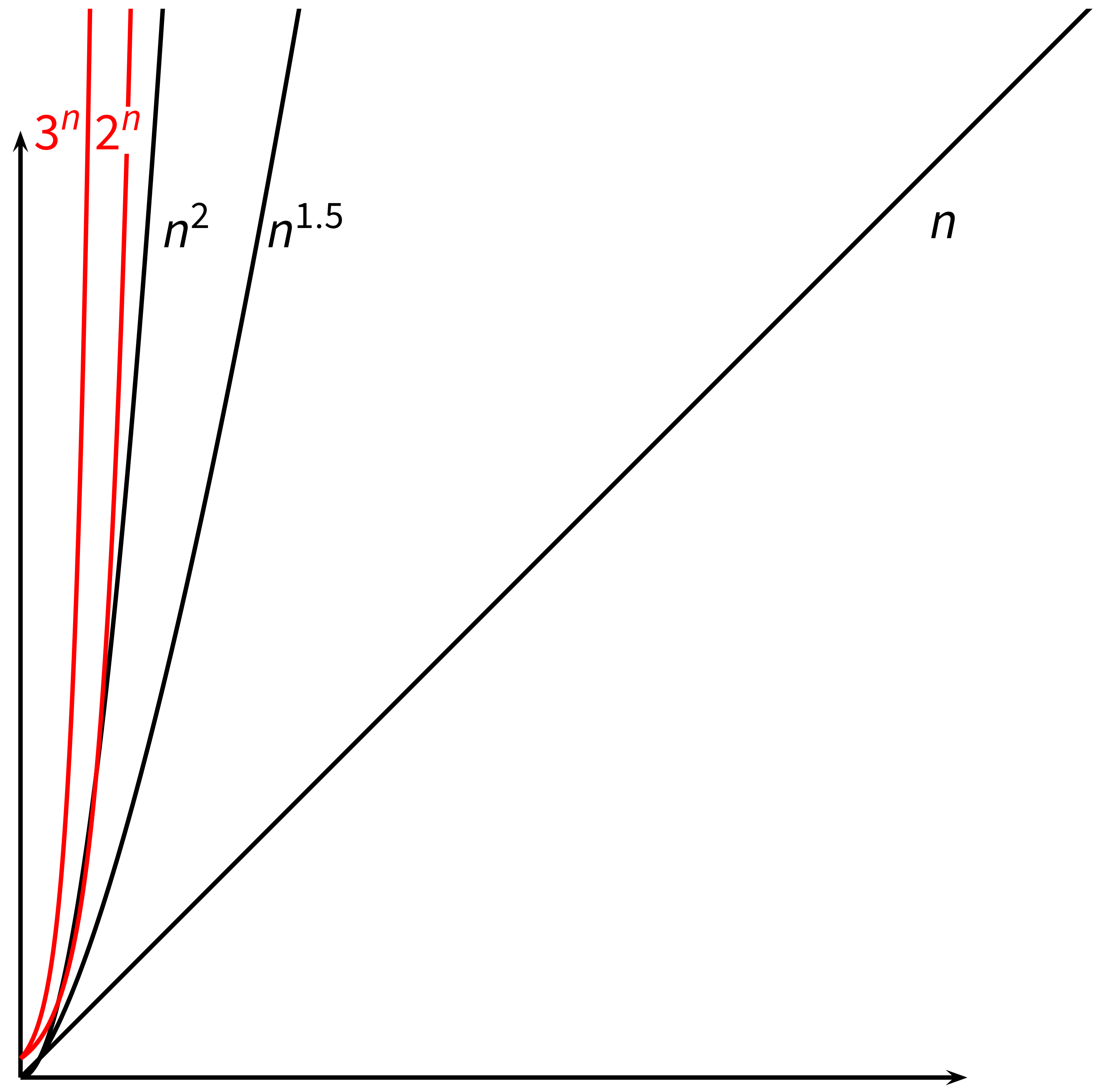
Visualizing the Relation between Functions



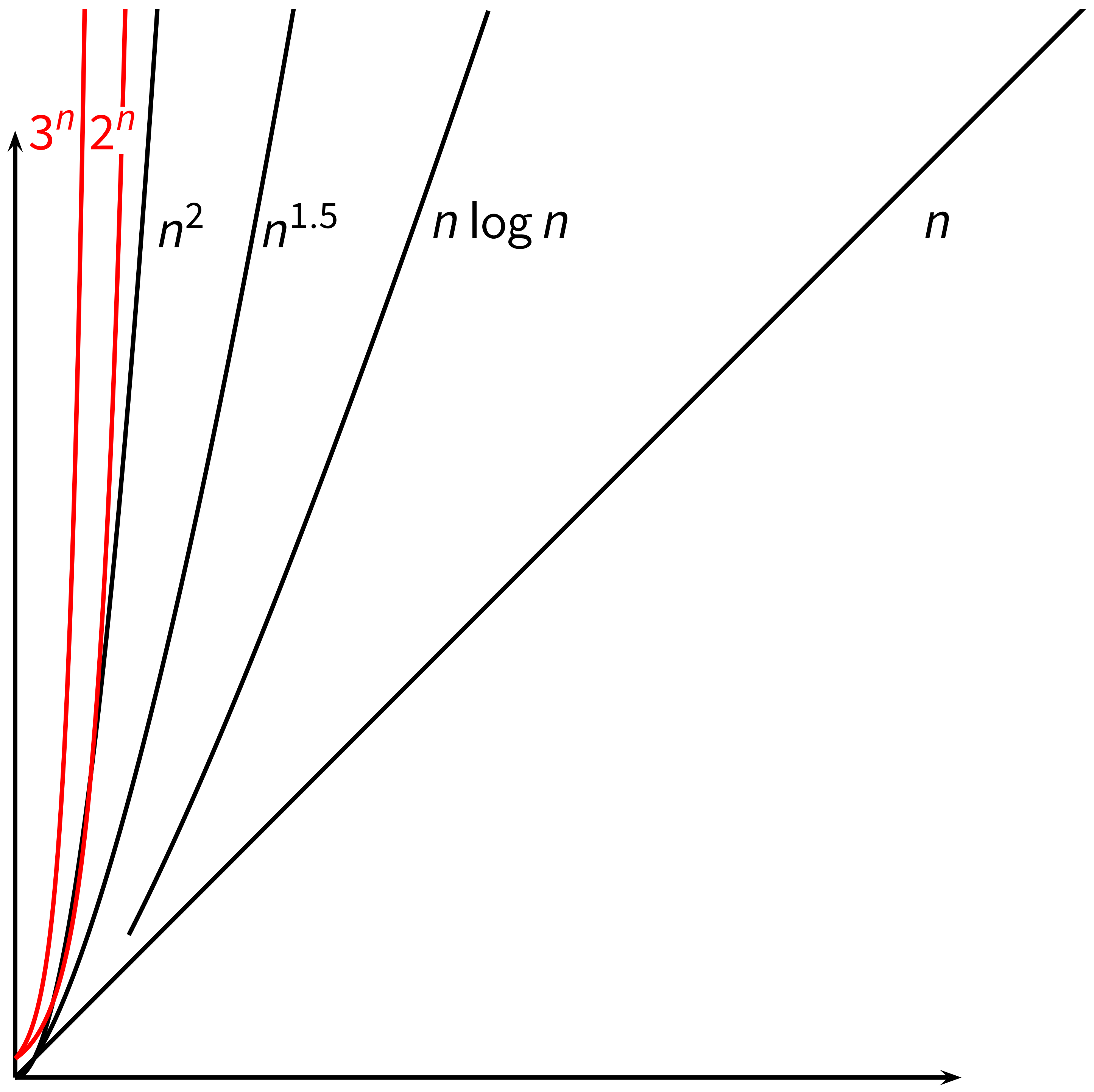
Visualizing the Relation between Functions



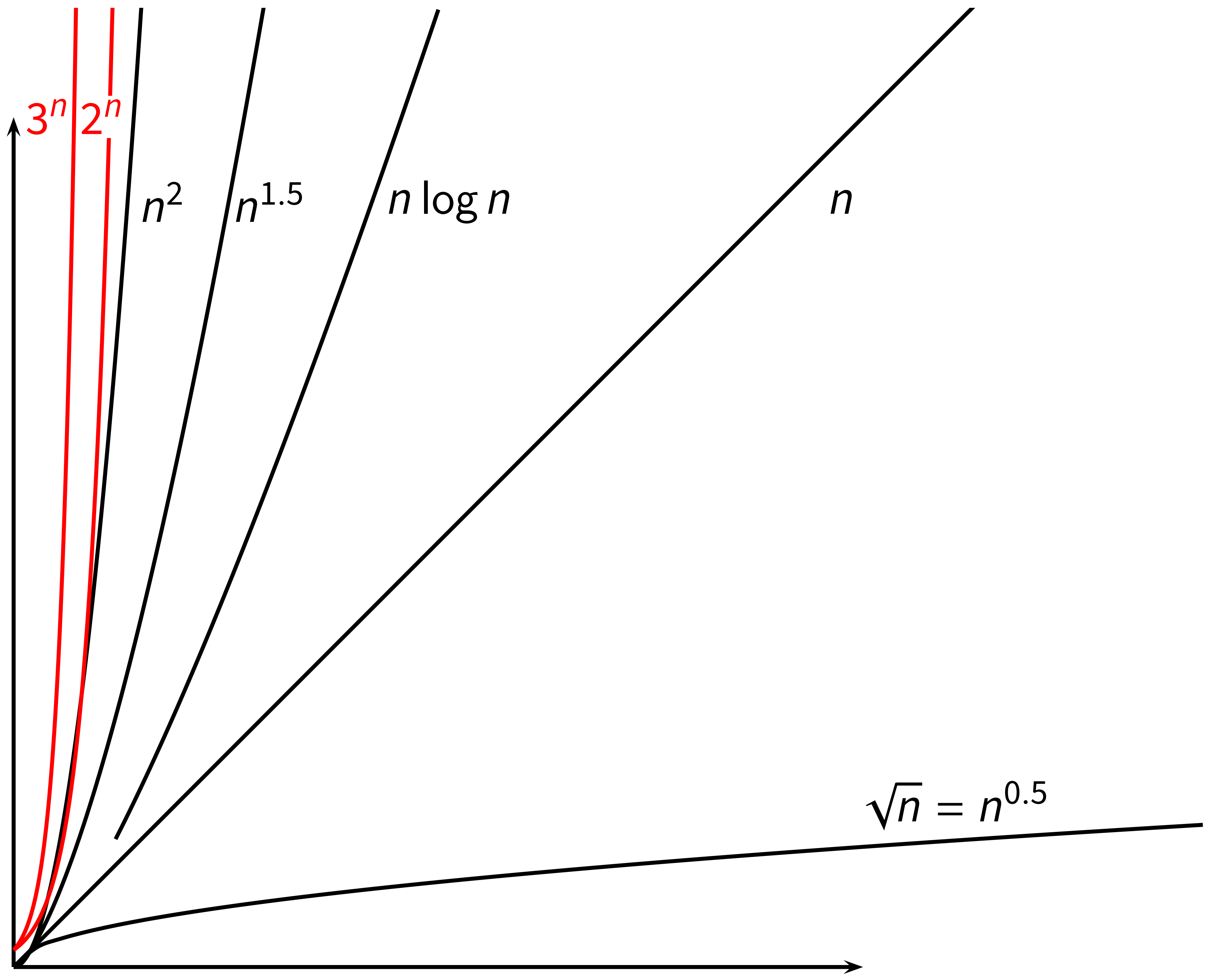
Visualizing the Relation between Functions



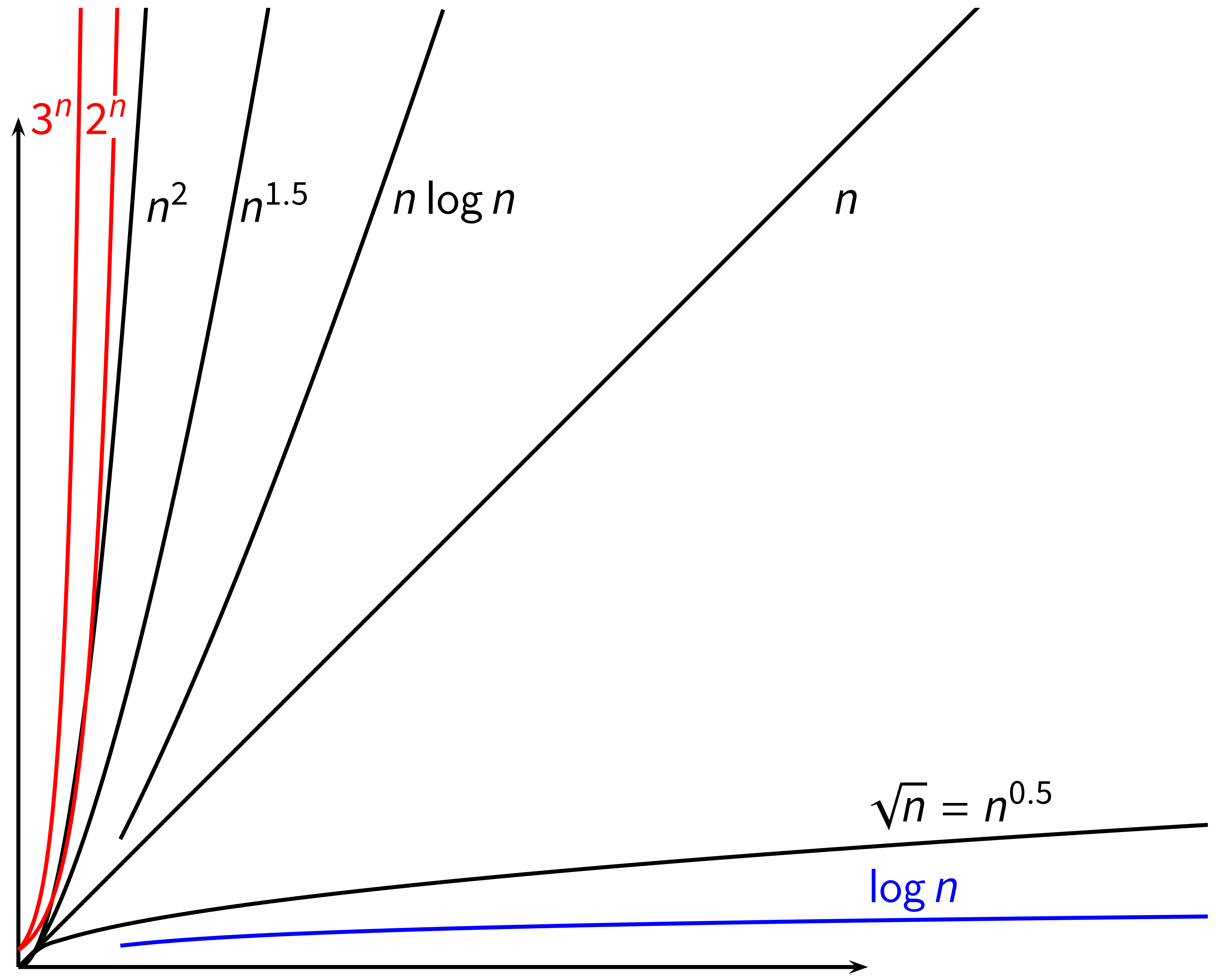
Visualizing the Relation between Functions



Visualizing the Relation between Functions



Visualizing the Relation between Functions



Visualizing the Relation between Functions