Basics of Complexity Analysis: The RAM Model and the Growth of Functions

Antonio Carzaniga

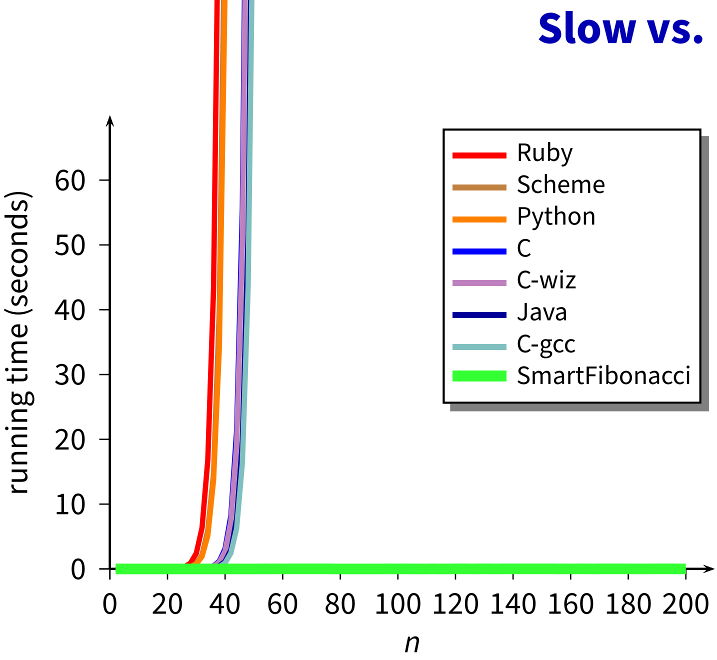
Faculty of Informatics
Università della Svizzera italiana

February 24, 2022

Outline

- Informal analysis of two Fibonacci algorithms
- The *random-access machine* model
- Measure of complexity
- Characterizing functions with their asymptotic behavior
- Big-*O*, omega, and theta notations

Slow vs. Fast Fibonacci



Slow vs. Fast Fibonacci

- We informally characterized our two Fibonacci algorithms

Slow vs. Fast Fibonacci

- We informally characterized our two Fibonacci algorithms
  - **Fibonacci**(n) is ***exponential*** in n
  - **SmartFibonacci**(n) is (almost) ***linear*** in n

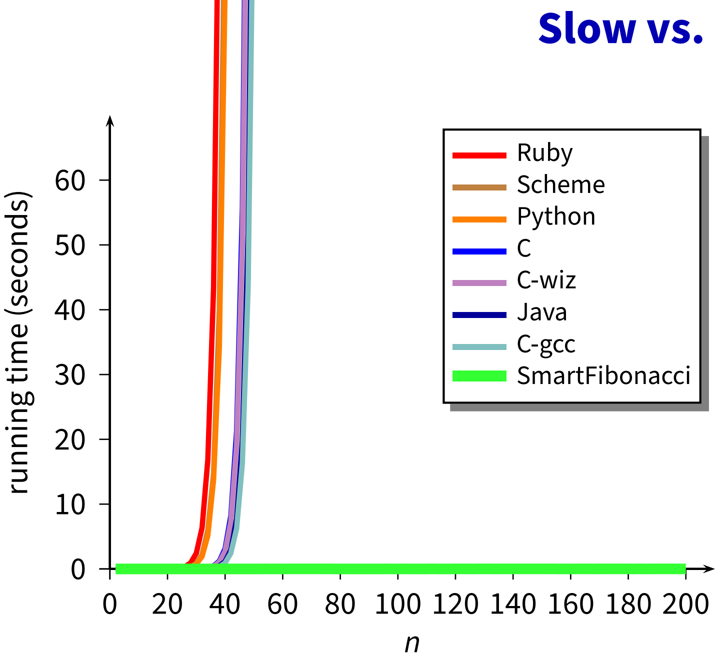
Slow vs. Fast Fibonacci

- We informally characterized our two Fibonacci algorithms
  - **FIBONACCI**(n) is ***exponential*** in n
  - **SMARTFIBONACCI**(n) is (almost) ***linear*** in n
- How do we characterize the complexity of algorithms?
  - in general

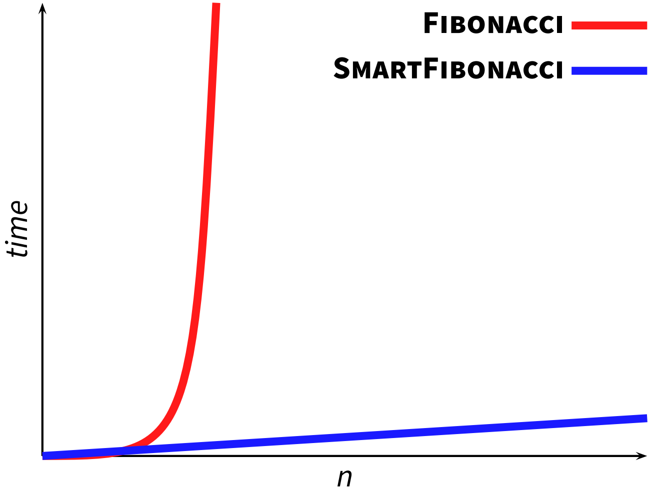
Slow vs. Fast Fibonacci

- We informally characterized our two Fibonacci algorithms
  - **Fibonacci**(n) is ***exponential*** in n
  - **SmartFibonacci**(n) is (almost) ***linear*** in n
- How do we characterize the complexity of algorithms?
  - in general
  - in a way that is ***specific to the algorithms***
  - but ***independent of implementation details***

Slow vs. Fast Fibonacci



Slow vs. Fast Fibonacci



A Model of the Computer

- An informal model of the ***random-access machine (RAM)***

A Model of the Computer

- An informal model of the ***random-access machine (RAM)***
- ***Basic types*** in the RAM model

A Model of the Computer

- An informal model of the ***random-access machine (RAM)***
- ***Basic types*** in the RAM model
  - integer and floating-point numbers
  - limited size of each "word" of data (e.g., 64 bits)

A Model of the Computer

- An informal model of the ***random-access machine (RAM)***
- ***Basic types*** in the RAM model
  - integer and floating-point numbers
  - limited size of each "word" of data (e.g., 64 bits)
- ***Basic steps*** in the RAM model

A Model of the Computer

- An informal model of the ***random-access machine (RAM)***

- ***Basic types*** in the RAM model
  - integer and floating-point numbers
  - limited size of each "word" of data (e.g., 64 bits)

- ***Basic steps*** in the RAM model
  - ***operations involving basic types***
  - load/store: assignment, use of a variable
  - arithmetic operations: addition, multiplication, division, etc.
  - branch operations: conditional branch, jump
  - subroutine call

A Model of the Computer

- An informal model of the ***random-access machine (RAM)***
- ***Basic types*** in the RAM model
  - integer and floating-point numbers
  - limited size of each "word" of data (e.g., 64 bits)
- ***Basic steps*** in the RAM model
  - ***operations involving basic types***
  - load/store: assignment, use of a variable
  - arithmetic operations: addition, multiplication, division, etc.
  - branch operations: conditional branch, jump
  - subroutine call
- A ***basic step*** in the RAM model takes a ***constant time***

Analysis in the RAM Model

```
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
```

Analysis in the RAM Model

```
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
```

cost	*times* $(n > 1)$

Analysis in the RAM Model

```
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
```

cost	*times* $(n > 1)$
c_1	1
c_2	0
c_3	1
c_4	0
c_5	1
c_6	1
c_7	n
c_8	$n-1$
c_9	$n-1$
c_{10}	$n-1$
c_{11}	1

$$T(n) = c_1 + c_3 + c_5 + c_6 + c_{11} + nc_7 + (n-1)(c_8 + c_9 + c_{10})$$

```
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
```

cost	*times* $(n > 1)$
c_1	1
c_2	0
c_3	1
c_4	0
c_5	1
c_6	1
c_7	n
c_8	$n-1$
c_9	$n-1$
c_{10}	$n-1$
c_{11}	1

$T(n) = nC_1 + C_2 \quad \Rightarrow T(n)$ is a *linear function* of n

Input Size

- We measure the complexity of an algorithm *as a function of the* ***size*** *of the input*
  - size measured in bits

Input Size

- We measure the complexity of an algorithm *as a function of the* ***size*** *of the input*
  - size measured in bits
  - did we do that for **SmartFibonacci**?

Input Size

- We measure the complexity of an algorithm *as a function of the* ***size*** *of the input*
  - size measured in bits
  - did we do that for **SmartFibonacci**?

- **Example:** given a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$, and a value x, output TRUE if A contains x, or FALSE otherwise

Input Size

- We measure the complexity of an algorithm *as a function of the **size** of the input*
  - size measured in bits
  - did we do that for **SmartFibonacci**?

- **Example:** given a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$, and a value x, output TRUE if A contains x, or FALSE otherwise

```
FIND(A, x)
1  for i = 1 to length(A)
2      if A[i] == x
3          return TRUE
4  return FALSE
```

Input Size

- We measure the complexity of an algorithm *as a function of the* ***size*** *of the input*
  - size measured in bits
  - did we do that for **SmartFibonacci**?

- **Example:** given a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$, and a value x, output TRUE if A contains x, or FALSE otherwise

```
FIND(A, x)
for i = 1 to length(A)
        if A[i] == x
                return TRUE
return FALSE
```

$$T(n) = Cn$$

Worst-Case Complexity

- In general we measure the complexity of an algorithm ***in the worst case***

Worst-Case Complexity

- In general we measure the complexity of an algorithm ***in the worst case***

- **Example:** given a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$, output TRUE if A contains two equal values $a_i = a_j$ (with $i \neq j$)

Worst-Case Complexity

- In general we measure the complexity of an algorithm ***in the worst case***

- **Example:** given a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$, output TRUE if A contains two equal values $a_i = a_j$ (with $i \neq j$)

```
FindEquals(A)
for i = 1 to length(A) − 1
    for j = i + 1 to length(A)
        if A[i] == A[j]
            return TRUE
return FALSE
```

Worst-Case Complexity

- In general we measure the complexity of an algorithm ***in the worst case***

- **Example:** given a sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$, output TRUE if A contains two equal values $a_i = a_j$ (with $i \neq j$)

```
FindEquals(A)
for i = 1 to length(A) − 1
    for j = i + 1 to length(A)
        if A[i] == A[j]
            return TRUE
return FALSE
```

$$T(n) = C\frac{n(n-1)}{2}$$

Constant Factors

- Does a load/store operation cost more than, say, an arithmetic operation?

$$x = 0 \quad \text{vs.} \quad y + z$$

Constant Factors

- Does a load/store operation cost more than, say, an arithmetic operation?

  $x = 0$ vs. $y + z$

- ***We do not care about the specific costs of each basic step***
  - these costs are likely to vary significantly with languages, implementations, and processors
  - so, we assume $c_1 = c_2 = c_3 = \cdots = c_i$

Constant Factors

- Does a load/store operation cost more than, say, an arithmetic operation?

  $x = 0$ vs. $y + z$

- ***We do not care about the specific costs of each basic step***
  - these costs are likely to vary significantly with languages, implementations, and processors
  - so, we assume $c_1 = c_2 = c_3 = \cdots = c_i$
  - we also ignore the specific *value* c_i, and in fact ***we ignore every constant cost factor***

Order of Growth

- We care only about the ***order of growth*** or *rate of growth* of $T(n)$

Order of Growth

- We care only about the ***order of growth*** or *rate of growth* of $T(n)$
  - so we ignore lower-order terms

    E.g., in

    $$T(n) = an^2 + bn + c$$

    we only consider the n^2 term and say that $T(n)$ is a quadratic function in n

Order of Growth

- We care only about the ***order of growth*** or *rate of growth* of $T(n)$
  - so we ignore lower-order terms

    E.g., in

    $$T(n) = an^2 + bn + c$$

    we only consider the n^2 term and say that $T(n)$ is a quadratic function in n

    We write

    $$\boxed{T(n) = \Theta(n^2)}$$

    and say that "$T(n)$ is theta of n-squared"

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \le 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most** c*

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) =$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) =$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) = A(14)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most** c*

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) =$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52) = 200 + A(100)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \leq 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52) = 200 + A(100)$
  - $A(n-1) = A(n^2)$

Don Knuth's A-notation

- Let $A(c)$ indicate a quantity that is ***absolutely at most*** c

  Example: $x = A(2)$ means that $|x| \le 2$

- When $x = A(y)$ we say that "x is absolutely at most y"
  - **warning:** this does not mean that x *equals* $A(y)$!
  - $A(y)$ denotes ***a set of values***
  - $x = A(y)$ really means $x \in A(y)$

- Calculating with the A notation
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$, but $x = A(4) \not\Rightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52) = 200 + A(100)$
  - $A(n-1) = A(n^2)$ for all n

- If $f(n)$ is such that $f(n) = kA(g(n))$ for all n sufficiently large and for some constant $k > 0$, 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 $g(n)$"

- If $f(n)$ is such that $f(n) = kA(g(n))$ for all n sufficiently large and for some constant $k > 0$, 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 $g(n)$"

  Examples:

  - $3n + 2 = O(n)$

- If $f(n)$ is such that $f(n) = kA(g(n))$ for all n sufficiently large and for some constant $k > 0$, 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 $g(n)$"

Examples:

  - $3n + 2 = O(n)$
  - $2\sqrt{n} + \log n = O(n^2)$

- If $f(n)$ is such that $f(n) = kA(g(n))$ for all n sufficiently large and for some constant $k > 0$, 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 $g(n)$"

Examples:

  - $3n + 2 = O(n)$
  - $2\sqrt{n} + \log n = O(n^2)$
  - let $T_{SF}(n)$ be the computational complexity of **SmartFibonacci** (the efficient algorithm); then

- If $f(n)$ is such that $f(n) = kA(g(n))$ for all n sufficiently large and for some constant $k > 0$, 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 $g(n)$"

Examples:

  - $3n + 2 = O(n)$
  - $2\sqrt{n} + \log n = O(n^2)$
  - let $T_{SF}(n)$ be the computational complexity of **SmartFibonacci** (the efficient algorithm); then

$$T_{SF}(n) = O(n)$$

From O to Ω and Θ

- If $f(n) = O(g(n))$ then we can also say that $g(n)$ asymptotically *dominates* $f(n)$, which we can also write as

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

  - which we read as "$f(n)$ is big-omega of $g(n)$" of simply "$f(n)$ is omega of $g(n)$"

- If $f(n) = O(g(n))$ then we can also say that $g(n)$ asymptotically *dominates* $f(n)$, which we can also write as

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

  - which we read as “$f(n)$ is big-omega of $g(n)$” of simply “$f(n)$ is omega of $g(n)$”

Examples:

  - $3n + 2 = \Omega(\log n)$

- If $f(n) = O(g(n))$ then we can also say that $g(n)$ asymptotically *dominates* $f(n)$, which we can also write as

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

  - which we read as "$f(n)$ is big-omega of $g(n)$" of simply "$f(n)$ is omega of $g(n)$"

Examples:

  - $3n + 2 = \Omega(\log n)$
  - let $T_F(n)$ be the computational complexity of **Fibonacci** (the inefficient algorithm); then

$$T_F(n) = \Omega((1.4)^n)$$

From O to Ω and Θ

- If $f(n) = O(g(n))$ then we can also say that $g(n)$ asymptotically *dominates* $f(n)$, which we can also write as

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

  - which we read as "$f(n)$ is big-omega of $g(n)$" of simply "$f(n)$ is omega of $g(n)$"

  Examples:

  - $3n + 2 = \Omega(\log n)$
  - let $T_F(n)$ be the computational complexity of **Fibonacci** (the inefficient algorithm); then

$$T_F(n) = \Omega((1.4)^n)$$

- When $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ we also write

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

Characterizing *Unknown* Functions

- The idea of the O, Ω, and Θ notations is very often to characterize a function that is *not completely known*

Characterizing *Unknown* Functions

- The idea of the O, Ω, and Θ notations is very often to characterize a function that is *not completely known*

  Example:

  Let $\pi(n)$ be the number of *primes* less than or equal to n

  What is the asymptotic behavior of $\pi(n)$?

Characterizing *Unknown* Functions

- The idea of the O, Ω, and Θ notations is very often to characterize a function that is *not completely known*

  Example:

  Let $\pi(n)$ be the number of *primes* less than or equal to n

  What is the asymptotic behavior of $\pi(n)$?

  - $\pi(n) = O(n)$ trivial ***upper bound***

Characterizing *Unknown* Functions

- The idea of the O, Ω, and Θ notations is very often to characterize a function that is *not completely known*

  Example:

  Let $\pi(n)$ be the number of *primes* less than or equal to n

  What is the asymptotic behavior of $\pi(n)$?

  - $\pi(n) = O(n)$ trivial ***upper bound***
  - $\pi(n) = \Omega(1)$ trivial ***lower bound***

Characterizing *Unknown* Functions

- The idea of the O, Ω, and Θ notations is very often to characterize a function that is *not completely known*

  Example:

  Let $\pi(n)$ be the number of *primes* less than or equal to n

  What is the asymptotic behavior of $\pi(n)$?

  - $\pi(n) = O(n)$ — trivial ***upper bound***
  - $\pi(n) = \Omega(1)$ — trivial ***lower bound***
  - $\pi(n) = \Theta(n/\log n)$ — non-trivial ***tight bound***

Characterizing *Unknown* Functions

- The idea of the O, Ω, and Θ notations is very often to characterize a function that is *not completely known*

  Example:

  Let $\pi(n)$ be the number of *primes* less than or equal to n

  What is the asymptotic behavior of $\pi(n)$?

  - $\pi(n) = O(n)$ — trivial ***upper bound***
  - $\pi(n) = \Omega(1)$ — trivial ***lower bound***
  - $\pi(n) = \Theta(n/\log n)$ — non-trivial ***tight bound***

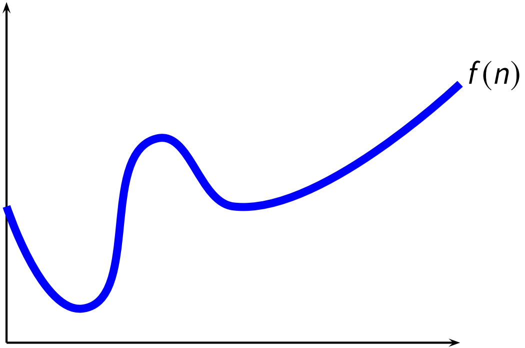
  In fact, the fundamental *prime number theorem* says that

$$\lim_{n \to \infty} \frac{\pi(n) \ln n}{n} = 1$$

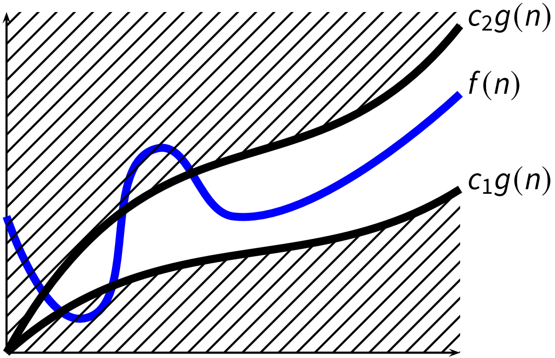
Θ-Notation

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

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

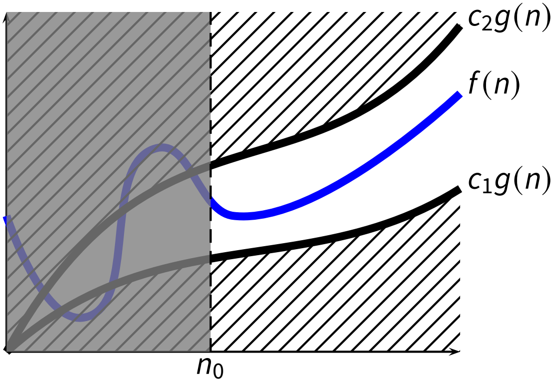


- Given a function $g(n)$, we define the *family of functions* $\Theta(g(n))$



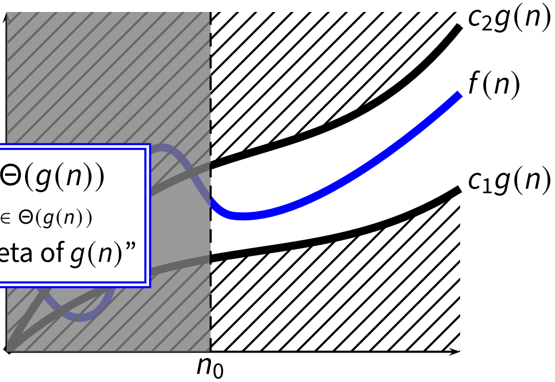
Θ-Notation

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



$$\begin{aligned}\Theta(g(n)) = \{f(n) &: \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0 \\ &: 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}\end{aligned}$$

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



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

i.e., $f(n) \in \Theta(g(n))$

"$f(n)$ is theta of $g(n)$"

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

Examples

- $T(n) = n^2 + 10n + 100$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Longrightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n}$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10\log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n\log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10\log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n\log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$
- $T(n) = \frac{10+n}{n^2}$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$
- $T(n) = \frac{10+n}{n^2} \quad \Rightarrow T(n) = \Theta(\frac{1}{n})$

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$
- $T(n) = \frac{10+n}{n^2} \quad \Rightarrow T(n) = \Theta(\frac{1}{n})$
- $T(n) =$ complexity of **SmartFibonacci**

Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$
- $T(n) = \frac{10+n}{n^2} \quad \Rightarrow T(n) = \Theta(\frac{1}{n})$
- $T(n) =$ complexity of **SmartFibonacci** $\quad \Rightarrow T(n) = \Theta(n)$

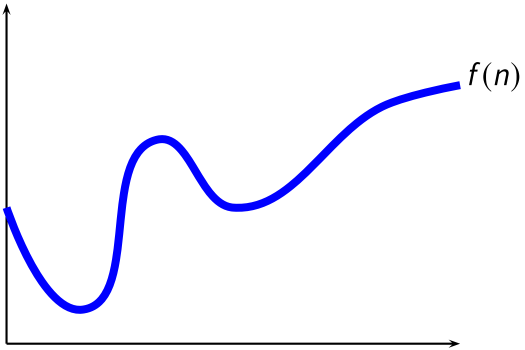
Examples

- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$
- $T(n) = \frac{10+n}{n^2} \quad \Rightarrow T(n) = \Theta(\frac{1}{n})$
- $T(n) =$ complexity of **SmartFibonacci** $\quad \Rightarrow T(n) = \Theta(n)$
- We characterize the behavior of $T(n)$ *in the limit*
- The Θ-notation is an ***asymptotic notation***

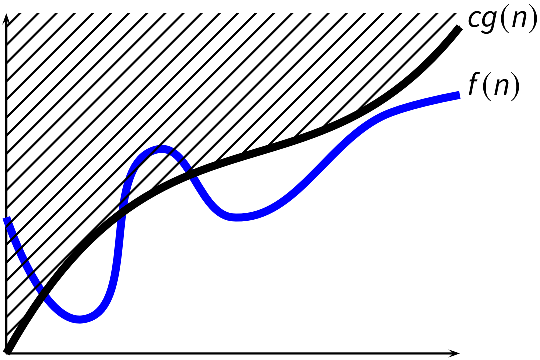
O-Notation

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

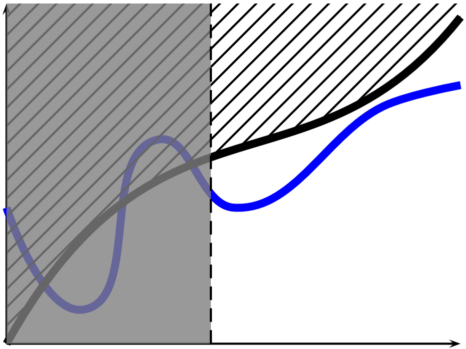
- Given a function $g(n)$, we define the *family of functions* $O(g(n))$



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

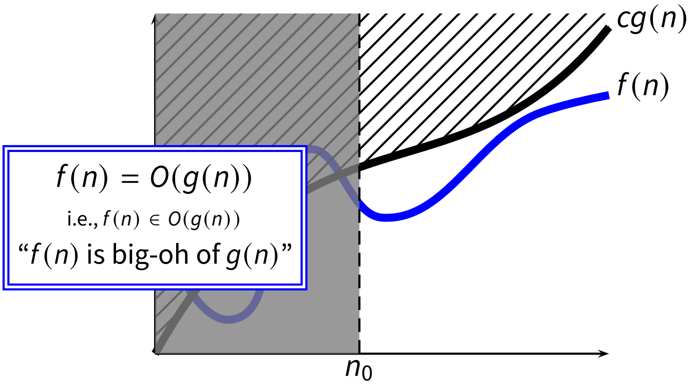


- Given a function $g(n)$, we define the *family of functions* $O(g(n))$



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

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



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

Examples

- $f(n) = n^2 + 10n + 100$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Longrightarrow f(n) = O(n^2)$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Longrightarrow f(n) = O(n^2) \quad \Longrightarrow f(n) = O(n^3)$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10 \log n$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10 \log n \quad \Rightarrow f(n) = O(2^n)$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10 \log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n \log n + n\sqrt{n}$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10 \log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n \log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10\log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n\log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10\log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n\log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow f(n) = O((1.5)^n)$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10\log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n\log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow f(n) = O((1.5)^n)$
- $f(n) = \frac{10+n}{n^2}$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10\log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n\log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow f(n) = O((1.5)^n)$
- $f(n) = \frac{10+n}{n^2} \quad \Rightarrow f(n) = O(1)$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10\log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n\log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow f(n) = O((1.5)^n)$
- $f(n) = \frac{10+n}{n^2} \quad \Rightarrow f(n) = O(1)$
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10\log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n\log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow f(n) = O((1.5)^n)$
- $f(n) = \frac{10+n}{n^2} \quad \Rightarrow f(n) = O(1)$
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$
- $f(n) = \Theta(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

Examples

- $f(n) = n^2 + 10n + 100 \quad \Rightarrow f(n) = O(n^2) \quad \Rightarrow f(n) = O(n^3)$
- $f(n) = n + 10 \log n \quad \Rightarrow f(n) = O(2^n)$
- $f(n) = n \log n + n\sqrt{n} \quad \Rightarrow f(n) = O(n^2)$
- $f(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow f(n) = O((1.5)^n)$
- $f(n) = \frac{10+n}{n^2} \quad \Rightarrow f(n) = O(1)$
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$
- $f(n) = \Theta(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
- $f(n) = O(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = O(h(n))$

Examples

- $n^2 - 10n + 100 = O(n \log n)$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$? NO

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$? NO
- $\sqrt{n} = O(\log^2 n)$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$? NO
- $\sqrt{n} = O(\log^2 n)$? NO

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$? NO
- $\sqrt{n} = O(\log^2 n)$? NO
- $n^2 + (1.5)^n = O(2^{\frac{n}{2}})$?

Examples

- $n^2 - 10n + 100 = O(n \log n)$? NO
- $f(n) = O(2^n) \Rightarrow f(n) = O(n^2)$? NO
- $f(n) = \Theta(2^n) \Rightarrow f(n) = O(n^2 2^n)$? YES
- $f(n) = \Theta(n^2 2^n) \Rightarrow f(n) = O(2^{n+2\log_2 n})$? YES
- $f(n) = O(2^n) \Rightarrow f(n) = \Theta(n^2)$? NO
- $\sqrt{n} = O(\log^2 n)$? NO
- $n^2 + (1.5)^n = O(2^{\frac{n}{2}})$? NO

Example

- So, what is the complexity of **FindEquals**?

```
FindEquals(A)
for i = 1 to length(A) − 1
        for j = i + 1 to length(A)
                if A[i] == A[j]
                        return TRUE
return FALSE
```

Example

- So, what is the complexity of **FINDEQUALS**?

```
FINDEQUALS(A)
for i = 1 to length(A) − 1
    for j = i + 1 to length(A)
        if A[i] == A[j]
            return TRUE
return FALSE
```

$$T(n) = \Theta(n^2)$$

- $n = length(A)$ is the ***size of the input***
- we measure the ***worst-case complexity***

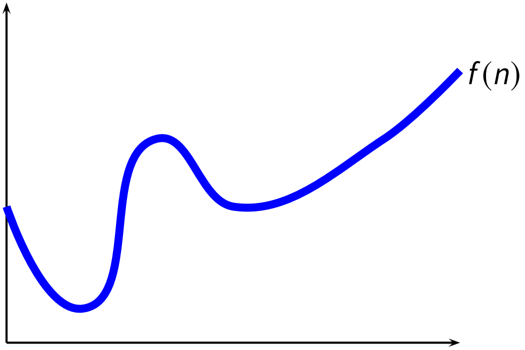
Ω-Notation

Ω-Notation

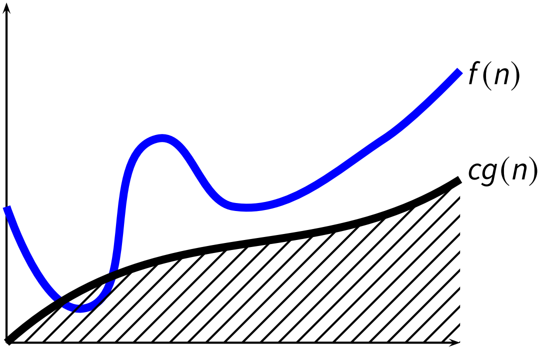
- Given a function $g(n)$, we define the *family of functions* $\Omega(g(n))$

Ω-Notation

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

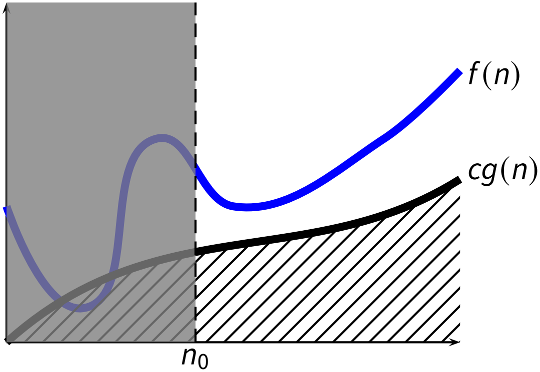


- Given a function $g(n)$, we define the *family of functions* $\Omega(g(n))$



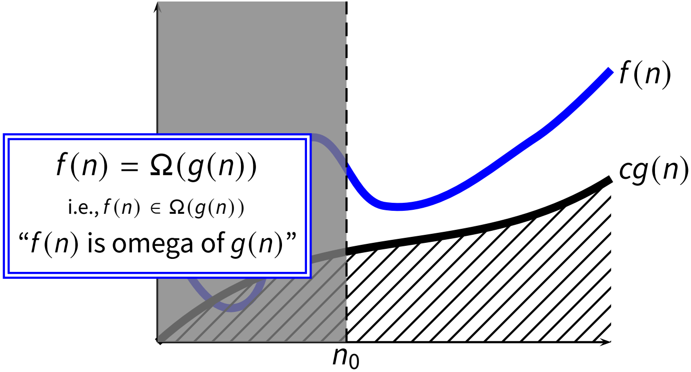
Ω-Notation

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



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

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



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

Θ, O, and Ω as Relations

- *Theorem:* for any two functions $f(n)$ and $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

Θ, O, and Ω as Relations

- *Theorem:* for any two functions $f(n)$ and $g(n)$,
  $f(n) = \Omega(g(n)) \land f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$
- The Θ-notation, Ω-notation, and O-notation can be viewed as the "asymptotic" $=$, $\geq$, and $\leq$ relations for functions, respectively

Θ, O, and Ω as Relations

- *Theorem:* for any two functions $f(n)$ and $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- The Θ-notation, Ω-notation, and O-notation can be viewed as the "asymptotic" $=$, $\geq$, and $\leq$ relations for functions, respectively

- The above theorem can be interpreted as saying

$$f \geq g \wedge f \leq g \Leftrightarrow f = g$$

Θ, O, and Ω as Relations

- *Theorem:* for any two functions $f(n)$ and $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- The Θ-notation, Ω-notation, and O-notation can be viewed as the "asymptotic" $=$, $\geq$, and $\leq$ relations for functions, respectively

- The above theorem 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 $f(n)$, and that $g(n)$ ***dominates*** $f(n)$

Θ, O, and Ω as Relations

- *Theorem:* for any two functions $f(n)$ and $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- The Θ-notation, Ω-notation, and O-notation can be viewed as the "asymptotic" $=$, $\geq$, and $\leq$ relations for functions, respectively

- The above theorem 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 $f(n)$, and that $g(n)$ ***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-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$?

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

  $n^2 + \Omega(n) - 1 = O(n^2)$?

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

$$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

  $n^2 + \Omega(n) - 1 = O(n^2)$? NO

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

  $n^2 + \Omega(n) - 1 = O(n^2)$? NO

  $n^2 + O(n) - 1 = O(n^2)$?

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

  $n^2 + \Omega(n) - 1 = O(n^2)$? NO

  $n^2 + O(n) - 1 = O(n^2)$? YES

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

  $$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

  $n^2 + \Omega(n) - 1 = O(n^2)$? NO

  $n^2 + O(n) - 1 = O(n^2)$? YES

  $n \log n + \Theta(\sqrt{n}) = O(n\sqrt{n})$?

Θ, O, and Ω as Anonymous Functions

- We can use the Θ-, O-, and Ω-notation to represent anonymous (unknown or unsecified) functions
  E.g.,

$$f(n) = 10n^2 + O(n)$$

  means that $f(n)$ is equal to $10n^2$ plus a function we don't know or we don't care to know that is asymptotically at most linear in n.

- Examples

  $n^2 + 4n - 1 = n^2 + \Theta(n)$? YES

  $n^2 + \Omega(n) - 1 = O(n^2)$? NO

  $n^2 + O(n) - 1 = O(n^2)$? YES

  $n \log n + \Theta(\sqrt{n}) = O(n\sqrt{n})$? YES

o-Notation

o-Notation

- The O-notation defines an upper bound that might not be ***asymptotically tight***

o-Notation

- The O-notation defines an upper bound that might not be ***asymptotically tight***

  E.g.,

  $n \log n = O(n^2)$ is not asymptotically tight

  $n^2 - n + 10 = O(n^2)$ is asymptotically tight

o-Notation

- The O-notation defines an upper bound that might not be ***asymptotically tight***

  E.g.,

  $n \log n = O(n^2)$ is not asymptotically tight

  $n^2 - n + 10 = O(n^2)$ is asymptotically tight

- We use the o-notation to denote upper bounds that are *not* asymtotically tight. So, given a function $g(n)$, we define the family of functions $o(g(n))$

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

ω-Notation

ω-Notation

- The Ω-notation defines a lower bound that might not be *asymptotically tight*

ω-Notation

- The Ω-notation defines a lower bound that might not be *asymptotically tight*

  E.g.,

  $2^n = \Omega(n \log n)$ is not asymptotically tight

  $n + 4n \log n = \Omega(n \log n)$ is asymptotically tight

ω-Notation

- The Ω-notation defines a lower bound that might not be *asymptotically tight*

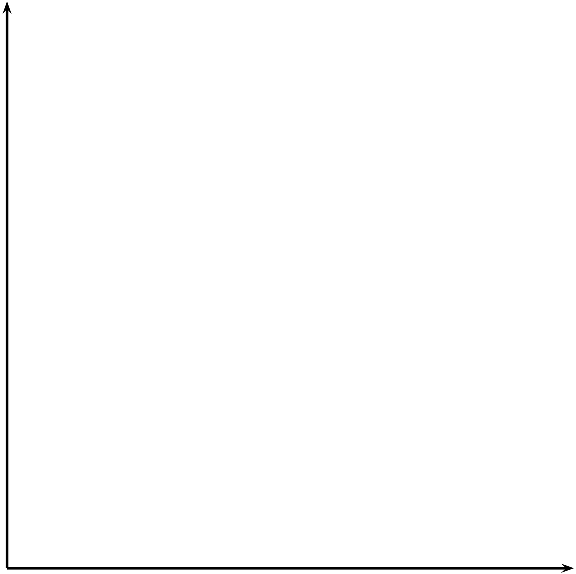
  E.g.,

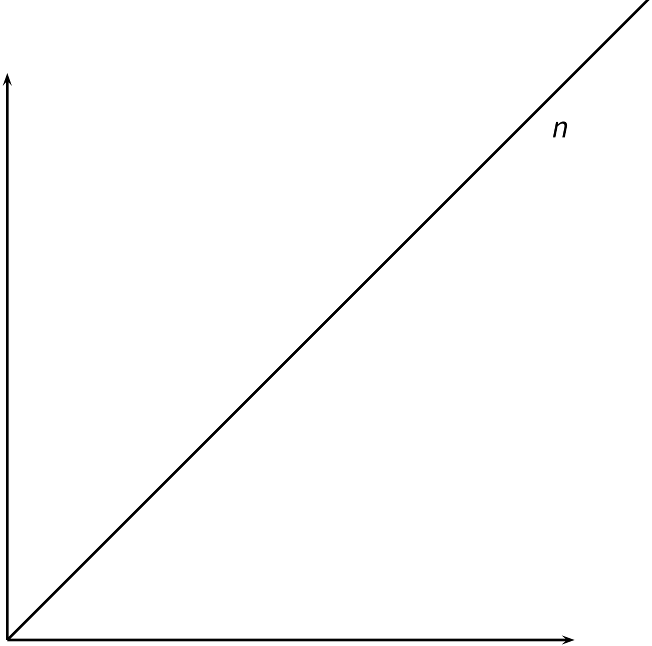
  $2^n = \Omega(n \log n)$ is not asymptotically tight

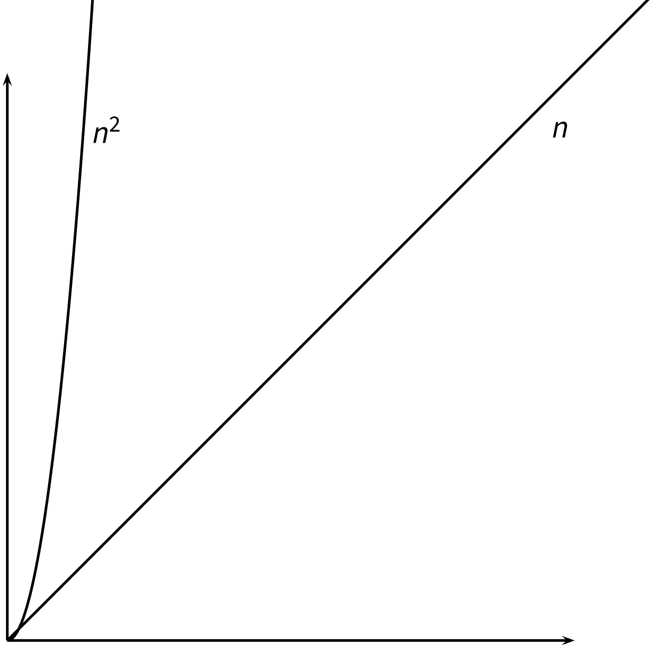
  $n + 4n \log n = \Omega(n \log n)$ is asymptotically tight

- We use the ω-notation to denote lower bounds that are *not* asymtotically tight. So, given a function $g(n)$, we define the family of functions $\omega(g(n))$

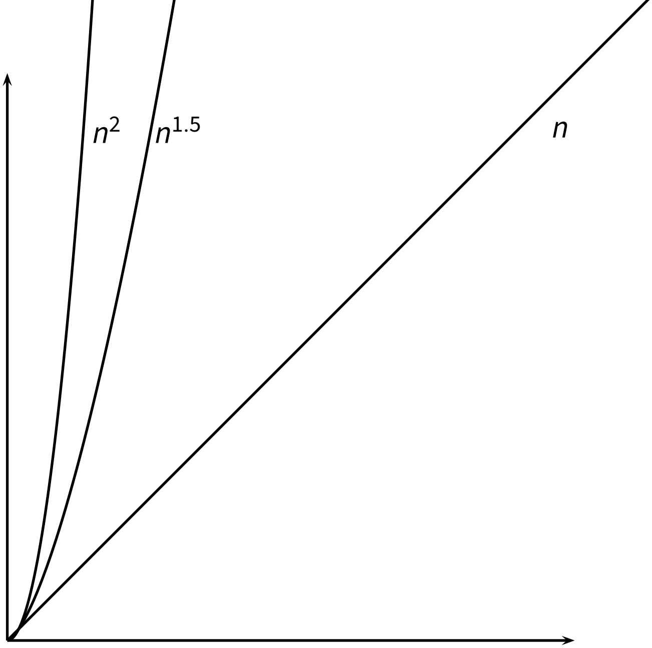
$$\omega(g(n)) = \{f(n) : \forall c > 0, \exists n_0 > 0 \\ : 0 \leq cg(n) < f(n) \text{ for all } n \geq n_0\}$$



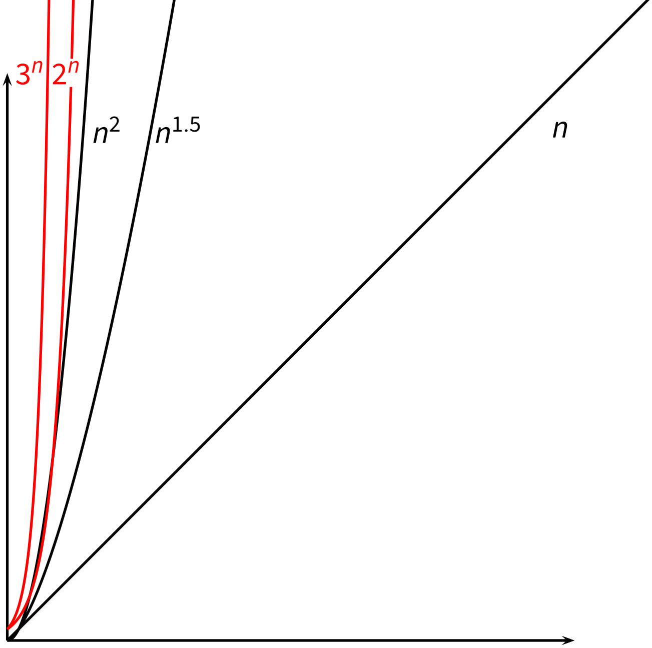

n



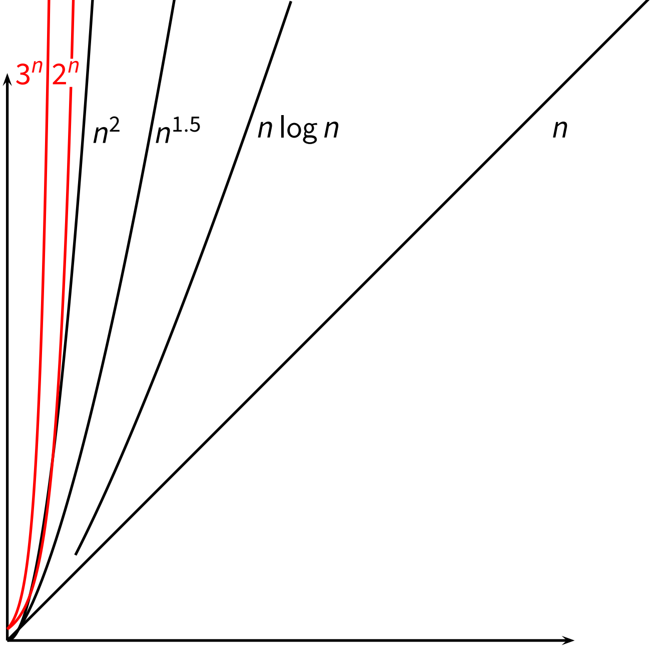
n^2
n



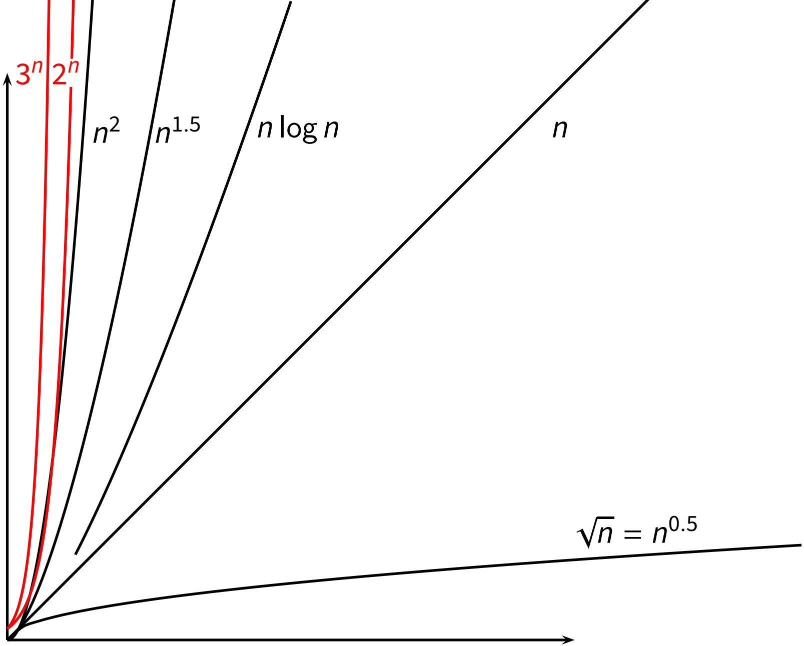
n^2
$n^{1.5}$
n



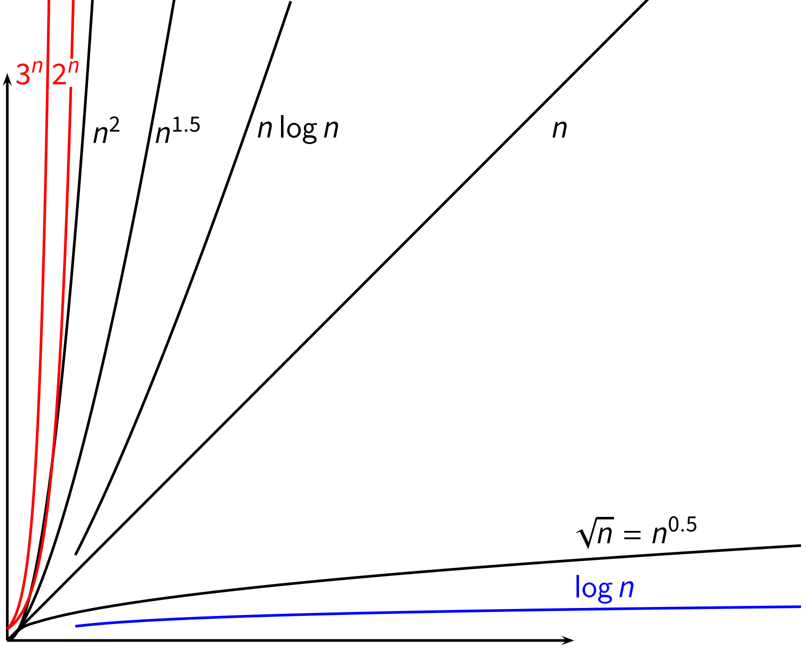
3^n
2^n
n^2
n^1.5
n



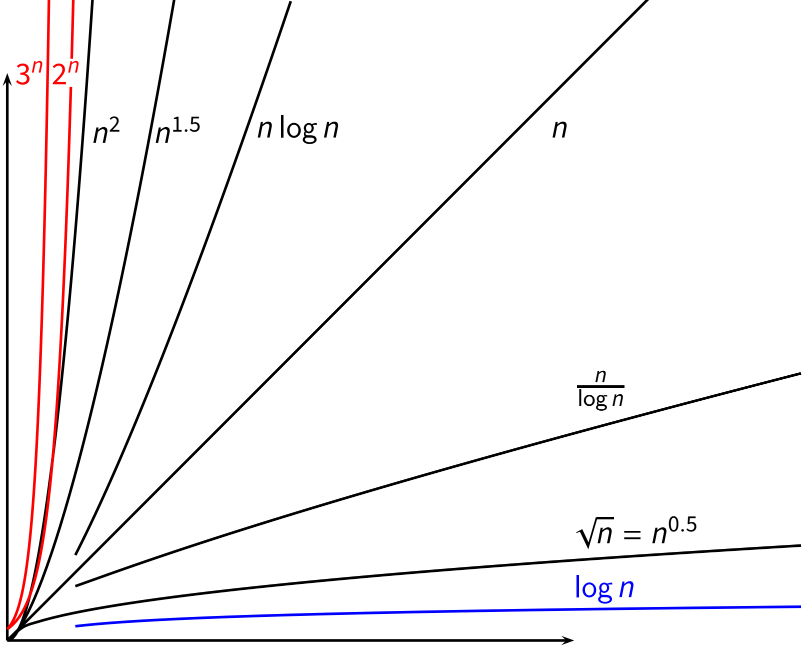
3^n
2^n
n^2
$n^{1.5}$
$n \log n$
n



3^n
2^n
n^2
n^1.5
n log n
n
√n = n^0.5



3^n
2^n
n^2
$n^{1.5}$
$n \log n$
n
$\sqrt{n} = n^{0.5}$
$\log n$



3^n
2^n
n^2
$n^{1.5}$
$n \log n$
n
$\frac{n}{\log n}$
$\sqrt{n} = n^{0.5}$
$\log n$