Modelo en memoria RAM
Funciones de Crecimiento

José Ortiz Bejar

Facultad de Ingeniería Eléctrica
Universidad Michoacana de San Nicolás de Hidalgo

February 20, 2020

Análisis informal de las dos implementaciones de Fibonacci

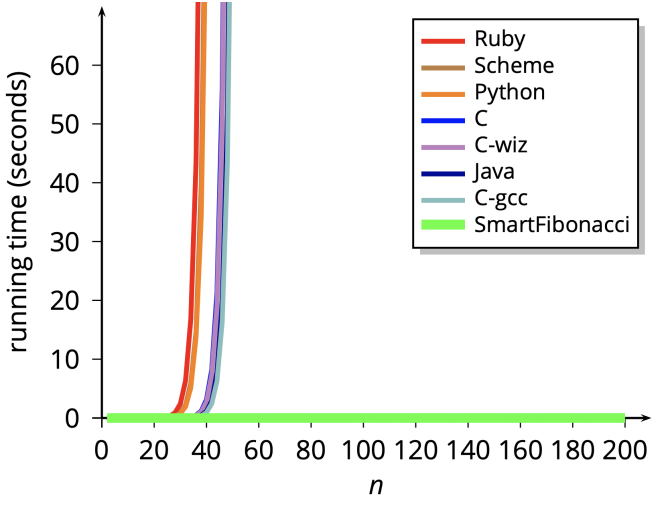
Modelo en RAM

Complejidad computacional

Caracterización asintótica

Notaciones o, omega y theta

Lento vs. Rápido



Lento vs. Rápido

- Informalmente determinamos las funciones de creceimiento de ambas implementaciones

Lento vs. Rápido

- Informalmente determinamos las funciones de creceimiento de ambas implementaciones
  - **Fibonacci**(n) es ***exponencial*** con respecto de n
  - **SmartFibonacci**(n) es (casi) ***lineal*** en función de n

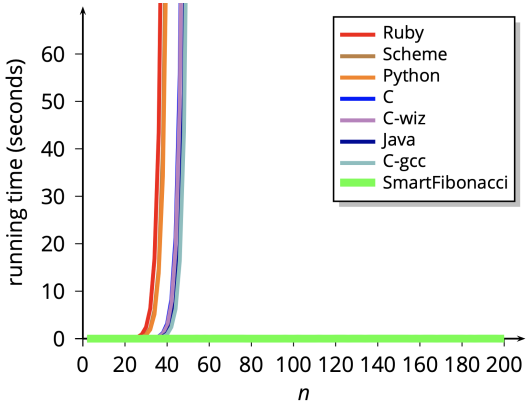
Lento vs. Rápido

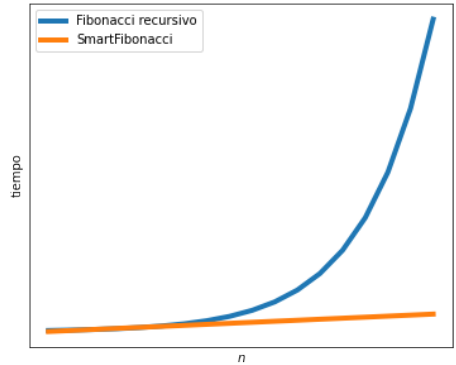
- Informalmente determinamos las funciones de creceimiento de ambas implementaciones
  - **Fibonacci**(n) es ***exponencial*** con respecto de n
  - **SmartFibonacci**(n) es (casi) ***lineal*** en función de n
- ¿Cómo podemos caracterizar la complejidad de los algoritmos?
  - en general

Lento vs. Rápido

- Informalmente determinamos las funciones de creceimiento de ambas implementaciones
  - **Fibonacci**(n) es ***exponencial*** con respecto de n
  - **SmartFibonacci**(n) es (casi) ***lineal*** en función de n
- ¿Cómo podemos caracterizar la complejidad de los algoritmos?
  - en general
  - de una forma ***específica para algoritmos***
  - pero ***independiente de los la implementación***

Modelo *Real* vs. *asintótico*





Modelo Computacional

- Un modelo informal ***random-access machine (RAM)***

Modelo Computacional

- Un modelo informal ***random-access machine (RAM)***
- ***Tipos básicos*** el modelo RAM

Modelo Computacional

- Un modelo informal ***random-access machine (RAM)***
- ***Tipos básicos*** el modelo RAM
  - numero enteros y de punto flotante
  - tamaño de palabra “word” (e.j., 64 bits)

Modelo Computacional

- Un modelo informal ***random-access machine (RAM)***
- ***Tipos básicos*** el modelo RAM
  - numero enteros y de punto flotante
  - tamaño de palabra “word” (e.j., 64 bits)
- ***Operaciones básicas*** en el modelo RAM

Modelo Computacional

- Un modelo informal ***random-access machine (RAM)***

- ***Tipos básicos*** el modelo RAM
  - numero enteros y de punto flotante
  - tamaño de palabra “word” (e.j., 64 bits)

- ***Operaciones básicas*** en el modelo RAM
  - ***operaciones que utilizan tipos básicos***
  - lectura/escritura: asignación (uso de variables)
  - operaciones aritméticas: suma, multiplicación, división, etc.
  - control de flujo: condicionales, saltos
  - llamadas a sub-rutinas

Modelo Computacional

- Un modelo informal ***random-access machine (RAM)***

- ***Tipos básicos*** el modelo RAM
  - numero enteros y de punto flotante
  - tamaño de palabra “word” (e.j., 64 bits)

- ***Operaciones básicas*** en el modelo RAM
  - ***operaciones que utilizan tipos básicos***
  - lectura/escritura: asignación (uso de variables)
  - operaciones aritméticas: suma, multiplicación, división, etc.
  - control de flujo: condicionales, saltos
  - llamadas a sub-rutinas

- Una ***Operación básica*** toma tiempo ***constante*** en el modelo RAM.

```
SmartFibonacci(n)  1   if n == 0
                   2       return 0
                   3   elseif n == 1
                   4       return 1
                   5   else pprev = 0
                   6       prev = 1
                   7       for i = 2 to n
                   8           f = prev + pprev
                   9           pprev = prev
                  10           prev = f
                  11   return f
```

Análisis utilizando el modelo RAM

```
SmartFibonacci(n)  1   if n == 0                          costo   operaciones (n > 1)
                   2       return 0
                   3   elseif n == 1
                   4       return 1
                   5   else pprev = 0
                   6       prev = 1
                   7       for i = 2 to n
                   8           f = prev + pprev
                   9           pprev = prev
                  10           prev = f
                  11   return f
```

Análisis utilizando el modelo RAM

```
SmartFibonacci(n)  1  if n == 0
                   2       return 0
                   3  elseif n == 1
                   4       return 1
                   5  else pprev = 0
                   6       prev = 1
                   7       for i = 2 to n
                   8            f = prev + pprev
                   9            pprev = prev
                  10            prev = f
                  11  return f
```

costo	*operaciones* $(n > 1)$
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

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

Análisis utilizando el modelo RAM

```
SmartFibonacci(n)  1  if n == 0
                   2      return 0
                   3  elseif n == 1
                   4      return 1
                   5  else pprev = 0
                   6      prev = 1
                   7      for i = 2 to n
                   8          f = prev + pprev
                   9          pprev = prev
                  10          prev = f
                  11  return f
```

costo	*operaciones* $(n > 1)$
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

$T(n) = nC_1 + C_2 \quad \Rightarrow T(n)$ es una ***función lineal*** con respecto de n

Tamaño de la entrada

- La complejidad de un algoritmo se estima *como función del* ***tamaño*** *de la entrada*
  - el tamaño en memoria es medido en bits

Tamaño de la entrada

- La complejidad de un algoritmo se estima *como función del **tamaño** de la entrada*
  - el tamaño en memoria es medido en bits
  - ¿Eso hicimos para analizar **SmartFibonacci**?

Tamaño de la entrada

- La complejidad de un algoritmo se estima *como función del **tamaño** de la entrada*
  - el tamaño en memoria es medido en bits
  - ¿Eso hicimos para analizar **SmartFibonacci**?

- **Ejemplo:** dada una secuencia $A = \langle a_1, a_2, \ldots, a_n \rangle$, y un valor x, determinar si x se encuentra en la A. Si A contiene a x regresar true, y falso sino se encuentra.

Tamaño de la entrada

- La complejidad de un algoritmo se estima *como función del* ***tamaño*** *de la entrada*
  - el tamaño en memoria es medido en bits
  - ¿Eso hicimos para analizar **SmartFibonacci**?

- **Ejemplo:** dada una secuencia $A = \langle a_1, a_2, \ldots, a_n \rangle$, y un valor x, determinar si x se encuentra en la A. Si A contiene a x regresar true, y falso sino se encuentra.

```
Find(A, x) 1   for i = 1 to length(A)
           2       if A[i] == x
           3           return true
           4   return false
```

Tamaño de la entrada

- La complejidad de un algoritmo se estima *como función del* ***tamaño*** *de la entrada*
  - el tamaño en memoria es medido en bits
  - ¿Eso hicimos para analizar **SmartFibonacci**?

- **Ejemplo:** dada una secuencia $A = \langle a_1, a_2, \ldots, a_n \rangle$, y un valor x, determinar si x se encuentra en la A. Si A contiene a x regresar true, y falso sino se encuentra.

```
Find(A, x) 1   for i = 1 to length(A)
           2       if A[i] == x
           3           return true
           4   return false
```

$$T(n) = Cn$$

Complejidad del peor caso

- En general nos interesa saber el tiempo en que el algoritmo resolverá el ***peor caso***

Complejidad del peor caso

- En general nos interesa saber el tiempo en que el algoritmo resolverá el ***peor caso***

- **Eajemplo:** dada una secuencia $A = \langle a_1, a_2, \ldots, a_n \rangle$, determinar si A contiene dos valores iguales $a_i = a_j$ (con $i \neq j$)

Complejidad del peor caso

- En general nos interesa saber el tiempo en que el algoritmo resolverá el ***peor caso***

- **Eajemplo:** dada una secuencia $A = \langle a_1, a_2, \ldots, a_n \rangle$, determinar si A contiene dos valores iguales $a_i = a_j$ (con $i \neq j$)

```
FindEquals(A) 1   for i = 1 to length(A) − 1
              2         for j = i + 1 to length(A)
              3               if A[i] == A[j]
              4                     return true
              5   return false
```

Complejidad del peor caso

- En general nos interesa saber el tiempo en que el algoritmo resolverá el ***peor caso***

- **Eajemplo:** dada una secuencia $A = \langle a_1, a_2, \ldots, a_n \rangle$, determinar si A contiene dos valores iguales $a_i = a_j$ (con $i \neq j$)

```
FindEquals(A) 1  for i = 1 to length(A) − 1
              2      for j = i + 1 to length(A)
              3          if A[i] == A[j]
              4              return true
              5  return false
```

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

Factores constantes

- ¿Una operación de lectura/escritura cuesta más que una operación aritmética?

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

Factores constantes

- ¿Una operación de lectura/escritura cuesta más que una operación aritmética?

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

- ***No nos preocupamos de costo específico de cada operación elemental***
  - estos costos pueden varias significativamente entre diferentes lenguajes, implementaciones y arquitecturas.
  - se asume que $c_1 = c_2 = c_3 = \cdots = c_i$

Factores constantes

- ¿Una operación de lectura/escritura cuesta más que una operación aritmética?

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

- ***No nos preocupamos de costo específico de cada operación elemental***
  - estos costos pueden varias significativamente entre diferentes lenguajes, implementaciones y arquitecturas.
  - se asume que $c_1 = c_2 = c_3 = \cdots = c_i$
  - también ignoramos el *valor* específico de c_i, de hecho ***se ignoran todos los costos de factor constante***

Orden de crecimiento

- Solo nos interesa el ***orden de crecimiento*** o *razón de crecimiento* de $T(n)$

Orden de crecimiento

- Solo nos interesa el ***orden de crecimiento*** o *razón de crecimiento* de $T(n)$
  - es decir ignoramos los términos de menor orden de crecimiento.

    Por ejemplo, en

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

    solo consideramos el término n^2 y decimos que $T(n)$ es cuadrática con respecto de n.

Orden de crecimiento

- Solo nos interesa el ***orden de crecimiento*** o *razón de crecimiento* de $T(n)$
  - es decir ignoramos los términos de menor orden de crecimiento.

    Por ejemplo, en

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

    solo consideramos el término n^2 y decimos que $T(n)$ es cuadrática con respecto de n.

    Lo anterior se escribe como:

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

    se lee como "$T(n)$ es theta de n-cuadradra"

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) =$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Rightarrow x = A(4)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) =$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) = A(14)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) =$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52) = 200 + A(100)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52) = 200 + A(100)$
  - $A(n-1) = A(n^2)$

Notacion A (Don Knuth's)

- Sea $A(c)$ una cantidad que su ***valor absoluto será a lo más*** c

  Ejemplo: $x = A(2)$ significa que $|x| \leq 2$

- Cuando $x = A(y)$ decimos que el valor absoluto de "x será a lo más y"
  - **warning:** esto no implica que x *es igual a* $A(y)$!
  - $A(y)$ denota ***un conjunto de valores***
  - $x = A(y)$ e verdad significa $x \in A(y)$

- Haciendo cálculos con la notación A
  - $\pi = 3.14159265\ldots = 3.14 + A(0.005)$
  - $A(3) + A(4) = A(7)$
  - $x = A(3) \Longrightarrow x = A(4)$, pero no $x = A(4) \nRightarrow x = A(3)$
  - $A(2)A(7) = A(14)$
  - $(10 + A(2))(20 + A(1)) = 200 + A(52) = 200 + A(100)$
  - $A(n-1) = A(n^2)$ para cualquier $n \geq 1$

De notación A a notación O

- Si $f(n)$ es tal que $f(n) = kA(g(n))$ para toda n suficientemente grande y alguna constante $k > 0$, entonces se dice que

$$f(n) = O(g(n))$$

  - se lee "$f(n)$ es o-grande (big-oh) de $g(n)$" o simplemente "$f(n)$ es o de $g(n)$"

De notación A a notación O

- Si $f(n)$ es tal que $f(n) = kA(g(n))$ para toda n suficientemente grande y alguna constante $k > 0$, entonces se dice que

$$f(n) = O(g(n))$$

  - se lee "$f(n)$ es o-grande (big-oh) de $g(n)$" o simplemente "$f(n)$ es o de $g(n)$"

Ejemplos:

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

De notación A a notación O

- Si $f(n)$ es tal que $f(n) = kA(g(n))$ para toda n suficientemente grande y alguna constante $k > 0$, entonces se dice que

$$f(n) = O(g(n))$$

  - se lee "$f(n)$ es o-grande (big-oh) de $g(n)$" o simplemente "$f(n)$ es o de $g(n)$"

Ejemplos:

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

De notación A a notación O

- Si $f(n)$ es tal que $f(n) = kA(g(n))$ para toda n suficientemente grande y alguna constante $k > 0$, entonces se dice que

$$f(n) = O(g(n))$$

  - se lee "$f(n)$ es o-grande (big-oh) de $g(n)$" o simplemente "$f(n)$ es o de $g(n)$"

Ejemplos:

- $3n + 2 = O(n)$
- $2\sqrt{n} + \log n = O(n^2)$
- sea $T_{SF}(n)$ la complejidad computacional del algoritmo **SmartFibonacci**; entonces

De notación A a notación O

- Si $f(n)$ es tal que $f(n) = kA(g(n))$ para toda n suficientemente grande y alguna constante $k > 0$, entonces se dice que

$$f(n) = O(g(n))$$

  - se lee "$f(n)$ es o-grande (big-oh) de $g(n)$" o simplemente "$f(n)$ es o de $g(n)$"

Ejemplos:

- $3n + 2 = O(n)$
- $2\sqrt{n} + \log n = O(n^2)$
- sea $T_{SF}(n)$ la complejidad computacional del algoritmo **SmartFibonacci**; entonces

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

- Si $f(n) = O(g(n))$ entonces podemos decir también que $g(n)$ *domina* asintóticamente a $f(n)$, lo cual se escribe como:

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

  - se lee como "$f(n)$ es omega-grande de $g(n)$" o simplemente como "$f(n)$ es omega de $g(n)$"

- Si $f(n) = O(g(n))$ entonces podemos decir también que $g(n)$ *domina* asintóticamente a $f(n)$, lo cual se escribe como:

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

  - se lee como "$f(n)$ es omega-grande de $g(n)$" o simplemente como "$f(n)$ es omega de $g(n)$"

Ejemplos:

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

- Si $f(n) = O(g(n))$ entonces podemos decir también que $g(n)$ *domina* asintóticamente a $f(n)$, lo cual se escribe como:

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

  - se lee como "$f(n)$ es omega-grande de $g(n)$" o simplemente como "$f(n)$ es omega de $g(n)$"

Ejemplos:

- $3n + 2 = \Omega(\log n)$
- sea $T_F(n)$ la complejidad de la versión recursiva de **Fibonacci**; entonces

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

- Si $f(n) = O(g(n))$ entonces podemos decir también que $g(n)$ *domina* asintóticamente a $f(n)$, lo cual se escribe como:

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

  - se lee como "$f(n)$ es omega-grande de $g(n)$" o simplemente como "$f(n)$ es omega de $g(n)$"

  Ejemplos:

  - $3n + 2 = \Omega(\log n)$
  - sea $T_F(n)$ la complejidad de la versión recursiva de **Fibonacci**; entonces

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

- Cuando $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$ se puede escribir como

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

Caracterización de funciones *Desconocidas*

- La idea de utlitzar las notaciónes O, Ω, Θ es con frecuencua caracterizar funciones que no son completamente conocidas.

Caracterización de funciones *Desconocidas*

- La idea de utlitzar las notaciónes O, Ω, Θ es con frecuencua caracterizar funciones que no son completamente conocidas.

  Ejemplo:

  Sea $\pi(n)$ el número de números *primos* menores o iguales a n

  ¿Cuál es el comportamiento asintótico de $\pi(n)$?

Caracterización de funciones *Desconocidas*

- La idea de utlitzar las notaciónes O, Ω, Θ es con frecuencua caracterizar funciones que no son completamente conocidas.

  Ejemplo:

  Sea $\pi(n)$ el número de números *primos* menores o iguales a n

  ¿Cuál es el comportamiento asintótico de $\pi(n)$?

  - trivial $\pi(n) = O(n)$ ***cota superior***

Caracterización de funciones *Desconocidas*

- La idea de utlitzar las notaciónes O, Ω, Θ es con frecuencua caracterizar funciones que no son completamente conocidas.

  Ejemplo:

  Sea $\pi(n)$ el número de números *primos* menores o iguales a n

  ¿Cuál es el comportamiento asintótico de $\pi(n)$?

  - trivial $\pi(n) = O(n)$ ***cota superior***
  - trivial $\pi(n) = \Omega(1)$ ***cota inferior***

Caracterización de funciones *Desconocidas*

- La idea de utlitzar las notaciónes O, Ω, Θ es con frecuencua caracterizar funciones que no son completamente conocidas.

  Ejemplo:

  Sea $\pi(n)$ el número de números *primos* menores o iguales a n

  ¿Cuál es el comportamiento asintótico de $\pi(n)$?

  - trivial $\pi(n) = O(n)$ ***cota superior***
  - trivial $\pi(n) = \Omega(1)$ ***cota inferior***
  - No trivial $\pi(n) = \Theta(n/\log n)$ ***cota ajustada***

Caracterización de funciones *Desconocidas*

- La idea de utlitzar las notaciónes O, Ω, Θ es con frecuencua caracterizar funciones que no son completamente conocidas.

  Ejemplo:

  Sea $\pi(n)$ el número de números *primos* menores o iguales a n

  ¿Cuál es el comportamiento asintótico de $\pi(n)$?

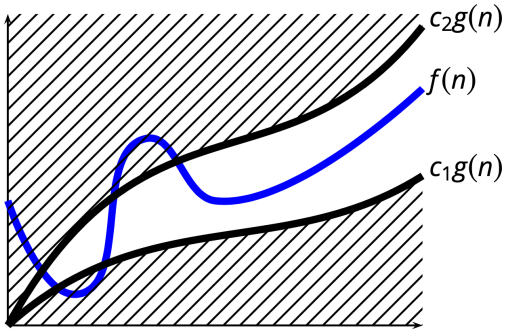
  - trivial $\pi(n) = O(n)$ ***cota superior***
  - trivial $\pi(n) = \Omega(1)$ ***cota inferior***
  - No trivial $\pi(n) = \Theta(n/\log n)$ ***cota ajustada***

  De hecho, el teorema fundamental de los *números primos* indica que:

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

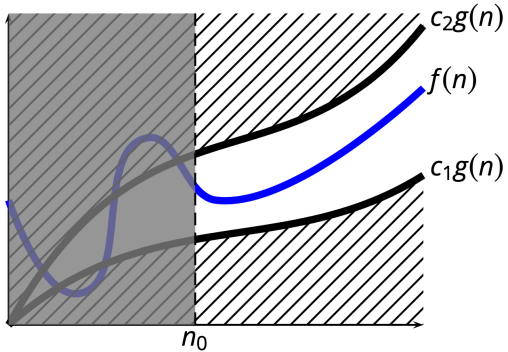
Notación Θ

- Dada una función $g(n)$, definimos la *familia de funciones* $\Theta(g(n))$



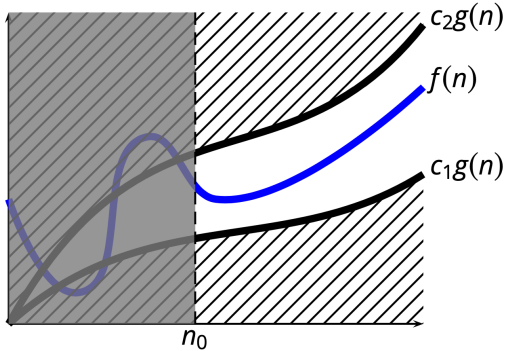
Notación Θ

- Dada una función $g(n)$, definimos la *familia de funciones* $\Theta(g(n))$



Notación Θ

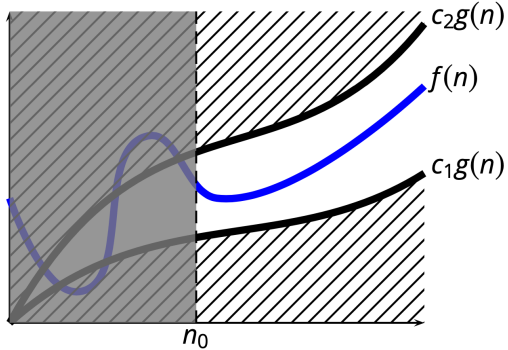
- Dada una función $g(n)$, definimos la *familia de funciones* $\Theta(g(n))$



$$\Theta(g(n)) = \{f(n) : \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para toda } n \geq n_0\}$$

Notación Θ

- Dada una función $g(n)$, definimos la *familia de funciones* $\Theta(g(n))$



$\Theta(g(n)) = \{f(n) : \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0$
$: 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para toda $n \geq n_0\}$

$f(n) = \Theta(g(n))$ se lee $f(n)$ es theta de $g(n)$

Ejemplos

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

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

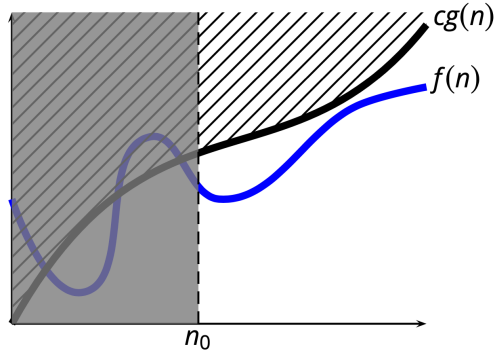
- $T(n) = n^2 + 10n + 100 \quad \Rightarrow T(n) = \Theta(n^2)$
- $T(n) = n + 10 \log n \quad \Rightarrow T(n) = \Theta(n)$
- $T(n) = n \log n + n\sqrt{n} \quad \Rightarrow T(n) = \Theta(n\sqrt{n})$
- $T(n) = 2^{\frac{n}{6}} + n^7 \quad \Rightarrow T(n) = \Theta(2^{\frac{n}{6}})$
- $T(n) = \frac{10+n}{n^2} \quad \Rightarrow T(n) = \Theta(\frac{1}{n})$
- $T(n) =$ complejidad de **SmartFibonacci** $\Rightarrow T(n) = \Theta(n)$
- Caracterizamos el comportamiento de $T(n)$ *en el limite*
- La notación Θ es una ***notación asintótica***

O-Notation

- Dada una función $g(n)$, definimos una *familia de funciones* $O(g(n))$

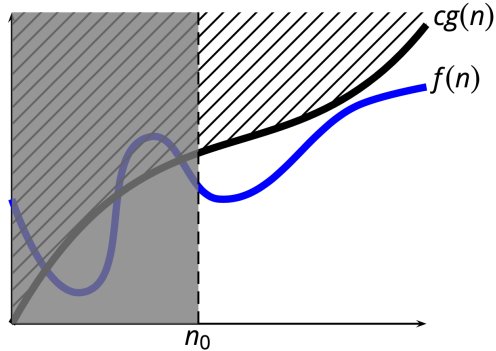
O-Notation

- Dada una función $g(n)$, definimos una *familia de funciones* $O(g(n))$



O-Notation

- Dada una función $g(n)$, definimos una *familia de funciones* $O(g(n))$



$\Theta(g(n)) = \quad f(n) \quad : \exists c > 0, \exists n_0 > 0$

$: 0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$

$f(n) = O(g(n))$ $f(n)$ es o-grande de $g(n)$

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

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

Examples

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

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

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

Ejemplos

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

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplos

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

Ejemplo

- ¿ Cual es la complejidad del método **FindEquals**?

```
FindEquals(A) 1  for i = 1 to length(A) − 1
              2      for j = i + 1 to length(A)
              3          if A[i] == A[j]
              4              return true
              5  return false
```

Ejemplo

- ¿ Cual es la complejidad del método **FindEquals**?

```
FindEquals(A) 1  for i = 1 to length(A) − 1
              2      for j = i + 1 to length(A)
              3          if A[i] == A[j]
              4              return true
              5  return false
```

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

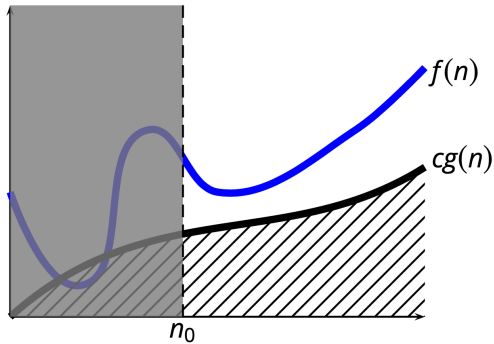
- $n = length(A)$ es el ***tamaño de la entrada***
- Caracterizamos el ***peor caso***

Notación Ω

- Dada una función $g(n)$, definimos una *familia de funciones* $\Omega(g(n))$

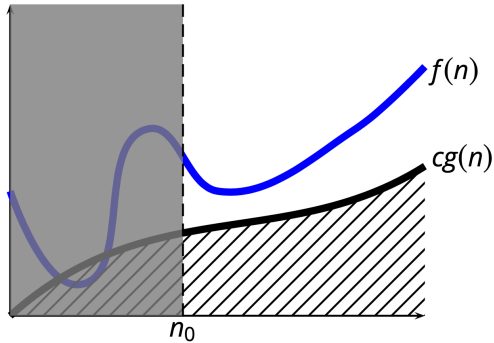
Notación Ω

- Dada una función $g(n)$, definimos una *familia de funciones* $\Omega(g(n))$



Notación Ω

- Dada una función $g(n)$, definimos una *familia de funciones* $\Omega(g(n))$



$\Omega(g(n)) = \quad f(n) \quad : \exists c > 0, \exists n_0 > 0$

$: 0 \leq cg(n) \leq f(n)$ para todo $n \geq n_0\}$

$f(n) = O(g(n))$ $f(n)$ es o-grande de $g(n)$

Θ, O, y Ω como Relaciones

- *Teorema:* para cualquier par de funciones $f(n)$ y $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

Θ, O, y Ω como Relaciones

- *Teorema:* para cualquier par de funciones $f(n)$ y $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- Las notaciones Θ Ω, y O pueden entenderse como las relaciones *asintóticas* de las funciones $=$, $\geq$, y $\leq$ respectivamente.

Θ, O, y Ω como Relaciones

- *Teorema:* para cualquier par de funciones $f(n)$ y $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- Las notaciones Θ Ω, y O pueden entenderse como las relaciones *asintóticas* de las funciones $=$, $\geq$, y $\leq$ respectivamente.

- El teorema anterior puede ser interpretado como

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

Θ, O, y Ω como Relaciones

- *Teorema:* para cualquier par de funciones $f(n)$ y $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- Las notaciones Θ Ω, y O pueden entenderse como las relaciones *asintóticas* de las funciones $=$, $\geq$, y $\leq$ respectivamente.

- El teorema anterior puede ser interpretado como

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

- Caundo $f(n) = O(g(n))$ decimos que $g(n)$ es una ***cota superior*** de $f(n)$, y que $g(n)$ ***domina*** $f(n)$

Θ, O, y Ω como Relaciones

- *Teorema:* para cualquier par de funciones $f(n)$ y $g(n)$,
  $f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

- Las notaciones Θ Ω, y O pueden entenderse como las relaciones *asintóticas* de las funciones $=$, $\geq$, y $\leq$ respectivamente.

- El teorema anterior puede ser interpretado como

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

- Caundo $f(n) = O(g(n))$ decimos que $g(n)$ es una ***cota superior*** de $f(n)$, y que $g(n)$ ***domina*** $f(n)$

- Cuando $f(n) = \Omega(g(n))$ decimos $g(n)$ es una ***cota inferior*** de $f(n)$

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

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

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

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

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

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

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

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

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

  $n^2 + O(n) - 1 = O(n^2)$? SÍ

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

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

  $n^2 + O(n) - 1 = O(n^2)$? SÍ

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

Θ, O, y Ω como funciones anónimas

- Podemos utilizar las notaciones Θ, O y Ω para representar funciones anónimas (desconocidas o no especificadas)
  Por ejemplo,

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

  significa que $f(n)$ es igual a $10n^2$ mas una función desconocida o que no nos interesa conocer, pero que es linal con respecto de n.

- Ejemplos

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

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

  $n^2 + O(n) - 1 = O(n^2)$? SÍ

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

Notación o

Notación *O*

- La cota superior definida por la notación *O* puede o no ser ***asintóticamente ajustada***

- La cota superior definida por la notación O puede o no ser ***asintóticamente ajustada***

  Por ejemplo,

  $n \log n = O(n^2)$ no es asintóticamente ajustada

  $n^2 - n + 10 = O(n^2)$ si e asintóticamente ajustada

Notación o

- La cota superior definida por la notación O puede o no ser ***asintóticamente ajustada***

  Por ejemplo,

  $n \log n = O(n^2)$ no es asintóticamente ajustada

  $n^2 - n + 10 = O(n^2)$ si e asintóticamente ajustada

- Se utiliza la notación o para representar cotas superiores que *no* son asintóticamente ajustadas. Es decir, dada una funcuón $g(n)$, se define la familia de funciones $o(g(n))$

Notación o

- La cota superior definida por la notación O puede o no ser ***asintóticamente ajustada***

  Por ejemplo,

  $n \log n = O(n^2)$ no es asintóticamente ajustada

  $n^2 - n + 10 = O(n^2)$ si e asintóticamente ajustada

- Se utiliza la notación o para representar cotas superiores que *no* son asintóticamente ajustadas. Es decir, dada una funcuón $g(n)$, se define la familia de funciones $o(g(n))$

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

Notación ω

- La cota inferir definida por la notación Ω puede o no ser *asintóticamente ajustada*

Notación ω

- La cota inferir definida por la notación Ω puede o no ser *asintóticamente ajustada* Por ejemplo,

  $2^n = \Omega(n \log n)$ no es asintóticamente ajustada

  $n + 4n \log n = \Omega(n \log n)$ si es asintóticamente ajustada

Notación ω

- La cota inferir definida por la notación Ω puede o no ser *asintóticamente ajustada* Por ejemplo,

  $2^n = \Omega(n \log n)$ no es asintóticamente ajustada

  $n + 4n \log n = \Omega(n \log n)$ si es asintóticamente ajustada

- Se utiliza la notacion ω para denotar cotas que *no* son asintóticamente ajustadas. Es decir, se define una familia de funciones $\omega(g(n))$

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