Complejidad Algorítmica

Mientras ($i \leq n$)
{
$x \leftarrow x+1$
}
$\rightarrow f(n) = an + b$

Para ($i < n$)
Para ($j < n$) {
x++
}
$\rightarrow f(n) = an^2 + bn + c$

Dr. Jaime Osorio Ubaldo

conjunto de llegada

Definición.

Sean $f,\ g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$.

dominio

Orden de complejidad: Big $\mathcal{O}$

Definición.

Sean $f,\ g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$.El conjunto de las funciones del orden de $g(n)$, denotado $\mathcal{O}(g(n))$, se define como sigue:

$\forall$ para todo

$\exists$ Existe al menos uno

$\exists!$ Existe un único

$c=2,\ k=1,\ \forall n\geq 1,\ [6n \leq 2(5n+3)]$

$-6 \leq 4n$

Definición.

Sean $f,\ g:\mathbb{N}\to\mathbb{R}^+\cup\{0\}$.El conjunto de las funciones del orden de $g(n)$, denotado $\mathcal{O}(g(n))$, se define como sigue:

$-3/2 \leq n$ $\quad 0$

$$\mathcal{O}(g(n)) = \{f(n)\ :\ \exists c\in\mathbb{R}^+, \exists k\in\mathbb{N}, \forall n\geq k, [\ f(n)\leq c\cdot g(n)\ ]\}$$

$c=1,\ k=1,\ \forall n\geq 1,\ n \leq c(5n+3)$

$n \leq 5n+3$

$\mathcal{O}(5n+3) = \{6n,\ n, \ldots\}$

$-3 \leq 4n$

$-3/4 \leq n$

$n^2 \in \mathcal{O}(5n+3) \ldots F$

$6n \in \mathcal{O}(5n+3) \ldots V$

Orden de complejidad: Big $\mathcal{O}$

Definición.

Sean $f, \ g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$.El conjunto de las funciones del orden de $g(n)$, denotado $\mathcal{O}(g(n))$, se define como sigue:

$\leq 1°$ $\quad$ $1°$ grado

$$\mathcal{O}(g(n)) = \{f(n) \ : \ \exists c \in \mathbb{R}^+, \exists k \in \mathbb{N}, \forall n \geq k, [\ f(n) \leq c \cdot g(n)\ ]\}$$

diremos que una función f es del orden $g(n)$ cuando $f \in \mathcal{O}(g(n))$.

$$\mathcal{O}(5n+3) = \{n, 2n, 100n, 8, 50000, \log n, \dots\}$$

Orden de complejidad: Big $\mathcal{O}$

Prob. Ordenamiento → Alg1 $\in O(n^2)$
→ Alg2 $\in O(n \log n)$

Definición.

Sean $f,\ g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$.El conjunto de las funciones del orden de $g(n)$, denotado $\mathcal{O}(g(n))$, se define como sigue:

$$\mathcal{O}(g(n)) = \{f(n)\ :\ \exists c \in \mathbb{R}^+, \exists k \in \mathbb{N}, \forall n \geq k, [\ f(n) \leq c \cdot g(n)\ ]\}$$

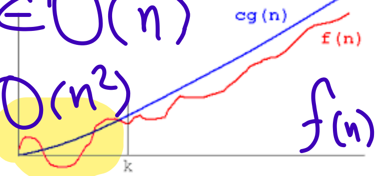
diremos que una función f es del orden $g(n)$ cuando $f \in \mathcal{O}(g(n))$.

$f(n) = 8n+20 \in O(n)$

$f(n) = n\log n + n^2 \in O(n^2)$

$f(n) = 20n^2+n+1 \in O(n^2)$

$f(n) = n\log n + n \in O(n \log n)$



$f(n) = \log n + 20 \in O(\log n)$ →

Ejemplos.

1. Para $f(n) = 3$ es $\mathcal{O}(1)$,

$\exists c = 4, \exists k = 1, \forall n \geq 1, [3 \leq c \cdot 1]$

$f(n) = 8000 \in O(1)$

Ejemplos.

1. Para $f(n) = 3$ es $\mathcal{O}(1)$,ya que

$$\exists c = 3, \exists k = 2, \forall n \geq k, [\ f(n) \leq c \cdot 1\ ]$$

Ejemplos.

1. Para $f(n) = 3$ es $\mathcal{O}(1)$,ya que

$$\exists c = 3, \exists k = 2, \forall n \geq k, [\ f(n) \leq c \cdot 1\ ]$$

2. Para $f(n) = 5n + 3$ es $\mathcal{O}(n)$,

$C = 60$, $K = 1$, $\forall n \geq 1$, $[5n+3 \leq 60n]$

V

Ejemplos.

1. Para $f(n) = 3$ es $\mathcal{O}(1)$,ya que

$$\exists c = 3, \exists k = 2, \forall n \geq k, [\ f(n) \leq c \cdot 1\ ]$$

2. Para $f(n) = 5n + 3$ es $\mathcal{O}(n)$,ya que

$$\exists c = 6, \exists k = 3, \forall n \geq k, [\ f(n) \leq c \cdot n\ ]$$

Definición.

Sean $f, \; g : \mathbb{N} \to \mathbb{R}^{+} \cup \{0\}$.

Definición.

Sean $f,\ g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$.El conjunto $\Omega(g(n))$, se lee omega de $g(n)$, se define como sigue:

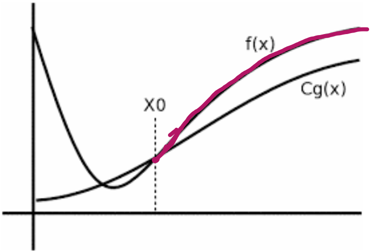
$$\Omega(g(n)) = \{f(n)\ :\ \exists c \in \mathbb{R}^+, \exists x_o \in \mathbb{N}, \forall n \geq x_0, [\ f(n) \geq c \cdot g(n)\ ]\}$$

$\Omega(n^2) = \{n^3, n^4, n^2, n^2 \log n, 2^n, n! \ldots\}$

Definición.

Sean $f, \ g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$.El conjunto $\Omega(g(n))$, se lee omega de $g(n)$, se define como sigue:

$$\Omega(g(n)) = \{f(n) \ : \ \exists c \in \mathbb{R}^+, \exists x_o \in \mathbb{N}, \forall n \geq x_0, [\ f(n) \geq c \cdot g(n)\ ]\}$$



Definición.

El conjunto de funciones $\theta(g(n))$, se lee orden exacto de $g(n)$, se define como

$$\theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$

Orden de Complejidad

Definición.

El conjunto de funciones $\theta(g(n))$, se lee orden exacto de $g(n)$, se define como

$$\theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$

es decir

$$\theta(g(n)) = \{f(n) \ : \ \exists c_1, c_2 \in \mathbb{R}^+, \exists n_o \in \mathbb{N}, \forall n \geq n_0, [\ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n$$

Orden de Complejidad

Definición.

El conjunto de funciones $\theta(g(n))$, se lee orden exacto de $g(n)$, se define como

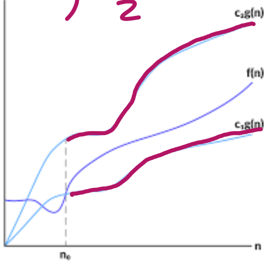
$$\theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$

es decir

$$\frac{n^2}{5} \le n^2+n+1 \le 5n^2$$

$$\theta(g(n)) = \{f(n) \ : \ \exists c_1, c_2 \in \mathbb{R}^+, \exists n_o \in \mathbb{N}, \forall n \geq n_0, [\ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g($$

$$\theta(n^2) = \{n^2, 2n^2, \frac{n^2}{2}, n^2+n+1, \ldots\}$$



f(n) = Θ(g(n))

Propiedades

Sean las funciones $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$

Propiedades

Sean las funciones $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$
2. $f(n) \in \mathcal{O}(g(n)) \to \mathcal{O}(f(n)) \subset \mathcal{O}(g(n))$

Propiedades

Sean las funciones $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$
2. $f(n) \in \mathcal{O}(g(n)) \to \mathcal{O}(f(n)) \subset \mathcal{O}(g(n))$
3. $\mathcal{O}(f(n)) = \mathcal{O}(g(n)) \leftrightarrow [\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(f(n))\ ]$

Propiedades

Sean las funciones $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$
2. $f(n) \in \mathcal{O}(g(n)) \to \mathcal{O}(f(n)) \subset \mathcal{O}(g(n))$
3. $\mathcal{O}(f(n)) = \mathcal{O}(g(n)) \leftrightarrow [\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(f(n))\ ]$
4. $[\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(h(n))\ ] \to f(n) \in \mathcal{O}(h(n))$

Propiedades

Sean las funciones $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$
2. $f(n) \in \mathcal{O}(g(n)) \to \mathcal{O}(f(n)) \subset \mathcal{O}(g(n))$
3. $\mathcal{O}(f(n)) = \mathcal{O}(g(n)) \leftrightarrow [\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(f(n))\ ]$
4. $[\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(h(n))\ ] \to f(n) \in \mathcal{O}(h(n))$
5. $[\ f(n) \in \mathcal{O}(g(n)) \wedge f(n) \in \mathcal{O}(h(n))\ ] \to f(n) \in \mathcal{O}(min(g(n), h(n)))$

Propiedades

Sean las funciones $f, g : \mathbb{N} \rightarrow \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$
2. $f(n) \in \mathcal{O}(g(n)) \rightarrow \mathcal{O}(f(n)) \subset \mathcal{O}(g(n))$
3. $\mathcal{O}(f(n)) = \mathcal{O}(g(n)) \leftrightarrow [\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(f(n))\ ]$
4. $[\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(h(n))\ ] \rightarrow f(n) \in \mathcal{O}(h(n))$
5. $[\ f(n) \in \mathcal{O}(g(n)) \wedge f(n) \in \mathcal{O}(h(n))\ ] \rightarrow f(n) \in \mathcal{O}(min(g(n), h(n)))$
6. $[\ f_1(n) \in \mathcal{O}(g(n)) \wedge f_2(n) \in \mathcal{O}(h(n))\ ] \rightarrow f_1(n) + f_2(n) \in \mathcal{O}(max(g(n), h(n)))$

Propiedades

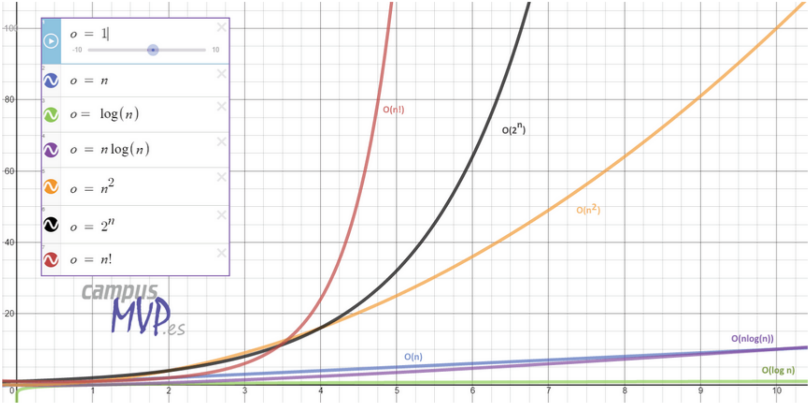
Pruebe cualquiera

Sean las funciones $f, g : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ se cumple lo siguiente:

1. $f(n) \in \mathcal{O}(f(n))$
2. $f(n) \in \mathcal{O}(g(n)) \to \mathcal{O}(f(n)) \subset \mathcal{O}(g(n))$
3. $\mathcal{O}(f(n)) = \mathcal{O}(g(n)) \leftrightarrow [\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(f(n))\ ]$
4. $[\ f(n) \in \mathcal{O}(g(n)) \wedge g(n) \in \mathcal{O}(h(n))\ ] \to f(n) \in \mathcal{O}(h(n))$
5. $[\ f(n) \in \mathcal{O}(g(n)) \wedge f(n) \in \mathcal{O}(h(n))\ ] \to f(n) \in \mathcal{O}(min(g(n), h(n)))$
6. $[\ f_1(n) \in \mathcal{O}(g(n)) \wedge f_2(n) \in \mathcal{O}(h(n))\ ] \to f_1(n) + f_2(n) \in \mathcal{O}(max(g(n), h(n)))$
7. $[\ f_1(n) \in \mathcal{O}(g(n)) \wedge f_2(n) \in \mathcal{O}(h(n))\ ] \to f_1(n) \cdot f_2(n) \in \mathcal{O}(g(n) \cdot h(n))$

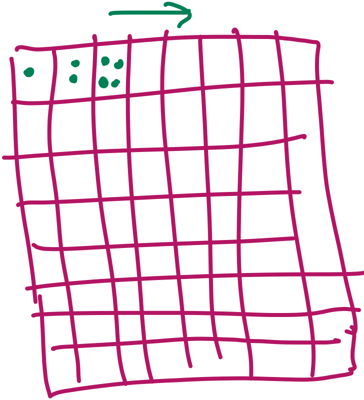
(2ptos PC2)

Orden de Complejidad



Orden de complejidad exponencial

La Leyenda de Sisa



1° 2° 3° 4° . . . 64°

$$1+2+4+8+\cdots+2^{63}=\frac{2^{64}-1}{2-1}$$

La Leyenda de Sisa

$=$

$$f(n)=2^n-1\in O(2^n)$$

Orden de complejidad

Efecto de duplicar el tamaño del problema.

f(n)	n=100	n=200
$k_1 \log n$	1h	1.15h
$k_2 n$	1h	2h
$k_3 n \log n$	1h	

Orden de complejidad

Efecto de duplicar el tamaño del problema.

f(n)	n=100	n=200
$k_1 \log n$	1h	1.15h
$k_2 n$	1h	2h
$k_3 n \log n$	1h	2.3h
$k_4 n^2$	1h	

Orden de complejidad

Efecto de duplicar el tamaño del problema.

ProblemaX

f(n)	n=100	n=200
$k_1 \log n$	1h	1.15h
$k_2 n$	1h	2h
$k_3 n \log n$	1h	2.3h
$k_4 n^2$	1h	4h
$k_5 n^3$	1h	8h
$k_6 2^n$	1h	$1.27 \times 10^{30} h$

Orden de complejidad

Efecto de duplicar el tamaño del problema.

f(n)	n=100	n=200
$k_1 \log n$	1h	1.15h
$k_2 n$	1h	2h
$k_3 n \log n$	1h	2.3h
$k_4 n^2$	1h	4h
$k_5 n^3$	1h	8h
$k_6 2^n$	1h	$1.27 \times 10^{30} h$

Los que se comportan de un modo más acorde con las expectativas del usuario no informático son los de complejidad lineal y $\mathcal{O}(n \log n)$:

Orden de complejidad

Efecto de duplicar el tamaño del problema.

f(n)	n=100	n=200
$k_1 \log n$	1h	1.15h
$k_2 n$	1h	2h
$k_3 n \log n$	1h	2.3h
$k_4 n^2$	1h	4h
$k_5 n^3$	1h	8h
$k_6 2^n$	1h	$1.27 \times 10^{30} h$

Los que se comportan de un modo más acorde con las expectativas del usuario no informático son los de complejidad lineal y $\mathcal{O}(n \log n)$: al duplicar el tamaño del problema se duplica aproximadamente el tiempo empleado, y al duplicar el tiempo disponible, el tamaño que es posible tratar también se duplica.

Orden de complejidad

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	n=178
$k_4 n^2$	n=100	

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	n=178
$k_4 n^2$	n=100	n=141
$k_5 n^3$	n=100	

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	n=178
$k_4 n^2$	n=100	n=141
$k_5 n^3$	n=100	n=126
$k_6 2^n$	n=100	

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	n=178
$k_4 n^2$	n=100	n=141
$k_5 n^3$	n=100	n=126
$k_6 2^n$	n=100	n=101

Orden de complejidad

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	n=178
$k_4 n^2$	n=100	n=141
$k_5 n^3$	n=100	n=126
$k_6 2^n$	n=100	n=101

1. El algoritmo de complejidad logarítmica tiene un comportamiento excepcionalmente bueno: doblar el tiempo disponible permite tratar problemas enormes en relación con el original.

Efecto de duplicar el tiempo disponible.

f(n)	t=1h	t=2h
$k_1 \log n$	n=100	n=10 000
$k_2 n$	n=100	n=200
$k_3 n \log n$	n=100	n=178
$k_4 n^2$	n=100	n=141
$k_5 n^3$	n=100	n=126
$k_6 2^n$	n=100	n=101

1. El algoritmo de complejidad logarítmica tiene un comportamiento excepcionalmente bueno: doblar el tiempo disponible permite tratar problemas enormes en relación con el original.
2. Los órdenes cuadráticos y cúbicos tienen un comportamiento claramente inferior al lineal: en la tabla anterior se observa que un incremento del 100 % en el tiempo disponible solo se consigue un incremento del 41 % y del 26 %, respectivamente, en el tamaño del problema.

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.
   Ejemplos.
   1. El problema de la matriz mortal (determinar, dado un conjunto finito de n n matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición, para obtener la matriz cero).

A, B, C, D, E

ABCDE
BACDE   ? = $\emptyset$
CDEBA

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.
   Ejemplos.
   1. El problema de la matriz mortal (determinar, dado un conjunto finito de n n matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición, para obtener la matriz cero).
   2. El problema de la ecuaciones diofánticas.

$$x^4 + y^4 = z^4$$

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.
   Ejemplos.
   1. El problema de la matriz mortal (determinar, dado un conjunto finito de n n matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición, para obtener la matriz cero).
   2. El problema de la ecuaciones diofánticas.
   3. El problema de la decisión (El Entscheidungsproblem de Hilbert)

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.
   Ejemplos.
   1. El problema de la matriz mortal (determinar, dado un conjunto finito de n n matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición, para obtener la matriz cero).
   2. El problema de la ecuaciones diofánticas.
   3. El problema de la decisión (El Entscheidungsproblem de Hilbert)
2. Problemas decidibles. Son aquellos problemas que poseen algoritmos capaces para resolverlos.

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.
   Ejemplos.
   1. El problema de la matriz mortal (determinar, dado un conjunto finito de n n matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición, para obtener la matriz cero).
   2. El problema de la ecuaciones diofánticas.
   3. El problema de la decisión (El Entscheidungsproblem de Hilbert)
2. Problemas decidibles. Son aquellos problemas que poseen algoritmos capaces para resolverlos.
   Tipos.
   1. Tratables

Clasificación de los Problemas

1. Problemas indecidibles. Son aquellos problemas que no poseen algoritmos capaces para resolverlos.
   Ejemplos.
   1. El problema de la matriz mortal (determinar, dado un conjunto finito de n n matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición, para obtener la matriz cero).
   2. El problema de la ecuaciones diofánticas.
   3. El problema de la decisión (El Entscheidungsproblem de Hilbert)
2. Problemas decidibles. Son aquellos problemas que poseen algoritmos capaces para resolverlos.
   Tipos.
   1. Tratables
   2. Intratables

Problemas Tratables

1. Los problemas de complejidad polinomial se denominan problemas tratables.

Problemas Tratables

1. Los problemas de complejidad polinomial se denominan problemas tratables.
2. En general, en un algoritmo de complejidad polinomial, es decir, de orden $\mathcal{O}(n^a)$, al multiplicar el tiempo disponible (o la velocidad del computador) por un factor k multiplica el tamaño del problema que es posible tratar por un factor $\sqrt[a]{k}$.

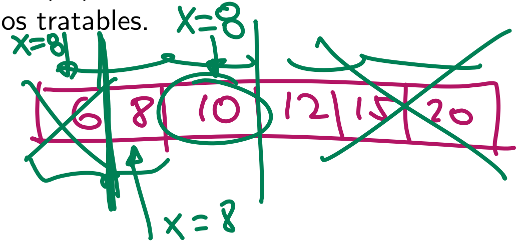
Problemas Tratables

1. Los problemas de complejidad polinomial se denominan problemas tratables.
2. En general, en un algoritmo de complejidad polinomial, es decir, de orden $\mathcal{O}(n^a)$, al multiplicar el tiempo disponible (o la velocidad del computador) por un factor k multiplica el tamaño del problema que es posible tratar por un factor $\sqrt[a]{k}$.
3. Si el orden de complejidad es $\mathcal{O}(n^a)$, cuando "a" es mayor estos problemas son cada vez menos tratables.
4. Ejemplos.

Problemas Tratables

1. Los problemas de complejidad polinomial se denominan problemas tratables.
2. En general, en un algoritmo de complejidad polinomial, es decir, de orden $\mathcal{O}(n^a)$, al multiplicar el tiempo disponible (o la velocidad del computador) por un factor k multiplica el tamaño del problema que es posible tratar por un factor $\sqrt[a]{k}$.
3. Si el orden de complejidad es $\mathcal{O}(n^a)$, cuando "a" es mayor estos problemas son cada vez menos tratables.
4. Ejemplos.
    1. Búsqueda binaria.

$E \log(n)$



Problemas Tratables

1. Los problemas de complejidad polinomial se denominan problemas tratables.
2. En general, en un algoritmo de complejidad polinomial, es decir, de orden $\mathcal{O}(n^a)$, al multiplicar el tiempo disponible (o la velocidad del computador) por un factor k multiplica el tamaño del problema que es posible tratar por un factor $\sqrt[a]{k}$.
3. Si el orden de complejidad es $\mathcal{O}(n^a)$, cuando "a" es mayor estos problemas son cada vez menos tratables.
4. Ejemplos.
   1. Búsqueda binaria.
   2. Ordenación por inserción. $\in O(n^2)$

Problemas Intratables

1. Los problemas de complejidad exponencial o mayor, reciben el nombre de intratables. Se llaman así los problemas cuyos mejores algoritmos tienen tiempos en $\Omega(2^n)$.

Problemas Intratables

1. Los problemas de complejidad exponencial o mayor, reciben el nombre de intratables. Se llaman así los problemas cuyos mejores algoritmos tienen tiempos en $\Omega(2^n)$.
2. Un algoritmo de tiempo de ejecución exponencial al duplicar la velocidad del procesador apenas afecta al tamaño del problema tratado, y duplicar el tamaño del problema conduce a tiempos de ejecución del orden de varios billones de veces la edad del universo (un siglo son aproximadamente 10^6 horas).

Ejemplos.

1. Un ejemplo típico es encontrar la ruta más corta para visitar varias ciudades (el problema del agente viajero).

$\in O(n!)$

Orden de complejidad

Ejemplos.

1. Un ejemplo típico es encontrar la ruta más corta para visitar varias ciudades (el problema del agente viajero).
2. El problema de la asignación cuando intervienen 3 o más dimensiones. Por ejemplo, la asignación de salones, horarios, profesores y asignaturas tiene 4 dimensiones.

Orden de complejidad

Ejemplos.

1. Un ejemplo típico es encontrar la ruta más corta para visitar varias ciudades (el problema del agente viajero).
2. El problema de la asignación cuando intervienen 3 o más dimensiones. Por ejemplo, la asignación de salones, horarios, profesores y asignaturas tiene 4 dimensiones.
3. El problema de la mochila.

Orden de complejidad

Ejemplos.

1. Un ejemplo típico es encontrar la ruta más corta para visitar varias ciudades (el problema del agente viajero).
2. El problema de la asignación cuando intervienen 3 o más dimensiones. Por ejemplo, la asignación de salones, horarios, profesores y asignaturas tiene 4 dimensiones.
3. El problema de la mochila.
4. La búsqueda del camino simple más largo de un grafo.

Ejemplos.

1. Un ejemplo típico es encontrar la ruta más corta para visitar varias ciudades (el problema del agente viajero).
2. El problema de la asignación cuando intervienen 3 o más dimensiones. Por ejemplo, la asignación de salones, horarios, profesores y asignaturas tiene 4 dimensiones.
3. El problema de la mochila.
4. La búsqueda del camino simple más largo de un grafo.
5. la verificación de la existencia de ciclos hamiltonianos en un grafo.

Orden de complejidad

Ejemplos.

1. Un ejemplo típico es encontrar la ruta más corta para visitar varias ciudades (el problema del agente viajero).
2. El problema de la asignación cuando intervienen 3 o más dimensiones. Por ejemplo, la asignación de salones, horarios, profesores y asignaturas tiene 4 dimensiones.
3. El problema de la mochila.
4. La búsqueda del camino simple más largo de un grafo.
5. la verificación de la existencia de ciclos hamiltonianos en un grafo.

Hasta la fecha no se ha encontrado un algoritmo que resuelva esta clase de problemas en tiempo polinomial. Los mejores algoritmos para resolver estos problemas crecen exponencialmente con el tamaño de la entrada y por esto se les cataloga como problemas intratables. Estos algoritmos requerieren más tiempo del disponible, excepto para tamaños de entrada muy pequeños.

Problemas P y NP

1. Problemas P. Este tipo problemas son resueltos por algoritmos de complejidad polinomial.

Problemas P y NP

1. Problemas P. Este tipo problemas son resueltos por algoritmos de complejidad polinomial.
2. Problema NP. Estos problemas se caracterizan por
    1. Dada una posible solución x_0, es posible comprobar si x_0 es solución o no del problema con un algoritmo de complejidad polinomial.

Problemas P y NP

1. Problemas P. Este tipo problemas son resueltos por algoritmos de complejidad polinomial.
2. Problema NP. Estos problemas se caracterizan por
    1. Dada una posible solución x_0, es posible comprobar si x_0 es solución o no del problema con un algoritmo de complejidad polinomial. Ejemplo.El problema de la mochila.

Problemas P y NP

1. Problemas P. Este tipo problemas son resueltos por algoritmos de complejidad polinomial.
2. Problema NP. Estos problemas se caracterizan por
   1. Dada una posible solución x_0, es posible comprobar si x_0 es solución o no del problema con un algoritmo de complejidad polinomial. Ejemplo.El problema de la mochila.
   2. No sabemos si pueden ser resueltos en por algoritmos de complejidad polinomial.

Problemas P y NP

1. Problemas P. Este tipo problemas son resueltos por algoritmos de complejidad polinomial.
2. Problema NP. Estos problemas se caracterizan por
   1. Dada una posible solución x_0, es posible comprobar si x_0 es solución o no del problema con un algoritmo de complejidad polinomial. Ejemplo.El problema de la mochila.
   2. No sabemos si pueden ser resueltos en por algoritmos de complejidad polinomial.

$$https://www.youtube.com/watch?v=YX40hbAHx3s$$

El Problema del Milenio

Probar que $N = NP$ **o** $P \neq NP$

Probar que $N = NP$ **o** $P \neq NP$

1. Nadie ha podido probarlo.

El Problema del Milenio

Probar que $N = NP$ **o** $P \neq NP$

1. Nadie ha podido probarlo.
2. Hay un premio de un millón de dólares por el Instituto de Matemática Clay para quien lo resuelva.

El Problema del Milenio

Probar que $N = NP$ **o** $P \neq NP$

1. Nadie ha podido probarlo.
2. Hay un premio de un millón de dólares por el Instituto de Matemática Clay para quien lo resuelva.
3. Ventajas. El avance de la inteligencia artificial (tal vez no sea lo más conveniente para muchas empresas).

El Problema del Milenio

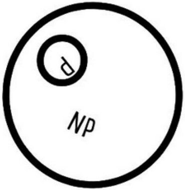
Probar que $N = NP$ **o** $P \neq NP$

1. Nadie ha podido probarlo.
2. Hay un premio de un millón de dólares por el Instituto de Matemática Clay para quien lo resuelva.
3. Ventajas. El avance de la inteligencia artificial (tal vez no sea lo más conveniente para muchas empresas).
4. Desventajas. La seguridad informática puede tambalear (teléfonos, comunicaciones por internet, acceso a datos cifrados, etc).

El Problema del Milenio

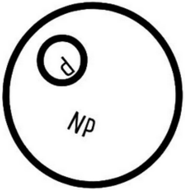
Probar que $N = NP$ **o** $P \neq NP$

1. Nadie ha podido probarlo.
2. Hay un premio de un millón de dólares por el Instituto de Matemática Clay para quien lo resuelva.
3. Ventajas. El avance de la inteligencia artificial (tal vez no sea lo más conveniente para muchas empresas).
4. Desventajas. La seguridad informática puede tambalear (teléfonos, comunicaciones por internet, acceso a datos cifrados, etc).



Probar que $N = NP$ **o** $P \neq NP$

1. Nadie ha podido probarlo.
2. Hay un premio de un millón de dólares por el Instituto de Matemática Clay para quien lo resuelva.
3. Ventajas. El avance de la inteligencia artificial (tal vez no sea lo más conveniente para muchas empresas).
4. Desventajas. La seguridad informática puede tambalear (teléfonos, comunicaciones por internet, acceso a datos cifrados, etc).



Conclusiones

1. Es una buena inversión dedicar tiempo a encontrar algoritmos con mejores tasas de crecimiento. Por ejemplo pasar de un algoritmo $\mathcal{O}(n^2)$ a otro $\mathcal{O}(n \log n)$ ha de considerarse una gran mejora.

Conclusiones

1. Es una buena inversión dedicar tiempo a encontrar algoritmos con mejores tasas de crecimiento. Por ejemplo pasar de un algoritmo $\mathcal{O}(n^2)$ a otro $\mathcal{O}(n \log n)$ ha de considerarse una gran mejora.
2. Encontrar un algoritmo $\mathcal{O}(\log n)$ para problemas que se resolvían en $\mathcal{O}(n)$ es una inmensa suerte.

Conclusiones

1. Es una buena inversión dedicar tiempo a encontrar algoritmos con mejores tasas de crecimiento. Por ejemplo pasar de un algoritmo $\mathcal{O}(n^2)$ a otro $\mathcal{O}(n \log n)$ ha de considerarse una gran mejora.
2. Encontrar un algoritmo $\mathcal{O}(\log n)$ para problemas que se resolvían en $\mathcal{O}(n)$ es una inmensa suerte.
3. Encontrar un algoritmo polinomial para problemas cuyos mejores algoritmos conocidos sean exponenciales, es un logro merecedor del premio Turing (el equivalente en Informática al premio Nobel).

Conclusiones

1. Es una buena inversión dedicar tiempo a encontrar algoritmos con mejores tasas de crecimiento. Por ejemplo pasar de un algoritmo $\mathcal{O}(n^2)$ a otro $\mathcal{O}(n \log n)$ ha de considerarse una gran mejora.
2. Encontrar un algoritmo $\mathcal{O}(\log n)$ para problemas que se resolvían en $\mathcal{O}(n)$ es una inmensa suerte.
3. Encontrar un algoritmo polinomial para problemas cuyos mejores algoritmos conocidos sean exponenciales, es un logro merecedor del premio Turing (el equivalente en Informática al premio Nobel).
4. Desgraciadamente, existen muchos problemas interesantes que han permanecido hasta la fecha en la categoría de exponenciales pese a los esfuerzos de numerosos investigadores, por lo que se sopecha con gran convicción que tales algoritmos no existen.

5. Si bien el criterio asintótico es muy útil hay circunstancias especiales en el que si el tamaño del problema no es muy grande o el algoritmo solo se usa unas cuantas veces se puede optar por usar el algoritmo teóricamente menos eficiente.

Ejercicio

Revisar los siguientes problemas NP y explique en qué consiste.

1. El problema de la mochila.
2. El problema del agente viajero.