

Introducción al estudio de algoritmos

1. Algoritmo, programa y pseudocódigo
2. Eficiencia y el principio de invarianza
3. Operaciones elementales
4. Casos mejor, peor y medio
5. Notación asintótica
6. Análisis de las estructuras de control
7. Resolución de recurrencias

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

1. Algoritmo, programa y pseudocódigo

- Algoritmo = Procedimiento o método para resolver un problema o lograr un fin
- Ejemplos
 - receta de cocina
 - factorización en números primos

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

1. Algoritmo, programa y pseudocódigo

- Programa = algoritmo expresado en un lenguaje de programación
- Pseudocódigo = lenguaje para especificar algoritmos

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

1. Algoritmo, programa y pseudocódigo

- Pseudocódigo para Algorítmica
 - Subconjunto del castellano orientado matemáticamente (ops. aritméticas)
 - Imita al Pascal/ADA (procedimientos, funciones, etc. menos “begin-end”)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

1. Algoritmo, programa y pseudocódigo

Procedimiento factorizar(n: natural)

factor = n

primo = 2

mientras (factor > primo)

hacer

si ((factor mod primo) = 0)

entonces

 factor = factor div primo

 escribir(primo)

sino primo = primo + 1

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

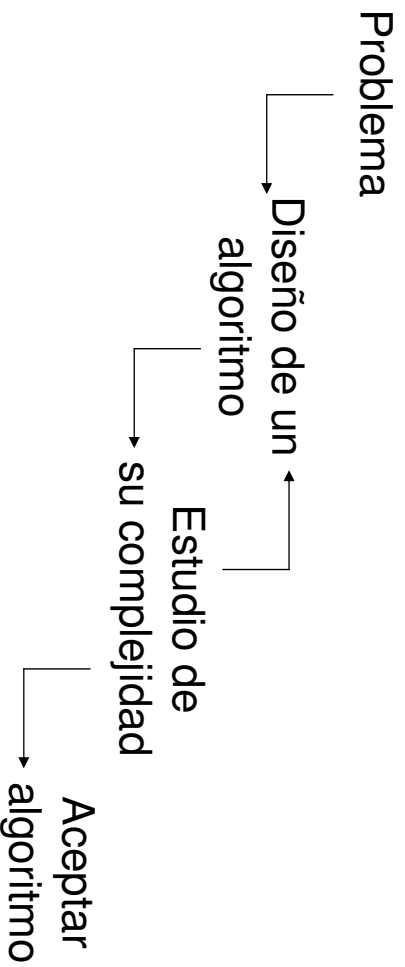
1. Algoritmo, programa y pseudocódigo

- Dos aspectos de un algoritmo
 - ¿Funciona? => corrección (verificación formal)
 - ¿Es rápido? => eficiencia (complejidad)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

1. Algoritmo, programa y pseudocódigo

- Proceso de diseño de algoritmos



Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Estudio de la eficiencia
- Objetivo
 - Dados dos algoritmos que resuelven un problema (correctos), compararlos para seleccionar el más eficiente (rápido)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Estudio de la eficiencia
 - Estudio teórico
 - Estudio empírico

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Estudio teórico
 - Se analizan los algoritmos y se caracteriza teóricamente su eficiencia
 - Estudio de la complejidad (función del tamaño de los datos de entrada)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Estudio empírico
 - Se programan ambos algoritmos y se efectúan pruebas con un conjunto de datos de muestra

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Principio de invarianza
 - La unidad con respecto a la que se caracteriza la eficiencia de un algoritmo podría depender de la *computadora* y de la *implementación*
 - ¡¡¡Debe evitarse!!!

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Principio de invarianza (informalmente)
 - Dos implementaciones distintas de un mismo algoritmo no diferirán en su eficiencia en más que una constante multiplicativa

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Principio de invarianza (formalmente)
 - Si dos implementaciones distintas del mismo algoritmo precisan respectivamente $t_1(n)$ y $t_2(n)$ unidades de tiempo, siendo n el tamaño o número de datos, siempre existen constantes positivas a y b tales que
$$t_1(n) \leq a \cdot t_2(n) \text{ y } t_2(n) \leq b \cdot t_1(n)$$
para un n suficientemente grande

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Principio de invarianza (ejemplos)
 - Las funciones $t_1(n) = 10.n + 15$ y $t_2(n) = 1000.n - 2$ son comparables o equivalentes (pueden corresponder a implementaciones distintas del mismo algoritmo)
 - Las funciones $t_1(n) = n$ y $t_2(n) = n^2$ NO son equivalentes (NO pueden corresponder a implementaciones distintas del mismo algoritmo)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Principio de invarianza (consecuencias)
 - No es preciso usar una unidad de tiempo en las funciones que caracterizan algoritmos => se usa una función abstracta del tamaño de los datos
 - El cambio de implementación (computadora) sólo produce mejoras *constantes*, mientras que el cambio de algoritmo puede producir mejoras *crecientes* con el tamaño de los datos

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

2. Eficiencia y el principio de invarianza

- Principio de invarianza (más ejemplos)

	C1	C2 = 100 x C1	Mejora (n)
A1	$t_{11}(n) = 10^{-4} \cdot 2^n$ s.	$t_{12}(n) = 10^{-6} \cdot 2^n$ s.	~ 7 u.
A2	$t_{21}(n) = 10^{-2} \cdot n^3$ s.	$t_{22}(n) = 10^{-4} \cdot n^3$ s.	$\sim 4,6 \cdot n$ u.

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

3. Operaciones elementales

- Operaciones elementales
 - Aquellas cuyo tiempo de ejecución se puede acotar superiormente por una constante sólo dependiente de la implementación, pero no del tamaño de los datos
 - No es relevante su tiempo de ejecución (complejidad de algoritmos salvo constante multiplicativa)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

3. Operaciones elementales

- Ejemplo
 - Si un algoritmo efectúa m multiplicaciones, s sumas y a asignaciones con tiempos de ejecución acotados por t_m , t_s y t_a , respectivamente, entonces el tiempo de ejecución del algoritmo estará acotado por $(m + s + a)$
 - Es decir, dependerá sólo del número de instrucciones (que a su vez dependerá del tamaño de los datos)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

3. Operaciones elementales

- No todas las operaciones matemáticas son elementales
- Ejemplos
 - función mínimo de un vector \equiv un bucle que recorre las n componentes
 - muchas operaciones pueden incurrir en un desbordamiento de pila

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

3. Operaciones elementales

- **Enfoque realista**
 - toda operación tiene un coste proporcional al tamaño de sus operandos
- **A efectos prácticos**
 - se consideran elementales las operaciones siguientes: suma, resta, multiplicación, división, módulo, operaciones booleanas, comparaciones, asignación

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

- El tiempo que tarda un algoritmo en resolver un problema de dimensión n puede depender del tipo de ejemplar de problema considerado
- **Ejemplo**
 - Buscar en un árbol degradado a lista requiere más tiempo que si el árbol está perfectamente balanceado, aunque ambos tengan igual número de datos

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

- Ejemplo (métodos de ordenación)
 - Ordenación por selección
 - Se selecciona en cada paso el elemento más pequeño del vector y se coloca el primero de los que quedan
 - Ordenación por inserción
 - Se toma en cada paso un elemento y se inserta en su posición, manteniendo los ya examinados ordenados

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

```
Procedimiento selección(T: vector[1..n] de entero)  
para i = 1 hasta n-1  
hacer   minj = i  
       minx = T[i]  
       para j = i+1 hasta n  
       hacer   si T[j] < minx  
           entonces minj = j  
           minx = T[j]  
       T[minj] = T[i]  
       T[i] = minx
```

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

Procedimiento inserción(T: vector[1..n] de entero)

para i = 2 hasta n

hacer x = T[i]

j = i-1

mientras (j > 0 y x < T[j])

hacer T[j+1] = T[j]

j = j-1

T[j+1] = x

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

- **Ejemplo (métodos de ordenación)**
 - Sean las matrices U y V ordenadas ascendente y descendente, de n componentes
 - La ordenación por selección requiere el mismo número de operaciones para ordenar U y V ($c_1 \cdot n^2$)
 - La ordenación por inserción requiere más operaciones para V ($c_2 \cdot n^2$) que para U ($c_3 \cdot n$) => el tipo de ejemplar afecta al tiempo de ejecución de la ordenación por inserción

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

- El *caso peor* de un algoritmo es aquel ejemplar que exige más trabajo para ser resuelto
- El *caso mejor* de un algoritmo es aquel ejemplar que exige menos trabajo para ser resuelto
- ¡El caso peor acota el tiempo para todos los ejemplares de un tamaño!

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

- El caso peor puede exigir mucho más tiempo que un ejemplar “típico”
- El tiempo del caso medio representa el tiempo aproximado para un ejemplar “típico”
- Se calcula la media del tiempo para todos los posibles ejemplares de tamaño n , asumiendo distribución uniforme

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

4. Casos mejor, peor y medio

- El caso medio tiene sentido cuando existen el caso mejor y el peor, es decir, el tiempo depende del tipo de ejemplar considerado
 - En la ordenación por selección todos los casos exigen el mismo trabajo
 - En la ordenación por inserción, hay caso peor y mejor, y en media se realizan $c_4 \cdot n^2$ operaciones

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (*Orden de $f(n)$*)

Sea $f: \mathbb{N} \rightarrow \mathbb{R}^+$ una función de los números naturales en los reales no negativos; el orden de $f(n)$ ($O(f(n))$) es el conjunto de todas las funciones $t: \mathbb{N} \rightarrow \mathbb{R}^+$ tales que existen una constante real c y un número natural n_0 que cumplan que $t(n) \leq c \cdot f(n)$ para todo $n \geq n_0$.

$$O(f(n)) = \{t: \mathbb{N} \rightarrow \mathbb{R}^+ / \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} (t(n) \leq c \cdot f(n) \forall n \geq n_0)\}$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Se suele hablar de que $g(n)$ “pertenece al o es del orden de $f(n)$ ”
- Por definición

$$g(n) \in O(f(n)) \Leftrightarrow \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} (g(n) \leq c \cdot f(n) \forall n \geq n_0)$$

- Ejemplo: $g(n) = 27n^2 + 15n + 2 \in O(n^2)$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observaciones
 - Por el principio de invarianza, si un algoritmo tarda un tiempo $t(n)$ segundos con $t(n) \in f(n)$, entonces la complejidad del algoritmo es del orden de $f(n)$
 - Al usar umbrales en las definiciones, nos desentendemos del comportamiento de las funciones en los valores pequeños de n
 - Por ejemplo, $n^3 - 3n^2 - n - 8 \in O(n^3)$ aunque sea negativa para $n < 4$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Regla del umbral

Si se consideran funciones estrictamente positivas, entonces el umbral no es necesario.

Sean $f, t: \mathbb{N} \rightarrow \mathbb{R}^+$,

$$t(n) \in O(f(n)) \Leftrightarrow \exists c \in \mathbb{R}^+ t(n) \leq c \cdot f(n)$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Regla del máximo

(2 funciones)

Sean $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$. Entonces $O(f(n)+g(n)) =$

$O(\max(f(n),g(n)))$

(m funciones)

Sean $f_1, \dots, f_m: \mathbb{N} \rightarrow \mathbb{R}^+$. Entonces $O(f_1(n) + \dots +$

$f_m(n)) = O(\max(f_1(n), \dots, f_m(n)))$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Ejemplos
 - $O(n^3 + n^2 + n \cdot \log(n)) = O(\max(n^3, n^2, n \cdot \log(n))) = O(n^3)$
 - Pero hay que tener cuidado, o podríamos concluir que $O(n) = O(n^2)$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Propiedades
 - Si $f(n) \in O(g(n))$, entonces $O(f(n)) \subseteq O(g(n))$
 - Si $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$, entonces $O(f(n)) = O(g(n))$
 - La relación $f \equiv g \Leftrightarrow f(n) \in O(g(n))$ es *reflexiva*, *transitiva* y *antisimétrica* bajo constante
 - Define un *orden parcial* entre funciones

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Demostración de que $f(n) \notin O(g(n))$
 - Por reducción al absurdo, o contradicción
 - Suponer que $f(n) \in O(g(n))$ y llegar a una contradicción (posiblemente aritmética)
 - Por ejemplo, $n^3 \notin O(n^2)$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Regla del límite

Sean $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$. Entonces:

- i) Si $\lim(f(n)/g(n)) \in \mathbb{R}^+$, entonces $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
- ii) Si $\lim(f(n)/g(n)) = 0$, entonces $f(n) \in O(g(n))$ y $g(n) \notin O(f(n))$
- iii) Si $\lim(f(n)/g(n)) = +\infty$, entonces $f(n) \notin O(g(n))$ y $g(n) \in O(f(n))$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Ejemplo
 - $\log n \in O(\sqrt{n})$ pero $\sqrt{n} \notin O(\log n)$
porque $\lim(\log n)/\sqrt{n} = 0$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (*Orden inferior* de $f(n)$)

Sea $f: \mathbb{N} \rightarrow \mathbb{R}^+$ una función de los números naturales en los reales no negativos; el *orden inferior* de $f(n)$ ($\Omega(f(n))$) es el conjunto de todas las funciones $t: \mathbb{N} \rightarrow \mathbb{R}^+$ tales que existen una constante real c y un número natural n_0 que cumplen que $t(n) \geq c \cdot f(n)$ para todo $n \geq n_0$.

$$\Omega(f(n)) = \{t: \mathbb{N} \rightarrow \mathbb{R}^+ / \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} (t(n) \geq c \cdot f(n) \forall n \geq n_0)\}$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observación
 - La notación $O(g(n))$ representa los *máximos* recursos que vamos a necesitar para resolver un problema de tamaño n
 - La notación $\Omega(g(n))$ representa los *mínimos* recursos que vamos a necesitar para resolver un problema de tamaño n

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Regla de la dualidad

Sean $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$. Entonces:

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

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observación
 - Sea $f(n)$ es la función que describe la complejidad de un algoritmo en el caso peor...
 - Si $f(n) \in O(g(n))$, entonces todo ejemplar de tamaño n se resuelve en tiempo en el orden de $g(n)$
 - Si $f(n) \in \Omega(g(n))$, entonces algún ejemplar de tamaño n requiere tiempo mayor de $g(n)$, pero los demás no necesariamente

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (*Orden exacto* de $f(n)$)

Sean $f, t: \mathbb{N} \rightarrow \mathbb{R}^+$ funciones de los números naturales en los reales no negativos;

Decimos que $t(n)$ está en el *orden exacto* de $f(n)$

($\theta(f(n))$) si $t(n) \in O(f(n))$ y $t(n) \in \Omega(f(n))$

Es decir

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

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observaciones y propiedades
 - Sólo los algoritmos que son igual de eficientes en el mejor y peor caso poseen un orden exacto
 - Para el orden inferior y el exacto también se cumplen las reglas del máximo y del umbral
 - Si $f(n) \in \theta(g(n))$, entonces $\theta(f(n)) = \theta(g(n))$,

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Regla del límite (orden exacto)

Sean $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$. Entonces:

- Si $\lim(f(n)/g(n)) \in \mathbb{R}^+$, entonces $f(n) \in \theta(g(n))$
- Si $\lim(f(n)/g(n)) = 0$, entonces $f(n) \in O(g(n))$ pero $f(n) \notin \theta(g(n))$
- Si $\lim(f(n)/g(n)) = +\infty$, entonces $f(n) \in \Omega(g(n))$ pero $f(n) \notin \theta(g(n))$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observación
 - En ocasiones es fácil demostrar que $t(n) \in O(f(n))$ para un subconjunto infinito de números naturales (por ejemplo, de la forma $n = 2^i$)
 - Es preciso extender la notación para dar cuenta de estos casos

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (*Orden condicionado de $f(n)$*)

Sea $f: \mathbb{N} \rightarrow \mathbb{R}^+$ una función de los números naturales en los reales no negativos, y sea el predicado $P: \mathbb{N} \rightarrow \{V, F\}$ una propiedad de los números naturales; el *orden de $f(n)$ condicionado a $P(n)$* ($O(f(n)|P(n))$) es el conjunto de todas las funciones $t: \mathbb{N} \rightarrow \mathbb{R}^+$ tales que existen una constante real c y un número natural n_0 que cumplen que $t(n) \leq c \cdot f(n)$ para todo $n \geq n_0$ tal que $P(n)$.

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- En notación de conjuntos

$$O(f(n)|P(n)) = \{t: N \rightarrow R^+ / \exists c \in R^+, n_0 \in N (P(n) \Rightarrow t(n) \leq c \cdot f(n) \forall n \geq n_0)\}$$

- Por ejemplo, se puede afirmar que $2^n \in O(4^n | n=2^i)$, es decir que $2^n \in O(4^n)$ siempre que n sea potencia de 2

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observación
 - El *objetivo* es

si se puede demostrar la pertenencia a un orden para un subconjunto infinito de números naturales (relativamente dispersos), extender la afirmación a todos los naturales

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (función *asintóticamente no decreciente*)

Una función $f: \mathbb{N} \rightarrow \mathbb{R}^+$ es asintóticamente no decreciente si existe un $n_0 \in \mathbb{N}$ tal que $f(n) \leq f(n+1)$ para todo $n \geq n_0$

(equivalente a afirmar $f(n) \leq f(m)$ para todo $m \geq n \geq n_0$)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (función *b-uniforme*)

Sea $b \geq 2$, $b \in \mathbb{N}$; una función $f: \mathbb{N} \rightarrow \mathbb{R}^+$ es *b-uniforme* si es asintóticamente no decreciente y además cumple que $f(b \cdot n) \in O(f(n))$, es decir, si además $\exists c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$ tal que $f(b \cdot n) \leq c \cdot f(n) \forall n \geq n_0$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Definición (función *uniforme*)

Una función $f: \mathbb{N} \rightarrow \mathbb{R}^+$ es *uniforme*, de ajuste o suave si es b -uniforme para todo $b \geq 2$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Ejemplos
 - Las funciones $\log(n)$, $n \cdot \log(n)$, n^2 y los polinomios con primer coeficiente positivo son uniformes
 - Las funciones $n^{\log(n)}$, 2^n o $n!$ no son uniformes

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

- Observaciones
 - Las funciones uniformes son funciones de crecimiento moderado
 - Si f es b -uniforme para algún entero $b \geq 2$, entonces es uniforme

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

5. Notación asintótica

Regla de la uniformidad

Sean $f: \mathbb{N} \rightarrow \mathbb{R}^+$ una función uniforme y sea $t: \mathbb{N} \rightarrow \mathbb{R}^+$ una función asintóticamente no decreciente, y sea $b \geq 2$, $b \in \mathbb{N}$; si $t(n) \in O(f(n)|n=b^i)$, entonces $t(n) \in O(f(n))$

5. Notación asintótica

- Observación
 - La regla de la uniformidad también se cumple para órdenes inferiores y exactos

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Composición secuencial
 - Si dos partes de un algoritmo $A = P_1; P_2$ poseen tiempos de ejecución t_1 y t_2 respectivamente, el tiempo de ejecución de A es $t_A = t_1 + t_2$
 - Si P_1 y P_2 son independientes, $t_A \in \theta(t_1 + t_2)$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Bucles “para”

```
para i = 1 hasta m  
hacer P(i)  
equivale a  
i = 1  
mientras i ≤ m  
hacer P(i)  
i = i + 1
```

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Bucles “para”
 - Supongamos $m: \mathbb{N} \rightarrow \mathbb{R}^+$ función de n , y t el tiempo de ejecución de $P(i)$ para todo $i \Rightarrow$ el bucle tiene un tiempo en $\theta(t.m(n))$
 - Si t no es constante, sino que $t: \mathbb{N} \rightarrow \mathbb{R}^+$ función de n , pero tal que el tiempo de $P(i)$ está acotado por $t(n)$ para todo i , entonces el bucle está en $O(t(n).m(n))$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Funciones recursivas
 - De calcula su complejidad por medio de recurrencias
 - Se obtiene una recurrencia y se resuelve

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Función de Fibonacci

```
funcion fibonacci(n: entero): entero  
  si n < 2  
    entonces devolver n  
  sino devolver fibonacci(n-1) + fibonacci(n-2)
```

- Complejidad? => Recurrencia?

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Función de Fibonacci (complejidad)

$$t(0) = 1; t(1) = 1;$$

$$t(n) = t(n-1) + t(n-2) + h(n)$$

con $h(n) =$ tiempo de la suma = cte?

=> (técnicas de resolución de recurrencias) =>

$$t(n) \in \theta(\alpha^n) \text{ siendo } \alpha = \text{razón áurea } (1 + \sqrt{5})/2$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Bucles “mientras” y “repetir”
 - Dos formas de estudiarlos
 - Intentar acotar el número de veces que se ejecutan las instrucciones interiores buscando una función de las variables del bucle, que se decremente en cada pasada
 - Tratar los bucles como funciones recursivas

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

```
Función buscar(T: vector[1..n] de entero, x: entero): entero
{ Suponemos por simplicidad que x aparece en T }
  i = 1; j = n
  mientras i < j
    hacer k = (i + j) div 2
      caso      x < T[k]: j = k - 1
              x = T[k]: i = k; j = k
              x > T[k]: i = k + 1
    devolver i
```

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Análisis por medio de una función decreciente
 - Una cantidad que decrece es el número de componentes consideradas en cada vuelta, es decir $d = j - i + 1$
 - Si estamos en la vuelta s , $d_{s+1} \leq d_s/2$
 - Se cumple que $t(n) \in O(\log n)$ (incluso $\theta(\log n)$)

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Análisis como función recursiva
 - $t(n)$ = tiempo máximo cuando quedan a lo sumo n elementos
 - $t(1) = a$
 - $t(n) = b + t(n \text{ div } 2)$ si $n > 1$
 - Se demuestra que $t(n) \in O(\log n)$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Barómetro
 - Es una instrucción que se ejecuta *al menos* tantas veces como cualquier otra del algoritmo
 - Sirve para no tener que contar *todas* las instrucciones
 - Típicamente, es la instrucción de control de un bucle

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

```
Función fibiterativo(n: entero): entero  
  i = 1  
  j = 0  
  para k = 1 hasta n { k ≤ n }  
    hacer   j = i + j  
          i = j - i  
  devolver j
```

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

6. Análisis de las estructuras de control

- Estudio del caso medio de la ordenación por inserción
 - Suponemos que las $n!$ ordenaciones de T son *equiprobables*
 - Barómetro = $(j > 0 \text{ y } x < T[j])$
 - Concluimos que $t(n) \in \theta(n^2)$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Una recurrencia es la definición de una función en términos de si misma
- Aparecen típicamente al calcular la complejidad de
 - algoritmos recursivos (especialmente de tipo *divide y vencerás*)
 - bucles de tipo “mientras” y “repetir”

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Múltiples tipos de recurrencias
 - Homogéneas
 - Homogéneas con raíces múltiples
 - No homogéneas básicas
 - No homogéneas generalizadas
 - Con cambio de variable
 - Tipo “divide y vencerás”
 - Otras

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Recurrencias homogéneas

$$a_0 t_n + a_1 t_{n-1} + \dots + a_k t_{n-k} = 0$$

- Resolución
 - Construcción del polinomio característico
 - Factorización
 - Cálculo de las constantes

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Recurrencias homogéneas (raíces simples)
 - Polinomio característico ($t_n = x^k$)
$$P(x) = a_0 x^k + a_1 x^{k-1} + \dots + a_k = (x - r_1) \dots (x - r_k)$$
 - Si r_i es raíz de $P(x)$, entonces r_i^n es solución de t_n
 - Solución general
$$t_n = c_1 r_1^n + \dots + c_k r_k^n$$
 - Cálculo de c_1, \dots, c_k usando t_1, \dots, t_k

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Ejemplo

$$t(0) = 0$$

$$t(1) = 5$$

$$t(n) = 3t(n-1) + 4t(n-2) \text{ si } n > 1$$

$$\Rightarrow t(n) = 4^n - (-1)^n$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Recurrencias homogéneas (raíces múltiples)
 - Polinomio característico ($t_n = x^k$)
$$P(x) = a_0x^k + a_1x^{k-1} + \dots + a_k = (x - r_1)^{m_1} \dots (x - r_s)^{m_s}$$
 - Si r_i es raíz múltiple de $P(x)$, entonces $r_i^n, n.r_i^n, \dots, n^{m_i-1}.r_i^n$ son solución de t_n
 - Solución general
$$t_n = c_{10}r_1^n + c_{11}n.r_1^n + \dots + c_{1m_1}n^{m_1-1}r_1^n + \dots + c_{sm_s}n^{m_s-1}r_s^n$$
 - Cálculo de c_{10}, \dots, c_{sm_s} usando t_1, \dots, t_k

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Ejemplo

$$t(0) = 0$$

$$t(1) = 1$$

$$t(2) = 2$$

$$t(n) = 5t(n-1) - 8t(n-2) + 4t(n-3) \text{ si } n > 2$$

$$\Rightarrow t(n) = 2^{n+1} - n2^{n-1} - 2$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Recurrencias no homogéneas básicas

$$a_0t_n + a_1t_{n-1} + \dots + a_k t_{n-k} = b^n \cdot Q(n)$$

con $Q(n)$ de grado d

- Resolución
 - Similar a las homogéneas

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Recurrencias no homogéneas básicas
 - Polinomio característico ($t_n = x^k$)
$$P(x) = (a_0x^k + a_1x^{k-1} + \dots + a_k)(x - b)^{d+1}$$
 - El polinomio de podría obtener operando sobre $t(n)$, $t(n-1)$, etc.
 - Resto de pasos idénticos a las homogéneas

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Ejemplo

$$t(n) = 2t(n-1) + (n+5)3^n$$

$$P(x) = (x - 2)(x - 3)^2$$

$$\Rightarrow t(n) = (t(0) - 9)2^n + (n+3)3^{n+1}$$

7. Resolución de recurrencias

- Recurrencias no homogéneas generalizadas

$$a_0 t_n + a_1 t_{n-1} + \dots + a_k t_{n-k} = b_1^n \cdot Q_1(n) + \dots + b_s^n \cdot Q_s(n)$$

con $Q_i(n)$ de grado d_i

- Resolución
 - Similar a las no homogéneas básicas

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Recurrencias no homogéneas generalizadas
 - Polinomio característico ($t_n = x^k$)
$$P(x) = (a_0 x^k + a_1 x^{k-1} + \dots + a_k)(x - b_1)^{d_1+1} \dots (x - b_s)^{d_s+1}$$
 - Resto de pasos idénticos a las básicas

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Ejemplo

$$t(0) = 0$$

$$t(n) = 2t(n-1) + n + 2^n \text{ si } n > 0$$

$$P(x) = (x - 2)^2(x - 1)^2$$

$$\Rightarrow t(n) = n \cdot 2^n + 2^{n+1} - n - 2$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Con cambio de variable
 - A veces, $t(n)$ en función de $t(n/2)$, ó $t(n/3)$
 - Se efectúa el cambio $n = 2^i$, ó $n = 3^i$
 - Se obtiene el orden de la recurrencia condicionado a $n = 2^i$, ó $n = 3^i$
 - Se aplica la *regla de la uniformidad* y se obtiene el orden incondicional

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Ejemplo

$$t(1) = 1$$

$$t(n) = 3t(n/2) + n \text{ si } n > 1, \text{ potencia de } 2$$

Cambio $n = 2^i$

$$\Rightarrow t(n) = 3n^{\log_3} - 2n \text{ para } n \text{ potencia de } 2$$

$$t(n) \in O(n^{\log_3}) | n=2^i \Rightarrow (R. UNIF.) \Rightarrow t(n) \in O(n^{\log_3})$$

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Tipo “divide y vencerás”
 - Responden a la forma

$$t(n) = l.t(n/b) + c.n^k \text{ si } n > n_0$$

con $l, k > 0, b > 1, c \geq 0, n_0 > 0, y t: \mathbb{N} \rightarrow \mathbb{R}^+$
asintóticamente no decreciente, y n/n_0 una potencia de b

Algorítmica – José María Gómez Hidalgo, Francisco Carrero García

7. Resolución de recurrencias

- Tipo “divide y vencerás”
 - Se resuelven con el cambio de variable $n = n_0 \cdot b^i$
 - Se cumple

$$t(n) \in \theta(n^k) \quad \text{si } | < b^k$$

$$t(n) \in \theta(n^k \cdot \log n) \quad \text{si } | = b^k$$

$$t(n) \in \theta(n^{\log |}) \quad \text{si } | > b^k$$