DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

La eficiencia de los algoritmos

Análisis y Diseño de Algoritmos

La eficiencia de los algoritmos

- Comparación de algoritmos
- Principio de invarianza
- Eficiencia
- Notaciones asintóticas
- Cálculo de la eficiencia de un algoritmo
- Resolución de recurrencias:
  Método de la ecuación característica
  - Recurrencias homogéneas
  - Recurrencias no homogéneas
  - Cambios de variable
  - Transformaciones del rango
- Apéndice: Fórmulas útiles

Comparación de algoritmos

A menudo dispondremos de más de un algoritmo para resolver un problema dado, ¿con cuál nos quedamos?

USO DE RECURSOS

- Computacionales:
  - Tiempo de ejecución
  - Espacio en memoria
- No computacionales:
  - Esfuerzo de desarrollo (análisis, diseño & implementación)

Comparación de algoritmos

Factores que influyen en el uso de recursos

- Recursos computacionales:
  - Diseño del algoritmo
  - Complejidad del problema (p.ej. tamaño de las entradas)
  - Hardware (arquitectura, MHz, MBs…)
  - Calidad del código
  - Calidad del compilador (optimizaciones)
- Recursos no computacionales:
  - Complejidad del algoritmo
  - Disponibilidad de bibliotecas reutilizables

Principio de invarianza

- Dos implementaciones de un mismo algoritmo no diferirán más que en una constante multiplicativa.
- Si $t_1(n)$ y $t_2(n)$ son los tiempos de dos implementaciones de un mismo algoritmo, se puede comprobar que:

$$\exists c, d \in R, \quad t_1(n) \leq c t_2(n); \quad t_2(n) \leq d t_1(n)$$

Eficiencia

Medida del uso de los recursos computacionales requeridos por la ejecución de un algoritmo en función del tamaño de las entradas.

T(n) Tiempo empleado para ejecutar el algoritmo con una entrada de tamaño n

Eficiencia

Tipos de análisis

¿Cómo medimos el tiempo de ejecución de un algoritmo?

- **Mejor caso**
  En condiciones óptimas (no se usa por ser demasiado optimista).
- **Peor caso**
  En el peor escenario posible (nos permite acotar el tiempo de ejecución).
- **Caso promedio**
  Caso difícil de caracterizar en la práctica.
- **Análisis probabilístico**
  Asume una distribución de probabilidad sobre las posibles entradas.
- **Análisis amortizado**
  Tiempo medio de ejecución por operación
  sobre una secuencia de ejecuciones sucesivas.

Eficiencia

Ejemplo

- Algoritmo 1: $T(n) = 10^{-4}\, 2^n$ segundos
  $n = 38$ datos
  $T(n) = 1$ año

- Algoritmo 2: $T(n) = 10^{-2}\, n^3$ segundos
  $n = 1000$ bits
  $T(n) = 1$ año

¿Cuál es mejor? Se precisa un **análisis asintótico**

Notaciones asintóticas

Estudian el comportamiento del algoritmo cuando el tamaño de las entradas es lo suficientemente grande, sin tener en cuenta lo que ocurre para entradas pequeñas y obviando factores constantes.

Notaciones asintóticas

Orden de eficiencia

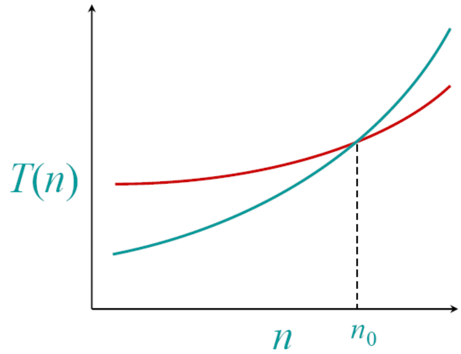
Un algoritmo tiene un tiempo de ejecución de **orden f(n)**, para una función dada *f*, si existe una constante positiva *c* y una implementación del algoritmo capaz de resolver cada caso del problema en un tiempo acotado superiormente por $c \cdot f(n)$, donde *n* es el tamaño del problema considerado.

Notaciones asintóticas

Notación O

Decimos que una función **T(n) es O(f(n))**
si existen constantes n_0 y c
tales que $T(n) \leq cf(n)$ para todo $n \geq n_0$:

$$\mathbf{T(n)\ es\ O(f(n)) \Leftrightarrow}$$
$$\mathbf{\exists c \in R,\ \exists n_0 \in N,\ tal\ que\ \forall n > n_0 \in N,\ T(n) \leq cf(n)}$$



Notaciones asintóticas

Ejemplos

T(n) = 3n

- T(n) es O(n), O(n log n), O(n^2), O(n^3) y O(2^n).

T(n) = $(n+1)^2$

- T(n) es O(n^2), O(n^3) y O(2^n).
- T(n) no es O(n) ni O(n log n).

T(n) = $32n^2 + 17n + 32$

- T(n) es O(n^2) pero no es O(n).

T(n) = $3n^3 + 345n^2$

- T(n) es O(n^3) pero no es O(n^2).

T(n) = 3^n

- T(n) es O(3^n) pero no es O(2^n).

Notaciones asintóticas

Notaciones Ω y Θ

Notación Ω (cota inferior)

T(n) es Ω(f(n)) cuando
$\exists c \in R^+, \exists n_0 \in N: \forall n \geq n_0 \Rightarrow T(n) \geq cf(n)$

Notación Θ (orden exacto)

T(n) es Θ(f(n)) cuando
T(n) es O(f(n)) y T(n) es Ω(f(n))

Órdenes de eficiencia

Órdenes de eficiencia más habituales

N	$O(\log_2 n)$	O(n)	$O(n \log_2 n)$	$O(n^2)$	$O(2^n)$	O(n!)
10	3 µs	10 µs	30 µs	0.1 ms	1 ms	4 s
25	5 µs	25 µs	0.1 ms	0.6 ms	33 s	10^{11} años
50	6 µs	50 µs	0.3 ms	2.5 ms	36 años	...
100	7 µs	100 µs	0.7 ms	10 ms	10^{17} años	...
1000	10 µs	1 ms	10 ms	1 s	...	...
10000	13 µs	10 ms	0.1 s	100 s	...	...
100000	17 µs	100 ms	1.7 s	3 horas	...	...
1000000	20 µs	1 s	20 s	12 días	...	...

Tiempos calculados suponiendo 1 µs por operación elemental.

$\mathbf{O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset O(2^n) \subset O(n!)}$

Órdenes de eficiencia

Orden lineal: O(n)

Tiempo de ejecución proporcional al tamaño de la entrada.

Ejemplo: Calcular el máximo de n números $a_1, \ldots, a_n$.

```
max ← a1
for i = 2 to n {
   if (ai > max)
      max ← ai
}
```


Órdenes de eficiencia

Orden cuadrático: O(n²)

Aparece cuando tenemos que enumerar todas las parejas posibles de elementos de un conjunto.

Ejemplo: Dado un conjunto de puntos en el plano $(x_1, y_1), \ldots, (x_n, y_n)$, encontrar la pareja más cercana.

```
min ← (x1 - x2)² + (y1 - y2)²
nearest ← (1,2)
for i = 1 to n {
   for j = i+1 to n {
      d ← (xi - xj)² + (yi - yj)²
      if (d < min) {
         min ← d
         nearest ← (i,j)
      }
   }
}
```

Órdenes de eficiencia

Órdenes O(log n) y O(n log n)

Aparecen en muchos algoritmos recursivos

p.ej. Estrategia "divide y vencerás"

Ejemplos:	**O(log n)**	Búsqueda binaria
	O(n log n)	Mergesort, Heapsort…

Orden exponencial $O(2^n)$

Aparece en muchos problemas de tipo combinatorio

Ejemplo: Enumerar todos los subconjuntos de un conjunto dado.

Cálculo de la eficiencia

Operación elemental

Operación de un algoritmo cuyo tiempo de ejecución se puede acotar superiormente por una constante.

- En nuestro análisis, sólo contará el número de operaciones elementales y no el tiempo exacto necesario para cada una de ellas.
- En la descripción de un algoritmo, puede ocurrir que una línea de código corresponda a un número de variable de operaciones elementales.

p.ej. `x ← max{A[k], 0 <= k < n}`

Cálculo de la eficiencia

Operaciones elementales

- Algunas operaciones matemáticas no son operaciones elementales.

  p.ej. sumas y productos
  dependen de la longitud de los operandos.

- Convención: En la práctica, suma, diferencia, producto, división, módulo, operaciones booleanas, comparaciones y asignaciones se consideran operaciones elementales, salvo que explícitamente se establezca lo contrario.

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

1. Sentencias simples
2. Bloques de sentencias
3. Sentencias condicionales
4. Bucles
5. Llamadas a funciones
6. Funciones recursivas

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

1. Sentencias simples

Consideraremos que cualquier sentencia simple
(lectura, escritura, asignación, etc.)
va a consumir un tiempo constante, **O(1)**,
salvo que contenga una llamada a una función.

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

2. Bloques de sentencias

- Tiempo total de ejecución = Suma de los tiempos de ejecución de cada una de las sentencias del bloque.
- Orden de eficiencia = Máximo de los órdenes de eficiencia de cada una de las sentencias del bloque.

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

3. Sentencias condicionales

El tiempo de ejecución de una sentencia condicional es el máximo del tiempo del bloque `if` y del tiempo del bloque `else`.

Si el bloque `if` es O(f(n)) y el bloque `else` es O(g(n)), la sentencia condicional será **O(max{f(n), g(n)})**.

¡Ojo! No se nos puede olvidar el tiempo de ejecución correspondiente a la comprobación de la condición.

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

4. Bucles

- Tiempo de ejecución
  = Suma de los tiempos invertidos en cada iteración
- En el tiempo de cada iteración se ha de incluir el tiempo de ejecución del cuerpo del bucle y también el asociado a la evaluación de la condición (y, en su caso, la actualización del contador).
- Si todas las iteraciones son iguales, el tiempo total de ejecución del bucle será el producto del número de iteraciones por el tiempo empleado en cada iteración.

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

5. Llamadas a funciones

Si una determinada función P tiene una eficiencia de O(f(n)), cualquier llamada a P es de orden O(f(n)).

- Las asignaciones con diversas llamadas a función deben sumar las cotas de tiempo de ejecución de cada llamada.

- La misma consideración es aplicable a las condiciones de bucles y sentencias condicionales.

Cálculo de la eficiencia

Reglas de cálculo de la eficiencia

6. Funciones recursivas

Las funciones de tiempo asociadas a funciones recursivas son también recursivas.

p.ej. $T(n) = T(n-1) + f(n)$

- Para analizar el tiempo de ejecución de un algoritmo recursivo, le asociamos una función de eficiencia desconocida, T(n), y la estimamos a partir de T(k) para distintos valores de k (menores que n).

Cálculo de la eficiencia

Ejemplo

```
int fact(int n)
{
  if (n <= 1)
     return 1;
  else
     return (n * fact(n - 1));
 }
```

Los bloques `if` y `else` son operaciones elementales, por lo que su tiempo de ejecución es O(1).

Cálculo de la eficiencia

$$
\begin{aligned}
T(n) &= 1 + T(n - 1) \\
&= 1 + (1 + T(n-2)) = 2 + T(n-2) \\
&= 2 + (1 + T(n-3)) = 3 + T(n-3) \\
&\dots \\
&= i + T(n-i) \\
&\dots \\
&= (n-1) + T(n - (n-1)) = (n-1) + 1
\end{aligned}
$$

Por tanto, T(n) es O(n)

La implementación recursiva del cálculo del factorial es de orden lineal.

Cálculo de la eficiencia

Ejemplo

```
int L(int n)
{
  if (n == 1)
     return 0;
  else
     return L(n/2) + 1;
}
```

$$T(n) = \begin{cases} 1, & n = 1 \\ 1 + T\left(\dfrac{n}{2}\right) & n > 1 \end{cases}$$

Cálculo de la eficiencia

$$\begin{aligned} T(n) &= T(n/2) + 1 \\ &= (T(n/4) + 1) + 1 = T(n/4) + 2 \\ &= (T(n/8) + 1) + 2 = T(n/8) + 3 \\ &\quad \ldots \\ &= T(n/2^{i}) + i \\ &\quad \ldots \\ &= T(n/2^{\log(n)}) + \log_2(n) \\ &= T(1) + \log_2(n) \end{aligned}$$

Por tanto, T(n) es O(log n)

Cálculo de la eficiencia

El mismo desarrollo se podría realizar más cómodamente con un cambio de variable: $n=2^m \rightarrow m = \log_2(n)$

$$
\begin{aligned}
T(2^m) &= T(2^{m-1}) + 1 = \\
&= (T(2^{m-2}) + 1) + 1 \\
&= (T(2^{m-3}) + 1) + 2 \\
&\ldots \\
&= T(2^{m-i}) + i \\
&\ldots \\
&= T(2^{m-m}) + m \\
&= T(1) + \log_2(n)
\end{aligned}
$$

Cálculo de la eficiencia

Ejemplo

$$T(n) = T\left(\frac{n}{2}\right) + n^2, \quad n \geq 2; \quad T(1) = 1$$

- Cambio de variable: $n = 2^m$

  $n^2 = (2^m)^2 = (2^2)^m = 4^m$

$$
\begin{aligned}
T(2^m) &= T(2^{m-1}) + 4^m = \\
&= T(2^{m-2}) + 4^{m-1} + 4^m \\
&\ldots \\
&= T(2^{m-i}) + [4^{m-(i-1)} + \ldots + 4^{m-1} + 4^m]
\end{aligned}
$$

Cálculo de la eficiencia

Ejemplo

$$T(n) = T\left(\frac{n}{2}\right) + n^2, \quad n \geq 2; \quad T(1) = 1$$

$$T(2^m) = T(1) + [4^1 + \cdots + 4^{m-2} + 4^{m-1} + 4^m]$$

$$= 1 + \sum_{i=1}^{m} 4^i = \sum_{i=0}^{m} 4^i = \frac{4^{m+2} - 1}{4 - 1} = \frac{4^2 4^m}{3} - \frac{1}{3}$$

$$= \frac{4^2}{3} n^2 - \frac{1}{3}$$

T(n) es O(n^2)

Resolución de recurrencias

Técnicas de resolución de recurrencias

- Método de sustitución
  1. Desarrollar la expresión
  2. Adivinar la fórmula de la expresión
  3. Demostrar por inducción
  4. Resolver constantes

- Árbol de recursividad
  Representación gráfica "intuitiva".

- Expansión de recurrencias
  Método algebraico equivalente al árbol de recursividad.

- Método de la ecuación característica

Método de sustitución

$$T(n) = \begin{cases} O(1) & si\ n = 1 \\ 4T(n/2) + n & si\ n > 1 \end{cases}$$

¿T(n) es $O(n^3)$?

Suponemos que $T(k) \leq ck^3$ para $k<n$

Demostramos por inducción que $T(n) \leq cn^3$

$T(n) = 4T(n/2) + n$

$\leq 4c(n/2)^3 + n = (c/2)n^3 + n = cn^3 - ((c/2)n^3 - n)$

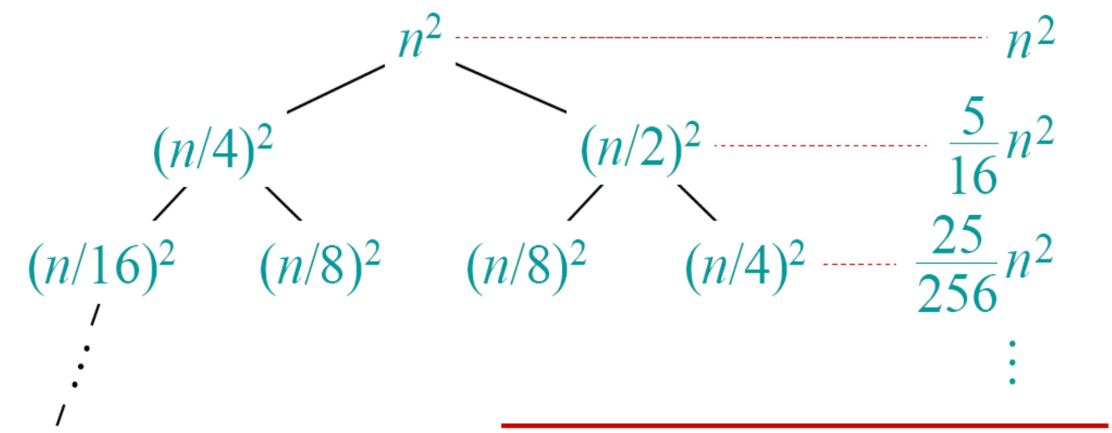
$\leq cn^3$ siempre que $((c/2)n^3 - n)>0$ (p.ej. $c\geq 2$ y $n\geq 1$)

PROBLEMA: ¿Podríamos encontrar una cota superior más ajustada?

Sugerencia: Probar con $T(n) \leq cn^2$ y $T(n) \leq c_1n^2-c_2n$

Árbol de recursividad

$$T(n) = T(n/4) + T(n/2) + n^2$$



$$\text{Total} = n^2\left(1 + \tfrac{5}{16} + \left(\tfrac{5}{16}\right)^2 + \left(\tfrac{5}{16}\right)^3 + \cdots\right)$$
$$= O(n^2)$$

Ecuación característica

Recurrencias homogéneas lineales con coeficientes constantes

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

- Lineal: No hay términos $t_{n-1}t_{n-j}$, t^2_{n-i}
- Homogénea: Igualadas a 0
- Con coeficientes constantes: a_i constantes

Ejemplo: Sucesión de Fibonacci

$$f_n = f_{n-1} + f_{n-2} \quad \Rightarrow \quad f_n - f_{n-1} - f_{n-2} = 0$$

Ecuación característica

Suposición: $t_n = x^n$

$$a_0 x^n + a_1 x^{n-1} + \cdots + a_k x^{n-k} = 0$$

Se satisface si x=0 o bien

$$a_0 x^k + a_1 x^{k-1} + \cdots + a_k = 0$$

ecuación característica

Polinomio característico

$$p(x) = a_0 x^k + a_1 x^{k-1} + \cdots + a_k$$

Ecuación característica

Teorema Fundamental del Álgebra:

$$p(x) = \prod_{i=1}^{k} (x - r_i)$$

Consideremos una raíz del polinomio característico, r_i

$p(r_i) = 0$

Por tanto, $\mathbf{r_i^n}$ es solución de la recurrencia.
Además, las combinaciones lineales de las soluciones de la recurrencia también son soluciones:

$$t_n = \sum_{i=1}^{k} c_i r_i^n$$

Cuando todos los r_i son distintos, éstas son las únicas soluciones

Ecuación característica

Resolución de recurrencias mediante el método de la ecuación característica:

1. Se obtiene la ecuación característica asociada a la recurrencia que describe el tiempo de ejecución T(n).
2. Se calculan las k raíces del polinomio característico.
3. Se obtiene la solución a partir de las raíces del polinomio característico.
4. Se determinan las constantes a partir de las k condiciones iniciales.

Ecuación característica

Ejemplo: Sucesión de Fibonacci

$$f_n = \begin{cases} n, & \text{si } n = 0 \text{ ó } n = 1 \\ f_{n-1} + f_{n-2} & \text{en otro caso} \end{cases}$$

$$f_n - f_{n-1} - f_{n-2} = 0$$

$$p(x) = x^2 - x - 1 \qquad r_1 = \frac{1+\sqrt{5}}{2} \quad y \quad r_2 = \frac{1-\sqrt{5}}{2}$$

Ecuación característica

Ejemplo: Sucesión de Fibonacci

$$f_n = c_1 r_1^n + c_2 r_2^n$$

$$\left.\begin{array}{ccccc} c_1 & + & c_2 & = & 0 \\ r_1 c_1 & + & r_2 c_2 & = & 1 \end{array}\right\} \Rightarrow \quad c_1 = \frac{1}{\sqrt{5}} \quad y \quad c_2 = -\frac{1}{\sqrt{5}}$$

$$f_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Ecuación característica

Ejemplo:

$$t_n = \begin{cases} 0, & n = 0 \\ 5, & n = 1 \\ 3t_{n-1} + 4t_{n-2}, & \text{en otro caso} \end{cases}$$

$$t_n - 3t_{n-1} - 4t_{n-2} = 0$$

$$x^2 - 3x - 4 = (x+1)(x-4)$$

$$t_n = c_1(-1)^n + c_2 4^n$$

A partir de las condiciones iniciales: $c_1 = -1$, $c_2 = 1$

Ecuación característica

Raíces múltiples

Supongamos que las raíces del polinomio característico NO son todas distintas.

r_i con multiplicidad m_i

$$t_n = \sum_{i=1}^{l} \sum_{j=0}^{m_i - 1} c_{ij} n^j r_i^n$$

Ecuación característica

Ejemplo:

$$t_n = \begin{cases} n, & \text{si } n = 0, 1 \text{ ó } 2 \\ 5t_{n-1} - 8t_{n-2} + 4t_{n-3} & \text{en otro caso} \end{cases}$$

$$t_n - 5t_{n-1} + 8t_{n-2} - 4t_{n-3} = 0$$

$$x^3 - 5x^2 + 8x - 4 = (x-1)(x-2)^2$$

$$t_n = c_1(1)^n + c_2 2^n + c_3 n 2^n$$

A partir de las condiciones iniciales: c_1 = -2, c_2 = 2, c_3 = -1/2

$$t_n = 2^{n+1} - n2^{n-1} - 2$$

Ecuación característica

Recurrencias no homogéneas

A partir de $a_0 t_n + a_1 t_{n-1} + \cdots + a_k t_{n-k} = b^n p(n)$

donde b es una constante y p(n) un polinomio de grado d

derivamos el polinomio característico

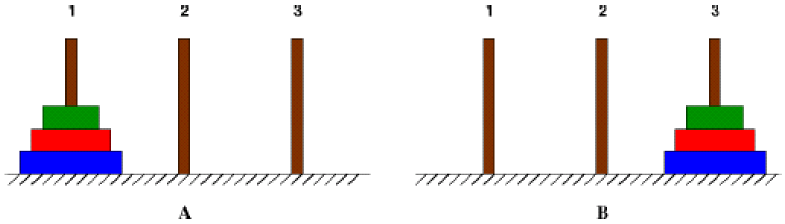
$$p(x) = (a_0 x^k + a_1 x^{k-1} + \cdots + a_k)(x-b)^{d+1}$$

que resolveremos igual que en el caso homogéneo.

Ecuación característica

Ejemplo:

Las torres de Hanoi



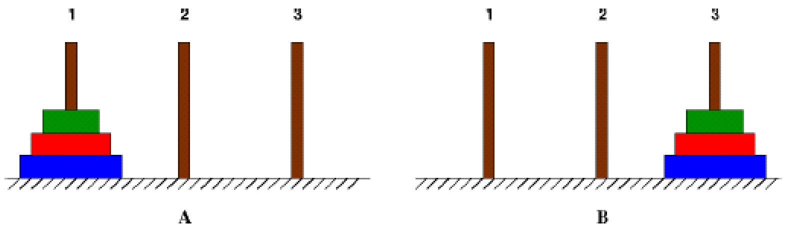

```
hanoi (int n, int inicial, int final, int tmp)
{
     hanoi (n - 1, inicial, tmp, final)
     final ← inicial
     hanoi (n - 1, tmp, final, inicial)
}
```

$$T(n) = \begin{cases} 0, & n = 0 \\ 2T(n-1)+1, & \text{en otro caso} \end{cases}$$

Ecuación característica

Ejemplo:

Las torres de Hanoi



$$t_n - 2t_{n-1} = 1$$

$$p(x) = (x-2)(x-1)$$

$$t_n = c_1 2^n + c_2 1^n \qquad T(n) = c_1 2^n + c_2$$

$$\left.\begin{array}{l} T(0)=0 \\ T(1)=1 \end{array}\right\} \Rightarrow \left.\begin{array}{ccccc} c_1 & + & c_2 & = & 0 \\ 2c_1 & + & c_2 & = & 1 \end{array}\right\} \Rightarrow \begin{array}{l} c_1 = 1 \\ c_2 = -1 \end{array}$$

$$T(n) = 2^n - 1$$

Ecuación característica

Ejemplo:

$$t_n = 2t_{n-1} + n; \qquad t_0 = 0$$

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

$$t_n = c_1 2^n + c_2 1^n + c_3 n 1^n$$

$$\left.\begin{array}{ccccccc} c_1 & + & c_2 & & & = & 0 \\ 2c_1 & + & c_2 & + & c_3 & = & 1 \\ 4c_1 & + & c_2 & + & 2c_3 & = & 4 \end{array}\right\} \Rightarrow \begin{array}{c} c_1 = 2 \\ c_2 = -2 \\ c_3 = -1 \end{array}$$

$$t_n = 2^{n+1} - n - 2$$

Ecuación característica

Recurrencias no homogéneas

$$a_0 t_n + a_1 t_{n-1} + \cdots + a_k t_{n-k} = b_1^n p_1(n) + \ldots + b_s^n p_s(n)$$

- $b_1, \ldots, b_s$ constantes
- $p_i(n)$ polinomio de grado d_i

Polinomio característico:

$$p(x) = (a_0 x^k + a_1 x^{k-1} + \cdots + a_k)(x-b_1)^{d_1+1} \cdots (x-b_s)^{d_s+1}$$

Ecuación característica

Cambios de variable

Cuando las recurrencias no se ajustan al tipo conocido…

$$T(n) = \begin{cases} 1, & n = 1 \\ 3T(n/2) + n, & n > 1 \end{cases}$$

aplicamos un cambio de variable $n = 2^i$

que nos permite definir una nueva recurrencia:

$$t_i = T(2^i)$$

Ecuación característica

Cambios de variable

$$t_i = 3t_{i-1} + 2^i \quad \Rightarrow \quad t_i = c_1 3^i + c_2 2^i$$

Finalmente, deshacemos el cambio de variable: $i = \log_2(n)$

$$T(n) = c_1 3^{\log_2(n)} + c_2 2^{\log_2(n)} = c_1 n^{\log_2(3)} + c_2 n$$

$$T(n) \text{ es } \mathrm{O}(n^{\log_2(3)})$$

Ecuación característica

Ejemplo

$$T(n)=\begin{cases} 1, & n=1 \\ 4T(n/2)+n, & n>1 \end{cases}$$

$$t_i=T(2^i) \qquad t_i=4t_{i-1}+2^i \qquad \Rightarrow \qquad t_i=c_1 4^i+c_2 2^i$$

$$T(n)=c_1 4^{\log_2(n)}+c_2 2^{\log_2(n)}=c_1 n^{\log_2(4)}+c_2 n=c_1 n^2+c_2 n$$

$$T(n) \text{ es } \mathrm{O}(n^2)$$

Ecuación característica

Ejemplo

$$T(n)=\begin{cases} 1, & n=1 \\ 2T(n/2)+n, & n>1 \end{cases}$$

$$t_i=T(2^i) \qquad t_i=2t_{i-1}+2^i \qquad \Rightarrow \qquad t_i=c_1 2^i+c_2 i 2^i$$

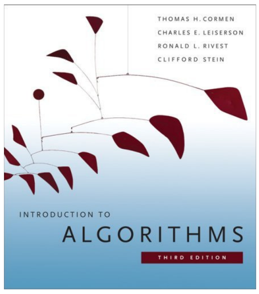
$$T(n)=c_1 2^{\log_2(n)}+c_2\log_2(n)2^{\log_2(n)}=c_1 n+c_2 n\log_2(n)$$

$$T(n) \text{ es } \mathrm{O}(n\log_2 n)$$

Ecuación característica

Recurrencia "divide y vencerás"

a.k.a. "fórmula maestra"



$$T(n) = aT\left(\frac{n}{b}\right) + cn^k$$

$$\text{con } a \geq 1, b \geq 2, k \geq 0, c > 0$$

$$T(n) = \begin{cases} \Theta(n^k), & a < b^k \\ \Theta(n^k \log_b(n)), & a = b^k \\ \Theta(n^{\log_b(a)}) & a > b^k \end{cases}$$

Ecuación característica

Transformaciones del rango

$$T(n) = \begin{cases} 1/3, & n = 1 \\ nT^2\left(\frac{n}{2}\right), & \text{en otro caso} \end{cases}$$

Inicialmente, realizamos un cambio de variable:

$$t_i = T(2^i) = 2^i T^2(2^{i-1}) = 2^i t_{i-1}^2$$

Como esta recurrencia no es lineal y el coeficiente 2^i no es constante, tomamos logaritmos:

$$u_i = \log(t_i) = i + 2\log(t_{i-1}) = i + 2u_{i-1}$$

Ecuación característica

Transformaciones del rango

$$u_i = i + 2u_{i-1}$$
$$u_i - 2u_{i-1} = i$$
$$u_i = c_1 2^i + c_2 1^i + c_3 i 1^i$$

Sustituyendo en la recurrencia para u_i, obtenemos:

$$i = u_i - 2u_{i-1} = c_1 2^i + c_2 + c_3 i - 2(c_1 2^{i-1} + c_2 + c_3(i-1))$$
$$i = (2-i)c_3 - c_2$$
$$(1+c_3)i = 2c_3 - c_2 \qquad \boxed{c_3 = -1; \quad c_2 = -2}$$

Ecuación característica

Transformaciones del rango

$$u_i = c_1 2^i - i - 2$$

Deshacemos los cambios:

$$t_i = 2^{u_i} = 2^{c_1 2^i - i - 2}$$
$$T(n) = t_{\log_2(n)} = 2^{c_1 n - \log_2 n - 2} = \frac{2^{c_1 n}}{4n}$$

A partir de las condiciones iniciales:

$$T(1) = 2^{c_1}/4 = 1/3 \Rightarrow c_1 = \log_2(4/3) = 2 - \log_2 3$$
$$T(n) = \frac{2^{c_1 n}}{4n} = \frac{2^{(2-\log_2 3)n}}{4n} = \frac{2^{2n}}{4n 2^{(\log_2 3)n}} = \frac{2^{2n}}{4n3^n}$$

Apéndice: Fórmulas útiles

Progresiones aritméticas

$$a_{i+1} = a_i + d$$

$$\sum_{i=1}^{n} a_i = \frac{1}{2} n (a_1 + a_n)$$

$$\sum_{i=1}^{n} i = \frac{1}{2} n (n+1)$$

Apéndice: Fórmulas útiles

Progresiones geométricas

$$a_{i+1} = r\, a_i$$

$$\sum_{i=1}^{n} a_i = \frac{a_1 (r^{n+1} - 1)}{r - 1}$$

$$\sum_{i=1}^{n} b^i = \frac{b^{n+1} - b}{b - 1}$$

Apéndice: Fórmulas útiles

Sumatorias

$$\sum_{i=1}^{n} a = na$$

$$\sum_{i=1}^{n} af(i) = a\sum_{i=1}^{n} f(i)$$

$$\sum (a+b) = \sum a + \sum b$$

$$\sum_i \sum_j a_i b_j = \sum_i a_i \sum_j b_j$$

Apéndice: Fórmulas útiles

Sumatorias

$$\sum_{i=1}^{n} i = \frac{1}{2} n(n+1)$$

$$\sum_{i=1}^{n} i^2 = \frac{1}{6} n(n+1)(2n+1)$$

$$\sum_{i=1}^{n} i^3 = \left[\frac{1}{2} n(n+1)\right]^2$$

$$\sum_{i=1}^{n} i(i+1) = \frac{1}{3} n(n+1)(n+2)$$

Apéndice: Fórmulas útiles

Potencias

$$x^{y+z} = x^y \cdot x^z$$

$$x^{y-z} = x^y / x^z$$

$$x^{y \cdot z} = (x^y)^z = (x^z)^y$$

$$x^{y^z} \neq x^{z^y}$$ ¡Ojo!

Apéndice: Fórmulas útiles

Logaritmos

$$\log_a(n) = \frac{\log_b(n)}{\log_b(a)}$$

$$\log_a(n\,m) = \log_a(n) + \log_a(m)$$

$$\log_a(n/m) = \log_a(n) - \log_a(m)$$

$$\log_a(n^p) = p \log_a(n)$$

$$n^{\log_a(m)} = m^{\log_a(n)}$$