Tema 1. Análisis de Algoritmos

Tabla de Contenidos

Método de resolución de un problema

- Descripción de pasos discretos, basada en unos fundamentos y resultados teóricos, de forma que la realización de estos pasos, junto a unos datos, lleva a una solución del problema

Semialgoritmo

- Descripción precisa de un método, con pasos numerados y bien ordenados que, si existe, da la solución al problema en un número finito de pasos

Algoritmo

- Semialgoritmo que es capaz de determinar que un problema no tiene solución en un número finito de pasos

Programa

- Algoritmo escrito en formato comprensible para un computador junto con las estructuras de datos que maneja

Ejemplo: $a + x = b,\ x \in N$

```
1. x ← a
2. Si x=b parar
3. x ← x + 1
4. Ir a 2
```

Si a > b nunca termina

$\Rightarrow$ Semialgoritmo

Mediremos la complejidad de los algoritmos

Complejidad

- Medida de la eficiencia de un algoritmo (en tiempo o espacio)
- Indica cómo es su comportamiento
  - Acotando el comportamiento sabremos cómo es su complejidad

Estrategias para medir la complejidad

- Empírica (mediante el computador)
- Teórica (o analítica): independiente del computador, programador o lenguaje
  - Basado en el principio de invarianza que dice que la diferencia de ejecutar un programa en una máquina u otra será multiplicarlo por una constante
- Híbrida

Tamaño y Func. Complejidad

Tamaño del problema

- Variable utilizada para las funciones de complejidad
- Suele depender del número de datos del problema
  - Ejemplo: Ordenar *n* número ⇒ **tamaño *n***

Función de complejidad

- Mide el tiempo/espacio relativo
- Proporciona el orden de complejidad (comportamiento)
  - Forma en que crece la complejidad
- Tiene como variable independiente el tamaño del problema
- El comportamiento determina la eficiencia
- No es única para un algoritmo
- **Las más representativas**
  - Mejor caso ⇒ $f_m(n)$
  - Peor caso ⇒ $f_p(n)$
  - Esperanza matemática ⇒ $f_a(n)$

$$f_m(n) < f_a(n) < f_p(n)$$

Funciones y comportamientos más usuales

- 1 O(1)
- log n O(log n)
- n O(n)
- n^2 $O(n^2)$
- n^3 $O(n^3)$
- .....
- 2^n $O(2^n)$
- $n \cdot 2^n$ $O(n \cdot 2^n)$

Ejemplo: *Ordenación por inserción directa*

```
for (i=2: i<=n; i++){
      x = a_i;
      a_0 = x;
      j = i-1;
      while (x<a_j) {
            a_j+1 = a_j:
            j = j-1;
      }
      a_j+1 = x
}
```

$$T(n) = \sum_{i=2}^{n} \left(1+1+1+ \left(\sum_{j=j-1}^{x<a_j} 2 \right) +1 \right)$$

Ejemplo: *Ordenación por inserción directa*

Caso mejor ⇒ O(n)

$$T(n)=\sum_{i=2}^{n}\left(1+1+1+\left(\sum_{j=j-1}^{x<a_j}2\right)+1\right)=\sum_{i=2}^{n}\left(1+1+1+1\right)=4(n-1)$$

Caso medio ⇒ O(n²)

$$T(n)=a(n-1)+2\sum_{i=2}^{n}\frac{i}{2}=4(n-1)+\frac{(n+2)(n-1)}{2}$$

Ejemplo: *Ordenación por inserción directa*

Caso peor $\Rightarrow O(n^2)$

$$T(n)=\sum_{i=2}^{n}\left(1+1+1+\left(\sum_{j=j-1}^{x<a_j}2\right)+1\right)=4(n-1)+2\sum_{i=2}^{n}\left(\sum_{j=j-1}^{x<a}1\right)=$$

$$=4(n-1)+2\sum_{i=2}^{n}(i-1)=4(n-1)+2\frac{1+(n-1)}{2}(n-1)=$$

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

Prob. tratable e intratable

Convenio de Edmonds

- Problema **tratable** $\Rightarrow$ complejidad *polimonial*
- Problema **intratable** $\Rightarrow$ complejidad *exponencial*

Consideraciones

- Los algoritmos con complejidad mayor que $n \cdot \log n$ son poco prácticos
- Los algoritmos con complejidad exponencial solo son válidos para problemas de tamaño pequeño

Notaciones simbólicas

- O, Ω, θ, o

Notación O

- Se utiliza para buscar cotas superiores del comportamiento de una función de complejidad
- $f(n) \in O(g(n))$ si $\exists n_0, c > 0 / \forall n > n_0,\ f(n) \leq c \cdot g(n)$
- El comportamiento de *f* está acotado por *g*
- ***O(g(n))***: conjunto de todas las funciones acotadas superiormente por *g*
- Ejemplo

$$\begin{cases} f(n) = a_m n^m + \ldots + a_1 n + a_0 \\ f(n) \in O(n^m) \end{cases}$$

Propiedades de la Notación O

- $g(n) = O(g(n))$
- $c \cdot O(g(n)) = O(g(n))$, donde *c* es una constante
- $O(g(n)) + O(g(n)) = O(g(n))$
- $O(g(n)) \cdot O(h(n)) = O(g(n) \cdot h(n))$
- $h(n) \cdot O(g(n)) = O(h(n) \cdot g(n))$
- $O(g(n)) + O(h(n)) = O(max(O(h(n)),O(g(n))))$
- $O(g(n)) - O(g(n)) = O(g(n))$

Notaciones y simbolismos

- **Notación Ω**
  - Se utiliza para acotar inferiormente la función de complejidad en el mejor de los casos
  - $f(n) \in \Omega(g(n))$ si $\exists n_0, c > 0 / \forall n > n_0,\ f(n) \geq c \cdot g(n)$
  - El comportamiento de *f* está acotado inferiormente por *g*
  - ***Ω(g(n))***: cjto. de todas las funciones acotadas inferiormente por *g*
- **Notación θ**

$$f(n) \in \theta(g(n))$$

$$\exists n_0, c_1, c_2 > 0 / \forall n > n_0, c_1 \cdot g(n) < f(n) < c_2 \cdot g(n)$$

  - $\theta(g(n)) = \Omega(g(n)) \bigcap O((g(n))$

- **Notación o**
  - $f(n) \in o(g(n))$ para $n \to \infty$ $\dfrac{f(n)}{g(n)} = 0$
  - Se dice que *f* es asintóticamente equivalente a *g*

Consideraciones

- Operaciones elementales
  - *Las que realiza un computador en tiempo acotado por una constante*
    - *Operaciones aritméticas básicas*
    - *Asignaciones de tipos predefinidos*
    - *Saltos*
    - *Comparaciones lógicas*
    - *Acceso a estructuras indexadas básicas*
- *Es posible realizar el estudio de la complejidad sólo en base a un conjunto reducido de sentencias*
  - *Las que más influyen*

Resultados prácticos

Suma de potencias

- $\sum_{i=1}^{n} i^k = O(n^{k+1})$
- $\sum_{i=1}^{n} c^i = O(c^n)$

Suma de logaritmos

- $\sum_{i=1}^{n} \log i = O(n \cdot \log n)$

Suma de fracciones

- $\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$
- $\sum_{i=1}^{n} \frac{1}{i^c} = O(1)$
- $\sum_{i=1}^{n} \frac{1}{c^i} = O(1)$

Reglas básicas para el cálculo

Procedimiento de análisis

- Desde el interior hacia el exterior

Sentencias simples

- Consumen un tiempo constate

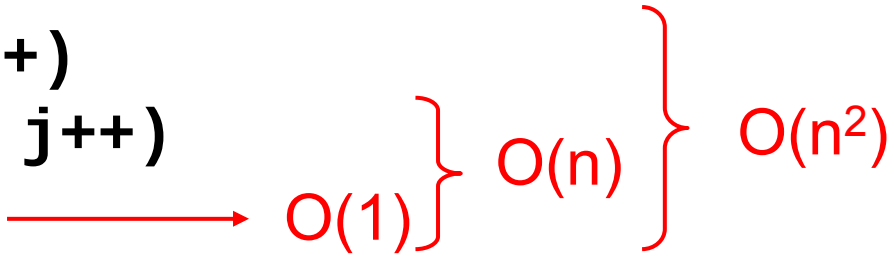
Bucles *for*

- **`for (i=1; i<=n; i++) P(i);` con t el tiempo necesario para calcular `P(i)`**
- Caso sencillo
  - *t* es una constante $\Rightarrow$ O(n·t)
- Caso complejo
  - *t* es función de *i*
  - Tiempo de cálculo de operación en la iteración $k \leq c \cdot k$
  - Tiempo del algoritmo $\leq \sum_{k=1}^{n} c \cdot k = c \sum_{k=1}^{n} k \in O(n^2)$

Reglas básicas para el cálculo

Bucles *for*

```
for (i=1; i<=n; i++)
   for (j=1; i<=n; j++)
        a[i][j]=0;
```




```
   for (i=1; i<=n; i++)
      for (j=i+1; j<=n; j++)
           a[i][j]=0;
```

?

Reglas básicas para el cálculo

Bucles *while*

- A priori no se sabe el número de iteraciones, depende de una condición
- `i=0; while(i<n && [i]!=elemento) i++;`
- Aproximación
  - en el peor de los casos se recorrerá *n* veces por lo que O(n)
- Técnica estándar
  - Buscar una función de las variables implicadas cuyo valor se decremente en cada iteración
- Técnica alternativa
  - Tratarlo como un algoritmo recursivo

If-then-else

- Se analiza (calcula) el peor caso de las dos alternativas posibles
- O(max(f(n),g(n))

Reglas básicas para el cálculo

Bucles *while*

```
i = 1;
j = n;
while (i < j) {
   k = (i+j) / 2;
   if (x < T[k]) j = k-1;
   else if (x == t[k]) {
         i=k;
         j=k;
   } else i=k+1;
}
```

- **Buscando una función**
  - Número de elementos a tratar en las sucesivas iteraciones:
    - n, n/2, n/4, n/8, ... hasta 1
    - En la vuelta *l* se tratan $n/2^l$ elementos
    - En la vuelta *n* (última) $n/2^l=1 \rightarrow n=2^l \rightarrow l = \log_2 n$

Reglas básicas para el cálculo

Bloques

- Se toma el máximo de los órdenes, teniendo en cuenta la regla de la suma

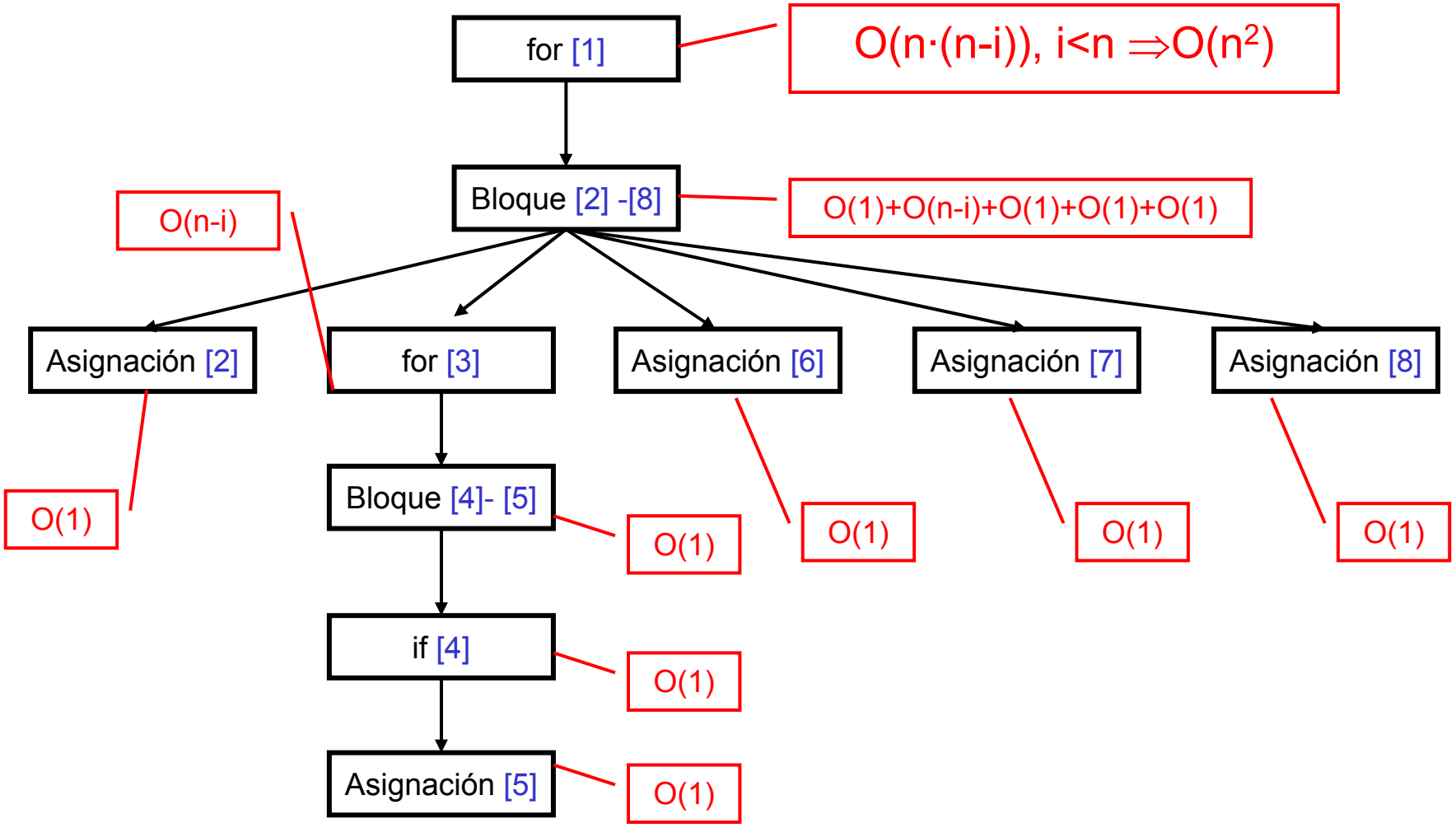
Funciones

- Depende del lugar en el que se realice la llamada
- Llamada a una función en una sentencia simple
  - **`a=f(3)`** con f(i) $\in$O(n), entonces O(n)
- Llamada a una función en una instrucción de tipo *for*
  - Depende de donde se coloque (asignación inicial, condición, incremento)
  - Asignación inicial $\Rightarrow$ se suma el orden de la función al del bucle
  - Condición o incremento $\Rightarrow$ Se suma su orden en el orden del cuerpo del bucle
- Recursivas $\Rightarrow$ Resolución de recurrencias

Ejemplo


```
for (i=0; i<n; i++) {                /*[1]*/
  minimo=i;                          /*[2]*/
  for(j=i+1; j<n; j++)               /*[3]*/
      if(A[j]<A[minimo])             /*[4]*/
          minimo=j;                  /*[5]*/
  temporal=A[minimo];                /*[6]*/
  A[minimo]=A[i];                    /*[7]*/
  A[i]=temporal;                     /*[8]*/
}
```

Ejemplo


for [1]
O(n·(n-i)), i<n ⇒O(n²)
Bloque [2] -[8]
O(1)+O(n-i)+O(1)+O(1)+O(1)
O(n-i)
Asignación [2]
for [3]
Asignación [6]
Asignación [7]
Asignación [8]
O(1)
Bloque [4]- [5]
O(1)
O(1)
O(1)
O(1)
if [4]
O(1)
Asignación [5]
O(1)

Funciones recursivas

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

Valores iniciales

T(0)=1, T(1)=1

$$T(n) = \begin{cases} c & \text{si } n \leq 1 \\ d+T(n-1) & \text{en otro caso} \end{cases}$$

Ecuación recurrente

Resolución: método de ***expansión de la recurrencia***

Reglas básicas para el cálculo

$$T(n) = \begin{cases} c & \text{si } n \leq 1 \\ d+T(n-1) & \text{en otro caso} \end{cases}$$

Valores iniciales

T(0)=1, T(1)=1

Resolución: método de ***expansión de la recurrencia***

$$\mathbf{T(n)}=1+T(n-1)= 1+[1+T(n-2)] = 2+T(n-2)_{\text{ si } n>2} = 3+T(n-3)_{\text{ si } n>3} =$$

$$= \mathbf{i + T(n-i) \text{ si } n>i}$$

Por ejemplo, para i=(n-1):

$$\mathbf{T(n)} = (n-1)+T[n-(n-1)] = (n-1)+T(1) = (n-1)+1= n$$

$$\mathbf{T(n) \in O(n)}$$

Métodos de resolución

- Expansión de la recurrencia
- Inducción matemática
- Ecuaciones características
  - Ecuaciones lineales homogéneas
    - Raíces simples
    - Raíces múltiples
  - Ecuaciones lineales no homogéneas
  - Ecuaciones no lineales

Resolución de recurrencias

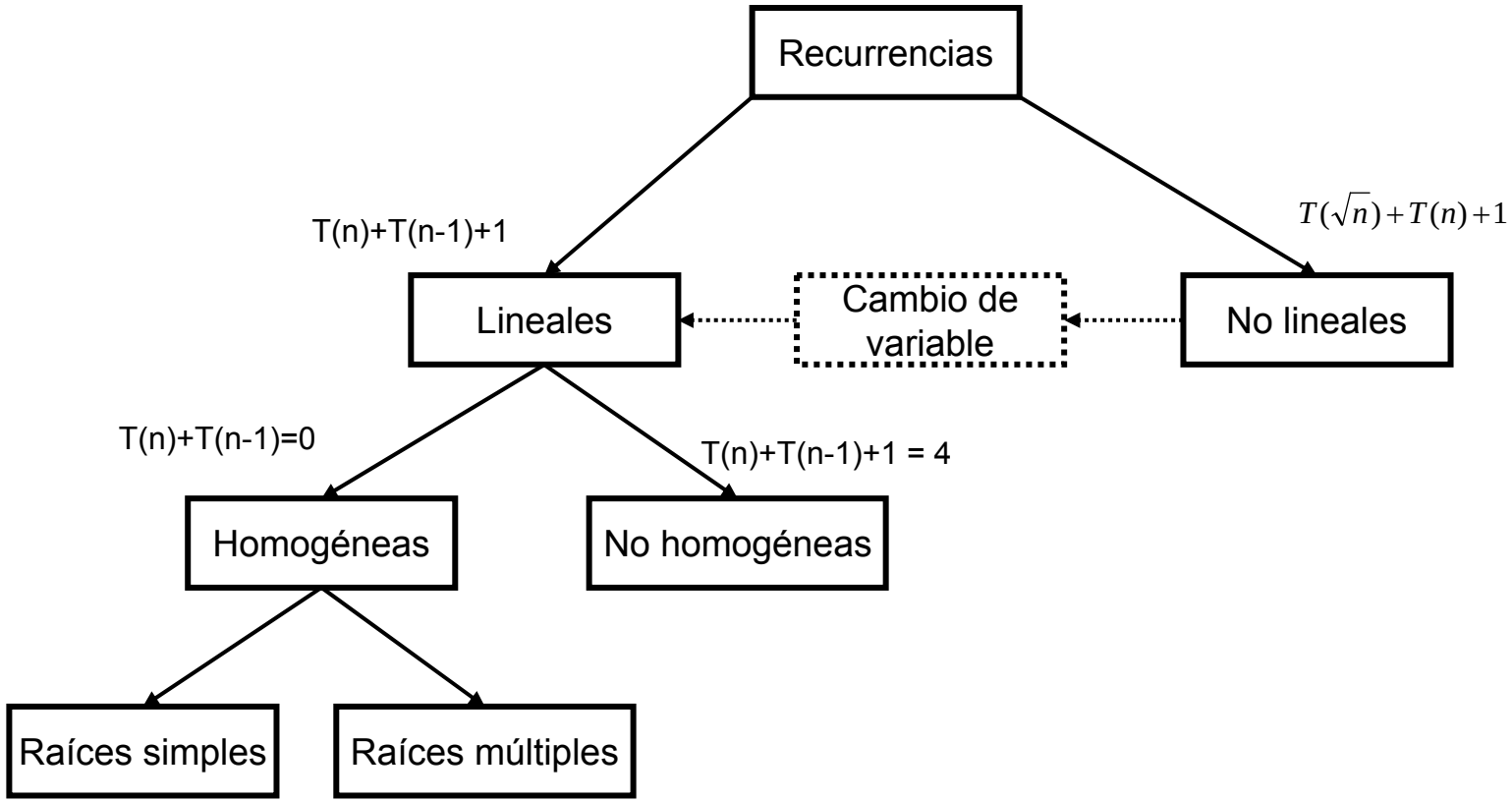
Expansión de la recurrencia

- Obtención de una formula general a partir de varios valores
- Se aplica cuando hay un solo término en *t*, aunque esté multiplicado o sumado por una constante
- Ejemplo: función para cálculo del factorial

Inducción matemática

- Se parte de la propuesta de un determinado orden
- Hay que demostrar que efectivamente la propuesta es correcta

Ecuaciones características



Ecuaciones Lineales Homogéneas

- Consideramos: T(n) = t_n
- Forma de la ecuación
  - $a_0 t_n + a_1 t_{n-1} + ... + a_k t_{n-k} = 0$
- Suposición: $\mathbf{t_n = X^n}$ (polinomio característico), con lo que
  - $a_0 x^n + a_1 x^{n-1} + ... + a_k x^{n-k} = 0$
- Soluciones
  - Trivial $\rightarrow$ 0
  - Sacando factor común $\mathbf{X^{n-k}}$: $a_0 x^k + a_1 x^{k-1} + ... + a_k = 0$
    - Raíces simples
    - Raíces múltiples

Ecuaciones Lineales Homogéneas

- Raíces simples $r_1, r_2, ..., r_k$ $\quad t_n = \sum_{i=1}^{k} c_i r_i^n$
- Ejemplo

$$T(n) \begin{cases} 3T(n-1) + 4T(n-2), n \geq 2 \\ n, n = 0,1 \end{cases}$$

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

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

$$t_n = x^n \qquad x^n - 3x^{n-1} - 4x^{n-2} = 0$$

$$(x^2 - 3x - 4)x^{n-2} = 0 \quad \Rightarrow$$

Ecuación característica

$$\boxed{x^2 - 3x - 4 = 0}$$

Ecuaciones Lineales Homogéneas

- Ejemplo (*continuación*)

$$x^2 - 3x - 4 = 0 \qquad \text{Raíces: } r_1 = -1, r_2 = 4$$

$$t_n = \sum_{i=1}^{k} c_i r_i^n = c_1(-1)^n + c_2 4^n$$

Condiciones iniciales: $\mathbf{t_0=0}$, $\mathbf{t_1=1}$

$$\left.\begin{aligned} 0 &= c_1 + c_2 \\ 1 &= -c_1 + 4c_2 \end{aligned}\right\} \quad c_1 = -\frac{1}{5} \qquad c_2 = \frac{1}{5}$$

$$t_n = \sum_{i=1}^{k} c_i r_i^n = c_1(-1)^n + c_2 4^n = \frac{1}{5}\left(4^n - (-1)^n\right) \in O(4^n)$$

Ecuaciones Lineales Homogéneas

- Raíces múltiples $t_n = c_i r_i^{\,n} + c_{i+1} r_{i+1}^{\,n} + \ldots + n^m r^n$

- Ejemplo
  - r_1=3
  - r_2=2 doble

$$t_n = c_1 3^n + c_2 2^n + c_3 n 2^n$$

- Ejemplo

$$T(n) \begin{cases} 5T(n-1) - 8T(n-2) + 4T(n-3), n \geq 3 \\ n, n = 0,1,2 \end{cases}$$

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

Ecuación característica $x^3 - 5x^2 + 8x - 4 = 0$

1

2, *doble*

Ecuaciones Lineales Homogéneas

- Ejemplo (*continuación*)

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

$$\begin{aligned} t_0 = 0 &\Rightarrow c_1 + c_2 = 0 \\ t_1 = 1 &\Rightarrow c_1 + 2c_2 + 2c_3 = 1 \\ t_2 = 2 &\Rightarrow c_1 + 4c_2 + 8c_3 = 2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} c_1 &= -2 \\ c_2 &= 2 \\ c_3 &= -\tfrac{1}{2} \end{aligned}$$

$$t_n = -2 + 2 \cdot 2^n - \frac{1}{2} n 2^n = 2^{n+1} - n 2^{n-1} - 2 \in O(2^{n+1})$$

Ecuaciones Lineales No Homogéneas

- Combinación lineal que no es igual a 0
  - $a_0 t_n + a_1 t_{n-1} + ... + a_k t_{n-k} = b^n p(n)$
  - *b* es una constante
  - *p(n)* es un polinomio de grado *d*

- Solución: reducir al caso homogéneo
  - Aplicación del polinomio característico (**$t_n = X^n$**)

$$(a_0 x^k + a_1 x^{k-1} + ... + a_k)(x-b)^{d+1} = 0$$

Ecuaciones Lineales No Homogéneas

- Ejemplo

$$T(n)\begin{cases}2T(n-1)+1, n\geq 1\\ 0, n=0\end{cases}$$

$$t_n - 2t_{n-1} = 1 \begin{cases} b=1 \\ p(n)=1 \\ d=0 \end{cases}$$

Ecuación característica

$$(x-2)(x-1)=0 \begin{cases} r_1=2 \\ r_2=1 \end{cases}$$

$$t_n = \sum_{i=1}^{k} c_i r_i^n = c_1 1^n + c_2 2^n$$

$$\begin{array}{l} t_0 = 0 \Rightarrow c_1 + c_2 = 0 \\ t_1 = 1 \Rightarrow c_1 + 2c_2 = 1 \end{array} \Rightarrow \begin{array}{l} c_1 = -1 \\ c_2 = 1 \end{array}$$

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

Ecuaciones Lineales No Homogéneas

- Ejemplo

$$T(n)\begin{cases}2T(n-1)+n, n\geq 1\\ n, n=0\end{cases}$$

$$t_n - 2t_{n-1} = n \begin{cases} b=1 \\ p(n)=n \\ d=1 \end{cases}$$

$$\boxed{(x-2)(x-1)^2=0}\begin{cases} r_1=2 \\ r_2=1, doble \end{cases}$$

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

$$\begin{aligned} t_0 &= 0 \Rightarrow c_1 + c_2 = 0 \\ t_1 &= 1 \Rightarrow 2c_1 + c_2 + c_3 = 1 \\ t_2 &= 4 \Rightarrow 4c_1 + c_2 + 2c_3 = 4 \end{aligned} \Rightarrow \begin{aligned} c_1 &= 2 \\ c_2 &= -2 \\ c_3 &= -1 \end{aligned}$$

$$t_n = 2\cdot 2^n + (-2) - n = 2^{n+1} - n - 2 \in O(2^{n+1})$$

Ecuaciones Lineales No Homogéneas

- Ejemplo

$$T(n)\begin{cases} 2T(n-1)+n+2^n, n\geq 1 \\ 0, n=0 \end{cases}$$

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

$$b_1 = 1,\ p_1(n) = n$$

$$b_2 = 2^n,\ p_2(n) = 1$$

$$(x-2)(x-1)^2(x-2)=0$$

$$\begin{cases} r_1 = 2, doble \\ r_2 = 1, doble \end{cases}$$

$$t_n = c_1 2^n + c_2 n\cdot 2^n + c_3 1^n + c_4 n\cdot 1^n$$

Ecuaciones Lineales No Homogéneas

- Ejemplo (*continuación)*

$$\begin{aligned} t_0 = 0 &\Rightarrow c_1 + c_3 = 0 \\ t_1 = 3 &\Rightarrow 2c_1 + 2c_2 + c_3 + c_4 = 3 \\ t_2 = 12 &\Rightarrow 4c_1 + 8c_2 + 2c_3 + 2c_4 = 12 \\ t_2 = 35 &\Rightarrow 8c_1 + 24c_2 + 2c_3 + 3c_4 = 35 \end{aligned} \quad \Rightarrow \quad \begin{aligned} c_1 &= 2 \\ c_2 &= 1 \\ c_3 &= -2 \\ c_4 &= -1 \end{aligned}$$

$$t_n = 2^{n+1} + n \cdot 2^n - 2 - n = n \cdot 2^n + 2^{n+1} - n - 2 \in (n \cdot 2^n)$$

Ecuaciones No Lineales

- Procedimiento
  - Buscar un cambio de variable que permita obtener de nuevo una función de ‘n’ para T
  - Resolver en la nueva variables
  - Deshacer el cambio

Ecuaciones No Lineales

- Ejemplo

$$T(n) = 4T(n/2) + n \qquad T(1) = 1 \qquad T(2) = 6$$

  - Cambio de variable

$$n = 2^m; \quad m = \log_2 n$$

$$T(2^m) = 4T(2^{m-1}) + 2^m \qquad T(2^m) = t_m$$

$$t_m = 4t_{m-1} + 2^m \Rightarrow (x-4) = 2^m \Rightarrow (x-4)(x-2) = 0$$

$$t_m = c_1 4^m + c_2 2^m$$

  - Las constantes se pueden calcular antes o después de deshacer el cambio, pero siempre hay que tener claro si son valores de *m* o de *n*

Ecuaciones No Lineales

- Ejemplo (continuación)
  - Valores iniciales

$$n\begin{cases} t_1 = 1 \\ t_2 = 6 \end{cases} \qquad m\begin{cases} t_1 = 1 \\ t_2 = 6 \end{cases}$$

  - Deshacemos el cambio

$$4^m = 2^{2^m} = 2^{m^2} = n^2 \quad \Rightarrow \quad t_n = c_1 n^2 + c_2 n$$

$$\left.\begin{cases} c_1 + c_2 = 1 \\ 4c_1 + 2c_2 = 6 \end{cases}\right| \Rightarrow \begin{matrix} c_1 = 2 \\ c_2 = -1 \end{matrix}$$

$$T(n) = 2n^2 - n \in O(n^2)$$

Ecuaciones No Lineales

- Ejemplo

$$T(n) = 2T(\sqrt{b}) + \log_2 n$$

  - Cambio de variable

$$n = 2^{2^m} \Rightarrow T(2^{2^m}) = 2T(2^{2^{m-1}}) + \log_2 2^{2^m}$$

$$T(2^{2^m}) = t_m \Rightarrow t_m = 2t_{m-1} + 2^m$$

$$(x-2)^2 = 0 \Rightarrow r = 2; \quad \text{doble}$$

$$t_m = c_1 2^m + c_2 m \cdot 2^m$$

Ecuaciones No Lineales

- Ejemplo (continuación)

$$t_m = c_1 2^m + c_2 m \cdot 2^m$$

  - Deshacemos el cambio $n = 2^{2^m}$

$$t_n = c_1 \log_2 n + c_2 \log_2(\log_2 n) \cdot \log_2 n$$

$$t_n \in O(\log_2(\log_2 n) \cdot \log_2 n)$$

Ecuaciones No Lineales

- Ejemplo

$$T(n) = T(n/2)T^2(n/4) \qquad T(1) = 2 \qquad T(2) = 4$$

  - Cambio de variable

$$n = 2^{\mathrm{m}} \Rightarrow T(2^m) = T(2^{m-1}) \cdot T^2(2^{m-2})$$

$$T(2^m) = t_m \Rightarrow \mathrm{t_m} = t_{m-1} \cdot t^2{}_{m-2}$$

  - En estos casos conviene realizar un cambio de rango. Para esto tomamos logaritmos

$$\log_2 \mathrm{t_m} = \log_2 t_{m-1} + 2 \cdot \log_2 t_{m-2}$$

  - Ya hemos alcanzado una expresión lineal en m

Ecuaciones No Lineales

- Ejemplo (continuación)

$$\log_2 t_m = \log_2 t_{m-1} + 2\log_2 t_{m-2}$$

  - Consideramos $V_m = \log_2 \mathrm{t_m}$

$$V_m = V_{m-1} + 2 \cdot V_{m-2} \Rightarrow (x^2 - x - 2) = 0$$

$$(x-2)(x+1) = 0 \Rightarrow \mathrm{V_m} = c_1 2^m + c_2 (-1)^m$$

  - Deshacemos el cambio teniendo en cuenta que $a^{\log b} = b^{\log a}$

$$\log_2 \mathrm{t_m} = c_1 2^m + c_2 (-1)^m \Rightarrow t_m = 2^{c_1 2^m + c_2 (-1)^m}$$

  - Deshacemos el cambio $n = 2^{\mathrm{m}}$

$$\mathrm{t_n} = 2^{c_1 n + c_2 (-1)^{\log_2 n}} \Rightarrow T(n) = 2^{c_1 n + c_2 (-1)^{\log_2 n}}$$

Ecuaciones No Lineales

- Ejemplo (continuación)

$$T(n) = 2^{c_1 n + c_2 (-1)^{\log_2 n}}$$

$$\begin{matrix} T(1) = 2 \\ T(2) = 4 \end{matrix} \Rightarrow \begin{cases} 2 = 2^{c_1 + c_2} \\ 4 = 2^{2c_1 + c_2} \end{cases} \Rightarrow \begin{cases} \log_2 = (c_1 + c_2)\log_2 2 \\ \log_2 4 = (2c_1 + c_2)\log_2 2 \end{cases}$$

$$\Rightarrow \left\{ \begin{matrix} 1 = c_1 + c_2 \\ 2 = 2c_1 + c_2 \end{matrix} \right| \Rightarrow \begin{matrix} c_1 = 1 \\ c_2 = 0 \end{matrix} \Rightarrow \mathrm{T(n)} = 2^{\mathrm{n}} \in O(2^n)$$