Instituto Politécnico Nacional

Escuela Superior de Cómputo

Análisis de algoritmos

Tema 06: Análisis de algoritmos recursivos

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom edgardoadrianfrancom

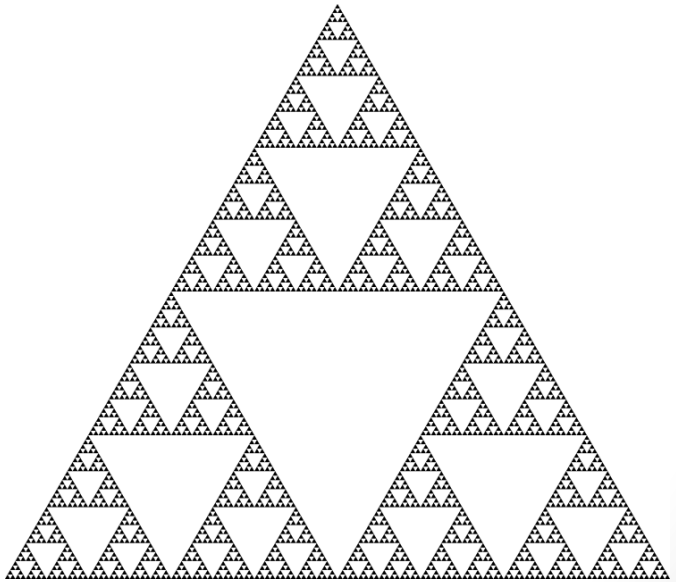
Contenido

- Recursividad
- Ecuaciones en recurrencia
  - Ejemplo 01: Factorial recursivo
  - Ejemplo 02: Fibonacci recursivo
  - Ejemplo 03: Torres de Hanói recursivo
  - Ejemplo 04: Búsqueda binaria recursiva
- Recurrencias homogéneas
  - Ejemplo
- Recurrencias no homogéneas
  - Ejemplo
- Recurrencias no lineales
  - Teorema maestro
    - Ejemplo 01
    - Ejemplo 02

Recursividad

- La **recursividad** es un concepto fundamental en matemáticas y en computación. Es una **alternativa** diferente **para implementar** estructuras de repetición ***(iteración).***

- Se puede usar en toda situación en la cual la solución pueda ser expresada como una secuencia de movimientos, pasos o transformaciones gobernadas por un conjunto de **reglas no ambiguas.**



- La **recursividad** es un recurso muy poderoso que **permite expresar soluciones simples y naturales a ciertos tipos de problemas**. Es importante considerar que no todos los problemas son naturalmente recursivos.

- Un **objeto recursivo** es aquel que **aparece en la definición de si mismo**, así como el que ***se llama a sí mismo*.**

- La recursividad es un fenómeno que se presenta en muchos problemas de forma natural, delegando la **solución de un problema en la solución de otro más pequeño**.

- El **análisis temporal de un algoritmo recursivo** vendrá en función del **tiempo requerido por la(s) llamada(s) recursiva(s)** que aparezcan en él.

- **El análisis temporal de un algoritmo iterativo es simple con base en la operación básica de este**, para los **algoritmos recursivos** nos vamos a encontrar con una dificultad añadida, pues **la función que establece su tiempo de ejecución** viene dada por **una ecuación en recurrencia**, es decir, $\boldsymbol{T(n) = E(n)}$, en donde en la expresión E aparece la propia función $\boldsymbol{T}$.

Ecuaciones en recurrencia

- Cuando se quiere calcular la demanda de recursos de un algoritmo definido recursivamente, la función complejidad que resulta no está definida sólo en términos del tamaño del problema y algunas constantes, sino en términos de la función complejidad misma.

- Además no es una sola ecuación, dado que existen otras (al menos una) que determinan la cantidad de recursos para los **casos base** de los algoritmos recursivos. Dada esta situación, **para poder obtener el comportamiento del algoritmo, es necesario resolver el sistema recurrente obtenido**.

Ejemplo 01: Factorial recursivo

- Considerando el **producto de `num*Factorial(num-1)` como operación básica**, y el **costo del caso base como 1 ($\mathbf{T(0) = 1}$)**, podemos construir la ecuación recurrente para calcular la complejidad del algoritmo como sigue:

```
int Factorial( int num )
{
      if ( num == 0 )
          return 1;
      else
         return num * Factorial( num - 1 );
}
```

Costo de la multiplicación

Costo de la llamada a Factorial (n-1)

$$T(n) = 1 + T(n-1)$$
$$T(0) = 1$$

Costo del caso base

Ejemplo 02: Fibonacci recursivo

- Considerando la **suma** `Fibonacci(num-1) + Fibonacci (num-2)` **como operación básica**, y el **costo 1** de los **2 casos base** podemos construir la ecuación recurrente para calcular la complejidad del algoritmo.

```
int Fibonacci(int num)
{
      if (num ==0)
          return 0;
      else if (num == 1)
          return 1;
      else
         return Fibonacci(num-1) + Fibonacci (num-2);
}
```

Costo de la llamada a Fibonacci(n-1)

Costo de la suma

Costo de la llamada a Fibonacci(n-2)

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

$$T(0) = 1$$
$$T(1) = 1$$

Costo de n=0 y n=1

Ejemplo 03: Torres de Hanói recursivo

- Considerando la **operación** `Mover_de(Src,Dst)` **como operación básica**, y tomado un **coste de 0 cuando N=0**, podemos construir la ecuación recurrente para calcular la complejidad del algoritmo.

```
Hanoi(N, Src, Aux, Dst)
{
        if(n>0)
        {
                Hanoi(N - 1, Src, Dst, Aux);
                Mover_de(Src,Dst);
                Hanoi(N - 1, Aux, Src, Dst);
        }
        return;
}
```

$$T(n) = T(n-1) + 1 + T(n-1)$$
$$T(0) = 0$$

Ejemplo 04: Búsqueda binaria recursiva

- Considerando la **operación como operación básica las comparaciones**, y tomado un **coste de 0 cuando Tam(numeros[])=0**, podemos construir la ecuación recurrente para calcular la complejidad del algoritmo.

```
int BusquedaBinaria(int num_buscado, int numeros[], int inicio, int centro, int final)
{
		if (inicio>final)
				return -1;
		else if (num_buscado == numeros[centro])
				return centro;
		else if (num_buscado < numeros[centro])
				return BusquedaBinaria(num_buscado,numeros,inicio,(int)((inicio+centro-1)/2),centro-1);
		else
				return BusquedaBinaria(num_buscado,numeros,centro+1,(int)((final+centro+1)/2),final);
}
```

$$T(n) = 3 + T(n/2)$$
$$T(0) = 0$$

Ejemplo 05: Merge-sort

- Sea A un arreglo de n elementos y p, r índices del rango a ordenar.

```
Merge-Sort(a, p, r)
{
      if ( p < r )
      {
            q = parteEntera((p+r)/2);
            Merge-Sort(a, p, q);
            Merge-Sort(a, q+1,r);
            Merge(a, p, q, r);
      }
}
```

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

Recurrencias homogéneas

- Son de la forma:

$$a_0T(n) + a_1T(n-1) + a_2T(n-2) + \ldots + a_kT(n-k) = 0$$

  - Donde los coeficientes a_i son números reales, y k es un número natural entre 1 y n.

- Para eliminar la recurrencia se buscan términos que sean combinaciones de funciones exponenciales de la forma:

$$T(n) = c_1p_1(n)r_1^n + c_2p_2(n)r_2^n + \ldots + c_kp_k(n)r_k^n = \sum_{i=1}^{k} c_ip_{i1}(n)r_i^n$$

  - Donde los valores $c_1, c_2, \ldots, c_n$ y $r_1, r_2, \ldots, r_n$ son números reales, y $p_1(n), \ldots, p_k(n)$ son polinomios en n con coeficientes reales.

$$a_0T(n) + a_1T(n-1) + a_2T(n-2) + \ldots + a_kT(n-k) = 0$$

- Para resolverlas haremos el cambio $x^k = T(n)$, con lo cual obtenemos la ***ecuación característica*** asociada:

$$a_0x^k + a_1x^{k-1} + a_2x^{k-2} + \ldots + a_k = 0$$

- Llamemos $r_1, r_2, \ldots, r_k$ a sus raíces, ya sean reales o complejas. Dependiendo del orden de multiplicidad de tales raíces, pueden darse los siguientes casos:
  - *Caso 1: Raíces distintas*
  - *Caso 2: Raíces con multiplicidad mayor que* 1

Caso 1: Raíces distintas

- Si todas las raíces de la ecuación característica son distintas, esto es, $r_i \neq r_j$ si $i \neq j$, entonces la solución de la ecuación en recurrencia viene dada por la expresión:

$$T(n) = c_1 r_1^n + c_2 r_2^n + \ldots + c_k r_k^n = \sum_{i=1}^{k} c_i r_i^n$$

  - Donde los coeficientes c_i se determinan a partir de las condiciones iniciales.

Caso 2: Raíces con multiplicidad mayor que 1

Supongamos que alguna de las raíces (p.e. r_1) tiene multiplicidad $m > 1$. Entonces la ecuación característica puede ser escrita en la forma:

$$(x - r_1)^m (x - r_2) \dots (x - r_{k-m+1})$$

- En cuyo caso la solución de la ecuación en recurrencia viene dada por la expresión:

$$T(n) = \sum_{i=1}^{m} c_i n^{i-1} r_i^n + \sum_{i=m+1}^{k} c_i r_{i-m+1}^n$$

  - Donde los coeficientes c_i se determinan a partir de las condiciones iniciales.

- Este caso puede ser generalizado, si $r_1, r_2, \ldots, r_k$ son las raíces de la ecuación característica de una ecuación en recurrencia homogénea, cada una de multiplicidad m_i, esto es, si la ecuación característica puede expresarse como:

$$(x - r_1)^{m_1}(x - r_2)^{m_2} \ldots (x - r_k)^{m_k} = 0$$

- Entonces la solución a la ecuación en recurrencia viene dada por la expresión:

$$T(n) = \sum_{i=1}^{m_1} c_{1i} n^{i-1} r_1^n + \sum_{i=1}^{m_1} c_{2i} n^{i-1} r_2^n + \ldots + \sum_{i=1}^{m_k} c_{ki} n^{i-1} r_k^n$$

  - Donde los coeficientes c_i se determinan a partir de las k condiciones iniciales.

Recurrencias homogéneas (Ejemplo)

- Se tiene la siguiente ecuación de recurrencia

$$T(n) = 5T(n-1) - 8T(n-2) + 4T(n-3), n \geq 2$$
$$T(k) = k \text{ para } k = 0, 1, 2$$

- Reordenando términos

$$T(n) - 5T(n-1) + 8T(n-2) - 4T(n-3) = 0$$

- Haciendo el cambio $x^3 = T(n)$ obtenemos su ecuación característica $x^3 - 5x^2 + 8x - 4 = 0$, lo que es igual a $(x-2)^2(x-1) = 0$, por lo tanto: $r_1 = 2, r_2 = 2$ y $r_3 = 1$

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

- De las condiciones iniciales obtenemos:

$$c_1 = 2, c_2 = -\frac{1}{2} \; y \; c_3 = -2$$

$$\boldsymbol{T(n) = 2^{n+1} - n2^{n-1} - 2 \in O(n2^n)}$$

Recurrencias no homogéneas

- Son de la forma:

$$a_0T(n) + a_1T(n-1) + a_2T(n-2) + \ldots + a_kT(n-k) = b^n p(n)$$

  - Donde los coeficientes a_i y b son números reales, y $p(n)$ es un polinomio en n de grado d.

- Para resolver este tipo de ecuaciones generalmente se deben manipular para llegar a una ecuación homogénea. Una formula general para resolverla es mediante la ecuación característica:

$$(a_0x^k + a_1x^{k-1} + a_2x^{k-2} + \ldots + a_k)(x-b)^{d+1} = 0$$

- Lo que nos lleva a aplicar el método para las recurrencias homogéneas.

- Generalizando este proceso, si tenemos una ecuación de la forma:

$$a_0T(n) + a_1T(n-1) + a_2T(n-2) + \ldots + a_0T(n-k) = b_1^n p_1(n) + b_2^n p_2(n) + \cdots + b_s^n p_s(n)$$

  - Donde los coeficientes a_i y b_j son números reales, y $p_j(n)$ son polinomios en n de grado d_j

- La ecuación característica es:

$$(a_0x^k + x^{k-1} + a_2x^{k-2} + \ldots + a_k)(x - b_1)^{d_1+1}(x - b_2)^{d_2+1} \ldots (x - b_s)^{d_s+1} = 0$$

Recurrencias no homogéneas (Ejemplo)

- Se tiene la siguiente ecuación de recurrencia

$$T(n) = 5T(n-1) - 8T(n-2) + 4T(n-3) + 2^n 3n, n \geq 2$$

$$T(k) = k \text{ para } k = 0, 1, 2$$

- Reordenando términos

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

- Haciendo el cambio $x^3 = T(n)$, $b = 2$ y $d = 1$ obtenemos su ecuación característica $(x^3 - 5x^2 + 8x - 4)(x-2)^2 = 0$, lo que es igual a $(x-2)^2(x-1)(x-2)^2 = 0$, por lo tanto: $r_1 = 2$, $r_2 = 2$, $r_3 = 1$, $r_4 = 2$ y $r_5 = 2$

- Si $c_4 \neq 0$

$$T(n) = c_1 2^n + c_2 n 2^n + c_3 n^2 2^n + c_4 n^3 2^n + c_5 1^n \in \boldsymbol{O(n^3 2^n)}$$

Ejemplo 01: Factorial recursivo

```
int Factorial( int num )
{
     if ( num == 0 )
         return 1;
     else
        return num * Factorial( num - 1 );
}
```

$$n! = \begin{cases} 1 & \text{si, } n = 0 \\ (n-1)! \times n & \text{si, } n > 0 \end{cases}$$

$$T(n) = 1 + T(n-1); n > 0$$
$$T(0) = 1$$

- Reordenando términos

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

- Haciendo el cambio $x^{k=1} = T(n)$, $b = 1$ y $d = 0$ obtenemos su ecuación característica $(x-1)(x-1) = 0$, lo que es igual a $(x-1)^2 = 0$, por lo tanto: $r_{1=}1$y $r_{2=}1$.

$$T(n) = c_1 + c_2 n$$

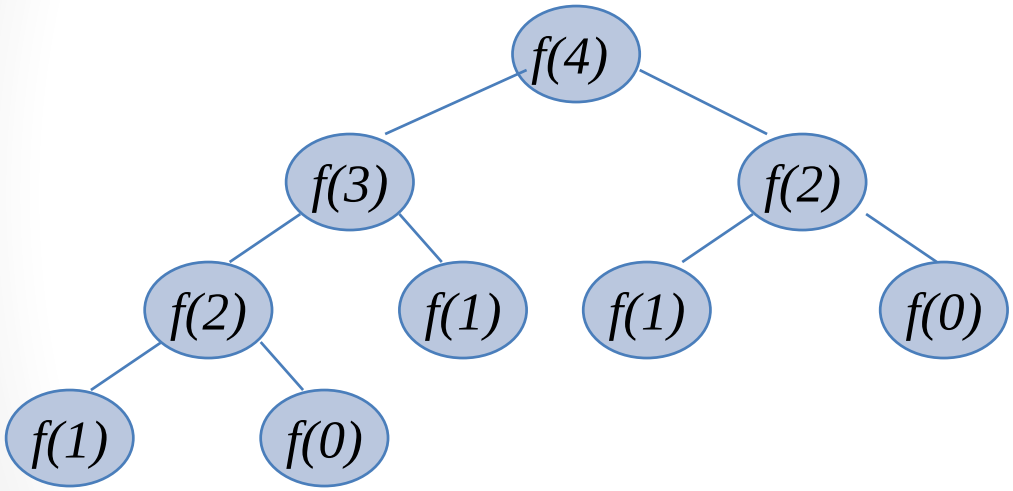
- Si $T(0) = 1$ y $T(1) = 2$ $c_1 = 1, c_2 = 1$

$$\boldsymbol{T(n) = 1 + n \in O(n)}$$

Ejemplo 02: Fibonacci recursivo

```
Function fibonacci (n:int): int;
if (n=0) return 0;
else if (n=1) return 1;
else return fibonacci(n-1) + fibonacci(n-2);
```

$$F(n) = \begin{cases} 0 & \text{si } n = 0; \\ 1 & \text{si } n = 1; \\ F(n-1) + F(n-2) & \text{si } n > 1. \end{cases}$$



¿Cuántas operaciones básicas se requieren para calcular:

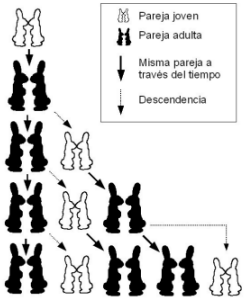
f(4)? **--> 9**

f(3)? **--> 5**

f(2)? **--> 3**

f(1)? **--> 1**

f(0)? **--> 1**

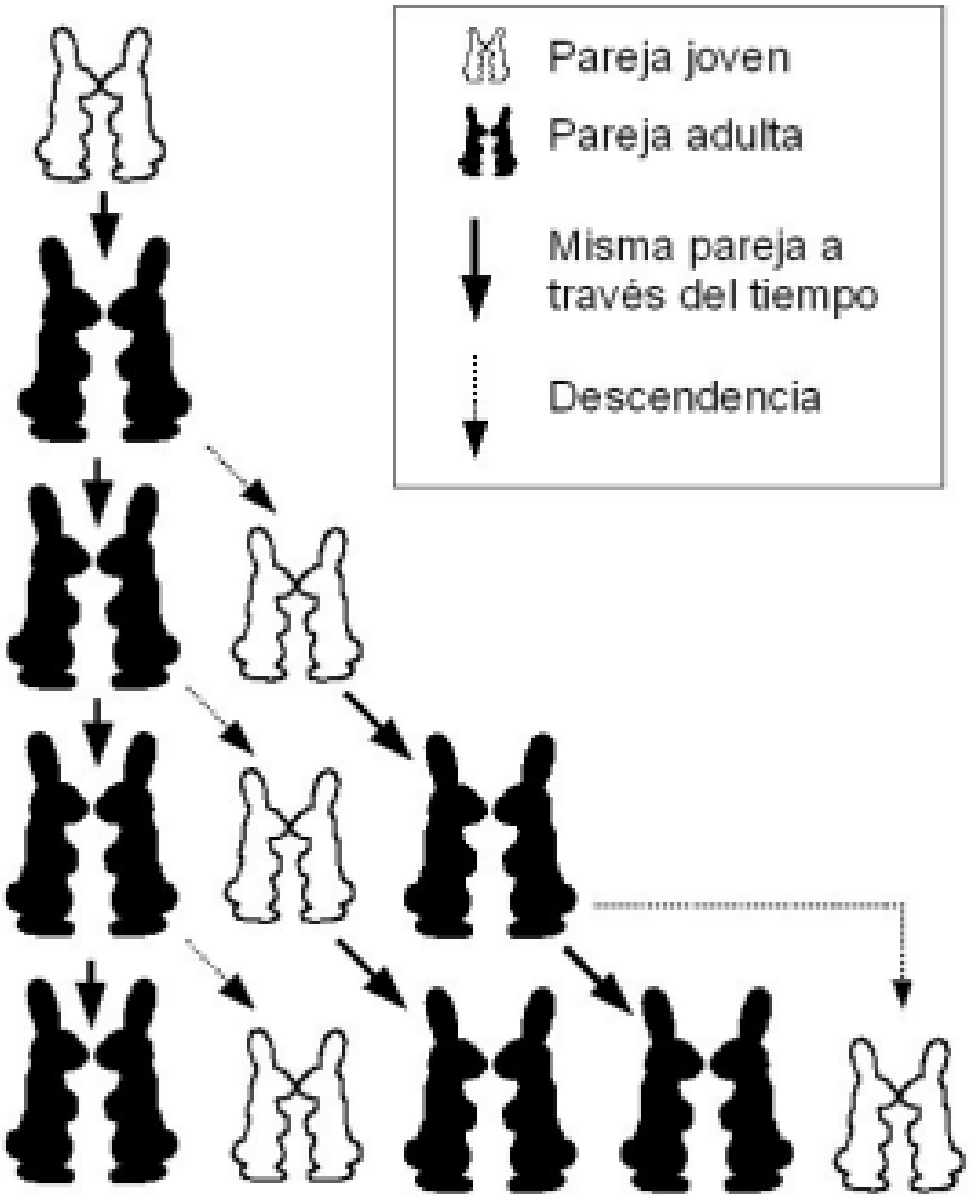


Relación:

*El término **T(n)** requiere **T(n-1)+1+T(n-2)** términos para calcularse.*

T(0)=1

T(1)=1


Pareja joven
Pareja adulta
Misma pareja a través del tiempo
Descendencia

- La ecuación en recurrencia definida para la sucesión de Fibonacci con las condiciones iniciales:

$$T(0) = 1 \text{ y } T(1) = 1.$$

$$T(n) = T(n-1) + T(n-2) + 1, n \geq 2$$

- Reordenando términos

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

- Haciendo el cambio $x^{k=1} = T(n)$, $b = 1$ y $d = 0$ obtenemos su ecuación característica
  - $(x^2 - x - 1)(x - 1) = 0$, cuyas raíces son:

$$r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2}, r_3 = 1$$

  - Y por lo tanto $T(n) = c_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^n + c_3$

- Para calcular las constantes c_1, c_2 y c_3 necesitamos utilizar las condiciones iniciales de la ecuación original, obteniendo:

$$T(0) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^0 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^0 + c_3 = c_1 + c_2 + c_3 = 1$$

$$T(1) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^1 + c_3 = 1$$

$$T(2) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^2 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^2 + c_3 = 3$$

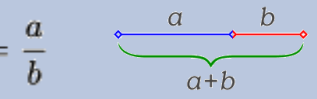
$$T(n) = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n + c_3$$

- Si c_1 o c_2 no valen 0 se tendrá $\in O(c^n)$

El número áureo surge de la división en dos de un segmento guardando las siguientes proporciones: La longitud total ***a+b*** es al segmento más largo ***a***, como ***a*** es al segmento más corto ***b***.

$\varphi = \left(\frac{1+\sqrt{5}}{2}\right) \approx 1.6180339887498948482045868343656\ldots$ **(número áureo)**

$$\frac{a+b}{a} = \frac{a}{b}$$



Ejemplo 03: Torres de Hanói

```
Solve(N, Src, Aux, Dst)
  if N is 0
    exit
  else
    Solve(N - 1, Src, Dst, Aux)   T(n-1)
    Move from Src to Dst          1
    Solve(N - 1, Aux, Src, Dst)   T(n-1)
```

Operación básica
movimiento de cero o un disco

$$T(0) = 0$$
$$T(1) = 1$$

- $\boldsymbol{T(n) = 2T(n-1) + 1 \text{ si } n > 0}$
- Acomodando los términos se tiene:

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

$$(a_0 x^k + x^{k-1} + a_2 x^{k-2} + \ldots + a_k)(x-b)^{d+1} = 0$$

  - No homogénea con $b = 1, p(n) = 0\ y\ d = 0;$

- $(a_0x^k + x^{k-1} + a_2x^{k-2} + \ldots + a_k)(x-b)^{d+1} = 0$
- $T(n) - 2T(n-1) = 1; b = 1, p(n) = 0\ y\ d = 0$
- Su Ecuación característica es:

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

$x1 = 2$
$x2 = 1$
Sustituir como raíces distintas

- Por lo tanto

$$T(n) = c_1 2^n + c_2 1^n \in \Theta(2^n)$$

Con los casos iniciales
$T(0) = 0$
$T(1) = 1$
Encontramos
c_1=1
c_2=-1
$T(n) = 2^n - 1 \in \Theta(2^n)$

Recurrencias no lineales

- El **teorema maestro** es un método matemático que se usa para resolver ciertos casos particulares de ecuaciones de recurrencia como la siguiente:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Donde el coeficientes $a \geq 1$ y $b > 1$ y $\left(\frac{n}{b}\right)$ puede ser tomada como $\left\lfloor \frac{n}{b} \right\rfloor$ o $\left\lceil \frac{n}{b} \right\rceil$ entonces $T(n)$ tiene los siguientes limites asintóticos :

1. Si $f(n) = O(n^{\log_b a - \varepsilon})$ para alguna $\varepsilon > 0$, entonces
   - $\boldsymbol{T(n) = \Theta\left(n^{\log_b a}\right)}$
2. Si $f(n) = \Theta(n^{\log_b a})$, entonces
   - $\boldsymbol{T(n) = \Theta\left(n^{\log_b a} \lg(n)\right)}$
3. Si $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ para alguna $\varepsilon > 0$ y si $af(n/b) \leq cf(n)$, para alguna constante $c < 1$ y suficientemente grandes n entonces
   - $\boldsymbol{T(n) = \Theta(f(n))}$

Logaritmos

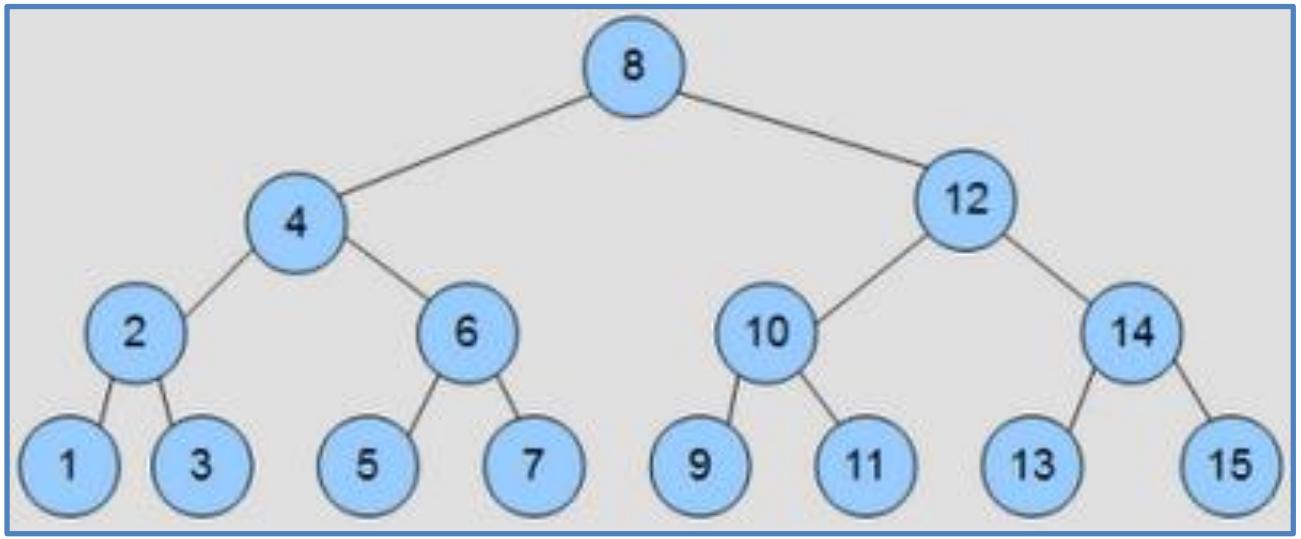
- Un logaritmo es un exponente o potencia, a la que un número fijo (llamado base), se ha de elevar para dar un cierto número.

- Entonces, el logaritmo es la función inversa de la función exponente.

$$\log_a c = b$$

$$a^b = c$$

- $\log_2(n)$ = el exponente a la que 2 se ha de elevar para obtener n.
  - P.g: $\log_2(8) = 3$ (porque $2^3 = 8$)
  - P.g: $\log_2(1024) = 10$ (porque $2^{10} = 1024$)



Propiedades de los logaritmos

1. El logaritmo de la base siempre es igual a uno, es decir:

$$\log_a a = 1$$

2. El logaritmo de 1 en cualquier base es siempre igual a cero:

$$\log_a 1 = 0$$

3. El logaritmo de un producto es igual a la suma de los logaritmos de sus factores:

$$\log_a (b \cdot c) = \log_a b + \log_a c$$

4. El logaritmo de una fracción es igual a la resta del logaritmo del numerador menos el logaritmo del denominador.

$$\log_a (b/c) = \log_a b - \log_a c$$

5. El logaritmo de una potencia es igual a la potencia multiplicando al logaritmo de la base de la potencia:

$$\log_a b^c = c \log_a b$$

6. El logaritmo de la base elevado a una potencia es igual a la potencia.

$$\log_a a^b = b$$

7. **Cambio de base de logaritmo:** El logaritmo en base a un número es igual a la fracción entre el logaritmo del primer número con base en un tercer número y el logaritmo del segundo número con base en un tercer número.

$$\log_a b = \log_c b / \log_c a$$

8. Un número elevado al logaritmo con base en el mismo número, es igual al número del logaritmo.

$$a^{\log_a b} = b$$

Ejemplo 01 Uso del teorema maestro

1. Si f(n) $=$ $\Omega(n^{\log_b a+\varepsilon})$ para algún $\varepsilon > 0$, y
   si $a\, f(n/b) \leq c\, f(n)$ para
   alguna constante c $<$ 1y suficientemente grandes n,
   $\Rightarrow$ T(n) $=$ Θ(f(n)).

$T(n) = 9T(n/3) + n$

$T(n) = 9T(n/3) + n$

$a = 9, \quad b = 3, \quad f(n) = n \quad \Rightarrow \quad n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$

$\therefore f(n) = O(n^{\log_3 9-\varepsilon}), \varepsilon = 6$

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

Ejemplo 02 Uso del teorema maestro

1. Si $f(n) = O(n^{\log_b a-\varepsilon})$ para algún $\varepsilon > 0$ $\Rightarrow T(n) = \Theta(n^{\log_b a})$
2. Si $f(n) = \Theta(n^{\log_b a})$ $\Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$
3. Si $f(n) = \Omega(n^{\log_b a+\varepsilon})$ para algún $\varepsilon > 0$, y
   si $a\, f(n/b) \leq c\, f(n)$ para
   alguna constante $c < 1$ y suficientemente grandes n,
   $\Rightarrow T(n) = \Theta(f(n))$.

$$T(n) = T(2n/3) + 1$$

$$T(n) = T(2n/3) + 1 \Rightarrow \quad Caso2$$
$$a = 1, \quad b = 3/2, \quad f(n) = 1 \quad \Rightarrow \quad n^{\log_b a} = n^{\log_{3/2} 1} = n^o = 1$$

$$\therefore T(n) = \Theta(\lg n)$$

Ejemplo 03 Uso del teorema maestro

1. Si $f(n) = O(n^{\log_b a - \varepsilon})$ para algún $\varepsilon > 0$ $\Rightarrow T(n) = \Theta(n^{\log_b a})$
2. Si $f(n) = \Theta(n^{\log_b a})$ $\Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$
3. Si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ para algún $\varepsilon > 0$, y
   si $a\, f(n/b) \leq c\, f(n)$ para
   alguna constante $c < 1$ y suficientemente grandes n,
   $\Rightarrow T(n) = \Theta(f(n))$.

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

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

$a = 2, \quad b = 2, \quad f(n) = n \quad \Rightarrow \quad n^{\log_b a} = n^{\log_2 2} = n^1 = n$

$f(n) = \Theta(n^{\log_b a}) \quad \Rightarrow \quad Caso2$

$$T(n) = \Theta(n\lg(n))$$

Ejemplo 04 Uso del teorema maestro

1. Si $f(n) = O(n^{\log_b a-\varepsilon})$ para algún $\varepsilon > 0 \Rightarrow T(n) = \Theta(n^{\log_b a})$
2. Si $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$
3. Si $f(n) = \Omega(n^{\log_b a+\varepsilon})$ para algún $\varepsilon > 0$, y
   si $a\, f(n/b) \leq c\, f(n)$ para
   alguna constante $c < 1$ y suficientemente grandes n,
   $\Rightarrow T(n) = \Theta(f(n))$.

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

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$

$$a = 2, \quad b = 2, \quad f(n) = n^2 = \Theta(n^2). \quad \Rightarrow \quad f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right)$$

$$n^{\log_b a+2} = n^{\log_2 4} = n^2$$

$$\Rightarrow \quad Caso3 \Rightarrow si\ f(n/b) \leq c\, f(n) \Rightarrow (n/2)^2 \leq c\, n^2\ para\ alguna\ c < 1 \Rightarrow c=1/2$$

$$T(n) = \Theta(f(n)).$$

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

Otra forma de ver el teorema maestro

Teorema 1.1 *Sean* $a, b, c, d \in \mathbb{R}$, $a \geq 1$, $b > 1$, $d > 0$, *y*

$$T(n) = \begin{cases} d & n = 1 \\ aT(n/b) + n^c & n > 1 \end{cases}$$

Donde n/b *puede interpretarse como* $\lfloor n/b \rfloor$ *o* $\lceil n/b \rceil$. *Entonces,*

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