ANÁLISIS Y DISEÑO DE ALGORITMOS I
Periodo I – 2023

Jesús Aranda
jesus.aranda@correounivalle.edu.co

Universidad del Valle
Escuela de Ingeniería de Sistemas y Computación

Este documento es una adaptación del material original del profesor Oscar Bedoya

Algoritmos en la computación

Técnica de diseño Dividir y conquistar

- Esta técnica considera la descomposición del problema original en más pequeños para su solución.

Se identifican tres etapas (a veces sólo hay dos):

- Dividir el problema en subproblemas
- Conquistar los subproblemas (solucionarlos recursivamente)
- Combinar las soluciones de los subproblemas para crear la solución al problema original

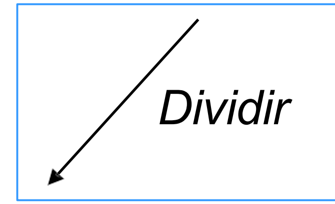
Algoritmos en la computación

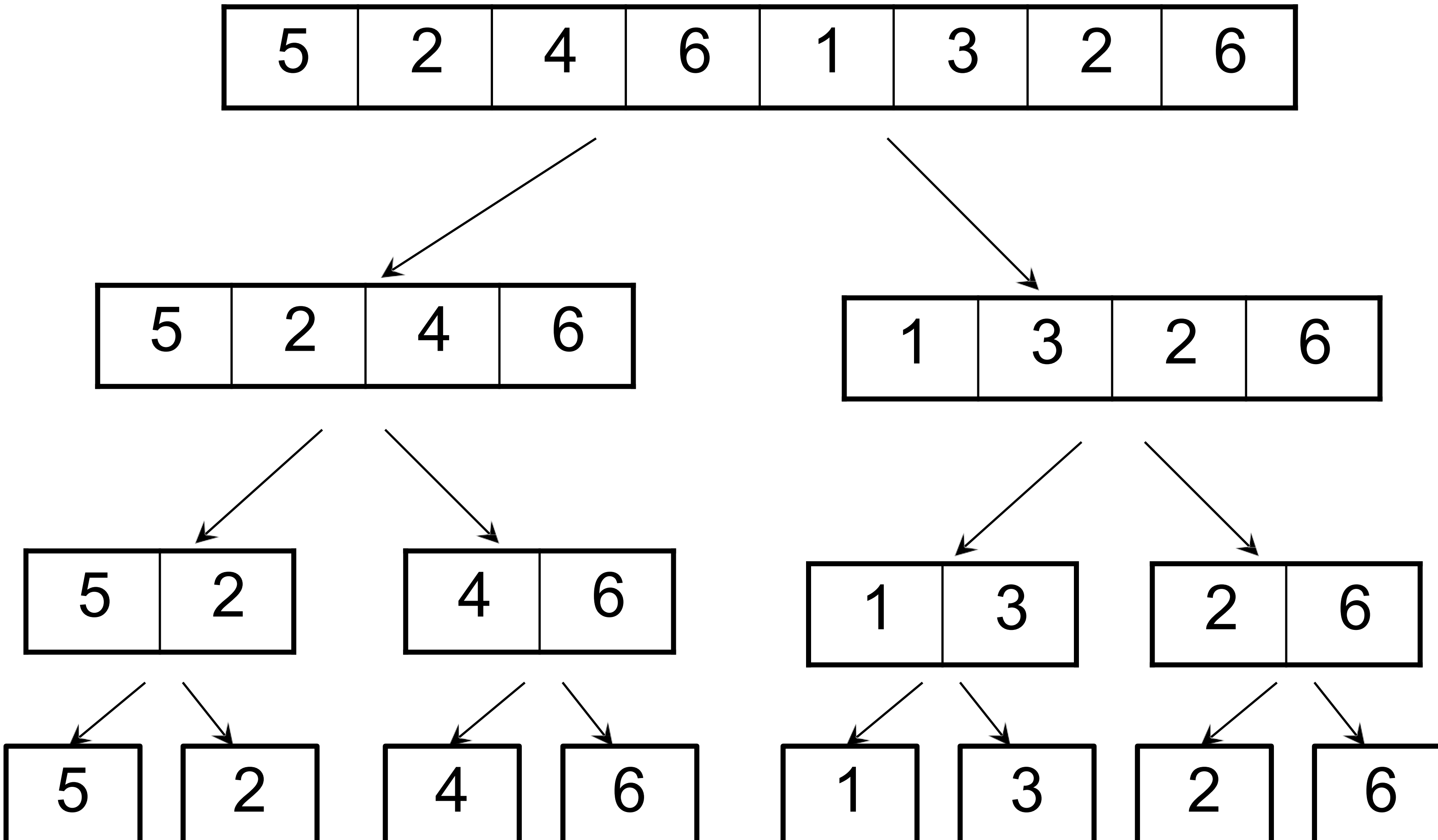
Técnica de diseño Dividir y conquistar

- Una gran variedad de algoritmos descomponen el problema y solucionan sus partes para poder luego encontrar la solución al problema original, por ejemplo:
  - Ordenamiento (Merge Sort, Quick Sort)
  - Multiplicación de matrices (Algoritmo de Strassen)
  - Filtrado colaborativo (Algoritmo para medir similitud de listas de preferencias)

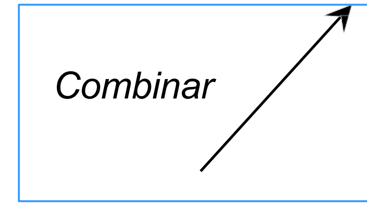
Algoritmos en la computación

Merge Sort

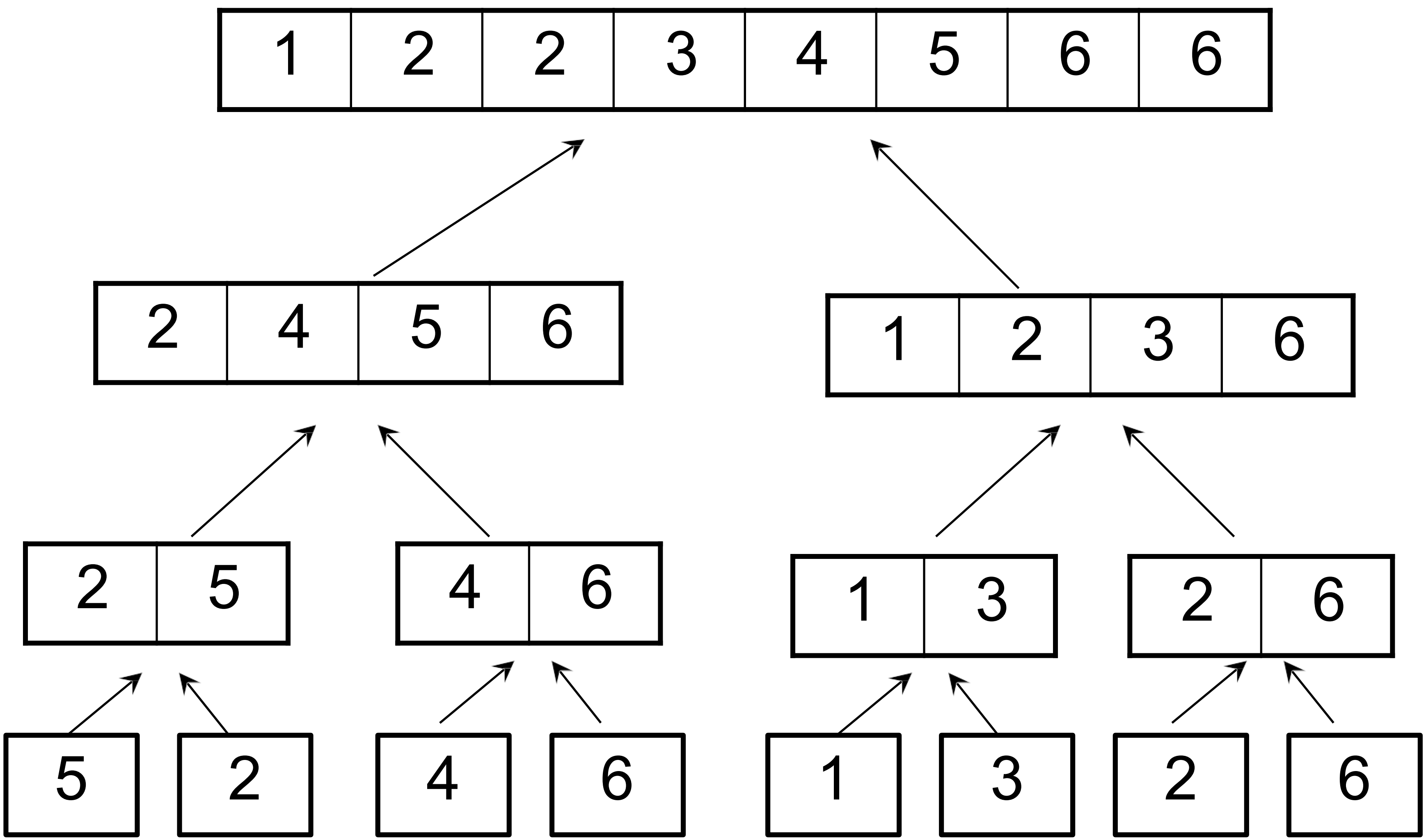




Algoritmos en la computación



Merge Sort



Algoritmos en la computación

Merge sort

Entrada: $\langle a_1, a_2, \ldots, a_n \rangle$

- Dividir: $\langle a_1, \ldots, a_q \rangle$ $\langle a_{q+1}, \ldots, a_n \rangle$
- Conquistar: $a'_1 \leq a'_2 \leq \ldots \leq a'_q$ y $a'_{q+1} \leq a'_{q+2} \leq \ldots \leq a'_n$
- Combinar: si $a'_1 \leq a'_{q+1}$ y $a'_2 > a'_{q+1}$ entonces $\langle a'_1, a'_{q+1}, \ldots \rangle$ sino ...

Algoritmos en la computación

La primera invocación al MERGE-SORT sería MERGE-SORT(A, 1, length(A))

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

Tomado de [CLRS09]

Algoritmos en la computación

Merge sort

$$T(n)=\begin{cases} 1 & n=1 \\ \text{Dividir}(n) + \text{Conquistar}(n) + \text{Combinar}(n) & \end{cases}$$

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

Algoritmos en la computación

Merge sort

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

al resolver la recurrencia, $T(n) = n \lg n$

¿Cómo podemos estimar el costo computacional de algoritmos que siguen la técnica divide y vencerás?

Ecuaciones de Recurrencia

Las ecuaciones de recurrencia permiten expresar el comportamiento computacional de un algoritmo recursivo (en particular, los que siguen la técnica divide y vencerás). La solución de dichas ecuaciones permitirían determinar el costo computacional de estos algoritmos.

Existen diferentes estrategias para solucionar ecuaciones de recurrencia:

- **Método de iteración**
- **Método maestro**
- **Método de sustitución**

Recurrencias

Método de iteración

Expandir la recurrencia y expresarla como una suma de términos que dependen de n y de las condiciones iniciales

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$, $T(1) = \Theta(1)$

Expandir la recurrencia 2 veces

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3\,(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3\,(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)\,))$$

$$n + 3*\lfloor n/4 \rfloor + 3^2*\lfloor n/16 \rfloor + 3^3T(\lfloor n/64 \rfloor)$$

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$

$n + 3\ (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$

$n + 3\ (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor\ + 3T(\lfloor n/64 \rfloor)\ ))$

$n + 3*\lfloor n/4 \rfloor + 3^2*\lfloor n/16 \rfloor\ + 3^3T(\lfloor n/64 \rfloor)$

¿Cuándo se detienen las iteraciones?

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$

$n + 3\ (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$

$n + 3\ (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor\ + 3T(\lfloor n/64 \rfloor)\ ))$

$n + 3 * \lfloor n/4 \rfloor + 3^2 * \lfloor n/16 \rfloor\ + 3^3 T(\lfloor n/64 \rfloor)$

¿Cuándo se detienen las iteraciones?

Cuando se llega a $T(1)$

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$

$n + 3\ (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$

$n + 3\ (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor\ + 3T(\lfloor n/64 \rfloor)\ ))$

$n + 3 * \lfloor n/4 \rfloor + 3^2 * \lfloor n/4^2 \rfloor\ + 3^3 T(\lfloor n/4^3 \rfloor)$

¿Cuándo se detienen las iteraciones?

Cuando se llega a T(1), esto es, cuando $(n/4^i)=1$

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$

$n + 3\,(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$

$n + 3\,(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor \; + 3T(\lfloor n/64 \rfloor)\,))$

$n + 3^{*}\lfloor n/4 \rfloor + 3^{2}{}^{*}\lfloor n/4^{2} \rfloor \; + 3^{3}\lfloor n/4^{3} \rfloor + \ldots + 3^{\log_4 n}T(1)$

¿Cuándo se detienen las iteraciones?

Cuando se llega a T(1), esto es, cuando $(n/4^{i})=1$

Recurrencias

$$T(n) = n + 3T(\lfloor n/4 \rfloor)$$

$$n + 3\,(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$$

$$n + 3\,(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$$

$$n + 3*\lfloor n/4 \rfloor + 3^2*\lfloor n/4^2 \rfloor + 3^3(\lfloor n/4^3 \rfloor) + \ldots + 3^{\log_4 n}\Theta(1)$$

Después de iterar, se debe tratar de expresar como una sumatoria con forma cerrada conocida

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$

$n + 3\ (\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$

$n + 3\ (\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor\ + 3T(\lfloor n/64 \rfloor)\ ))$

$n + 3*\lfloor n/4 \rfloor + 3^2*\lfloor n/4^2 \rfloor\ + 3^3(\lfloor n/4^3 \rfloor) + \ldots + 3^{\log_4 n}\Theta(1)$

$\leq\ n + 3n/4 + 3^2 n/4^2 + 3^3 n/4^3 + \ldots + 3^{\log 4n}\Theta(1)$

Recurrencias

$T(n) = n + 3T(\lfloor n/4 \rfloor)$

$n + 3\,(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor))$

$n + 3\,(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor)))$

$n + 3^*\lfloor n/4 \rfloor + 3^{2*}\lfloor n/4^2 \rfloor + 3^3(\lfloor n/4^3 \rfloor) + \ldots + 3^{\log 4n}\Theta(1)$

$\leq n + 3n/4 + 3^2 n/4^2 + 3^3 n/4^3 + \ldots + 3^{\log 4n}\Theta(1)$

$$= n\sum_{i=0}^{\log_4 n-1}(3/4)^i + 3^{\log_4 n}\Theta(1)$$

$$= n\left(\frac{(3/4)^{\log_4 n}-1}{(3/4)-1}\right) + \Theta(n^{\log_4 3}) = n * 4(1-(3/4)^{\log_4 n}) + \Theta(n^{\log_4 3})$$

Serie geométrica

$$\sum_{k=0}^{n} x^k = x^0 + x^1 + x^2 + x^3 + \cdots + x^n = \frac{(x^{n+1}-1)}{(x-1)}$$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \quad si\ |x| < 1$$

$= O(n)$

Recurrencias

Resuelva por el método de iteración

$T(n) = 2T(n/2) + 1, T(1)= \Theta(1)$

Recurrencias

Resuelva por el método de iteración

$T(n) = 2T(n/2) + 1$, $T(1)= \Theta(1)$

$T(n) = 2T(n/2) + n$, $T(1)= \Theta(1)$

Recurrencias

Resuelva por el método de iteración

$T(n) = 2T(n/2) + 1,\ T(1) = \Theta(1)$

$T(n) = 2T(n/2) + n,\ T(1) = \Theta(1)$

$T(n) = T(\lceil n/2 \rceil) + 1,\ T(1) = \Theta(1)$

Recurrencias

Resuelva por el método de iteración

$T(n) = 2T(n/2) + 1,\ T(1) = \Theta(1)$

$T(n) = 2T(n/2) + n,\ T(1) = \Theta(1)$

$T(n) = T(\lceil n/2 \rceil) + 1,\ T(1) = \Theta(1)$

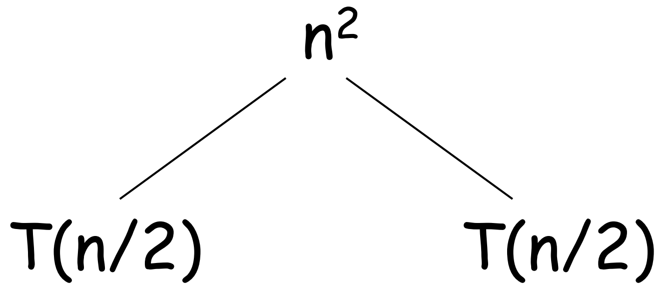
Demuestre que $T(n) = 2T(\lfloor n/2 \rfloor) + n$, es $\Omega(n \log n)$

Recurrencias

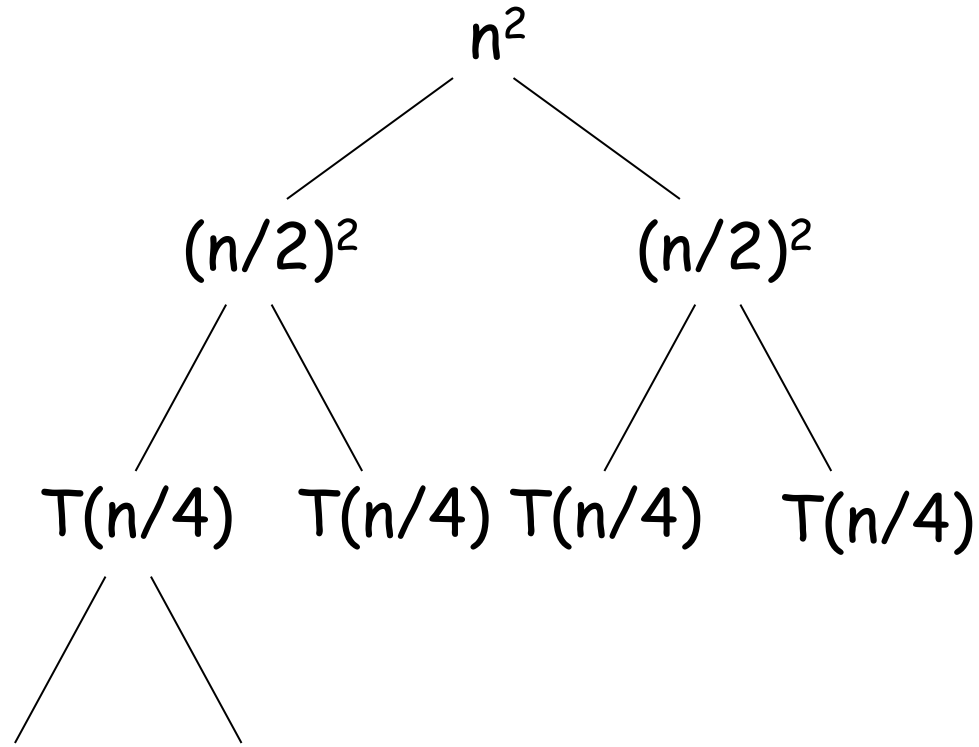
Iteración con árboles de recursión

$T(n) = 2T(n/2) + n^2 \quad T(1)= \Theta(1)$

Recurrencias

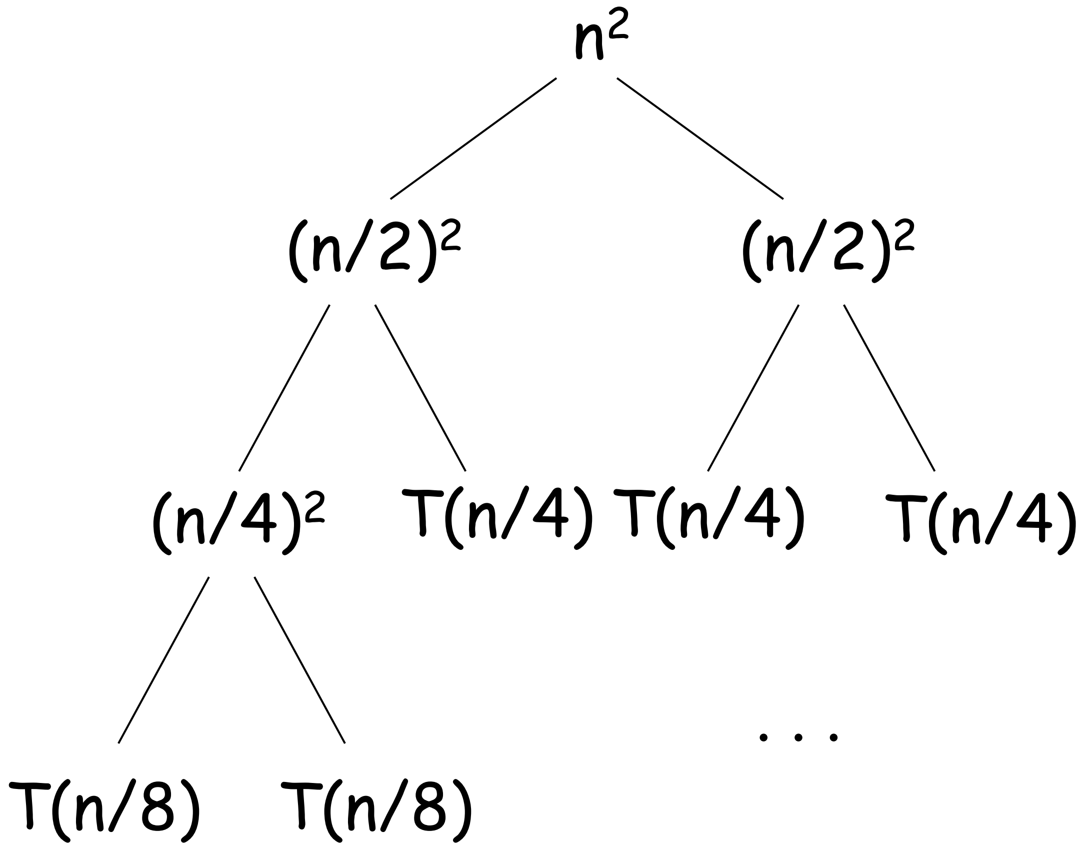


Recurrencias

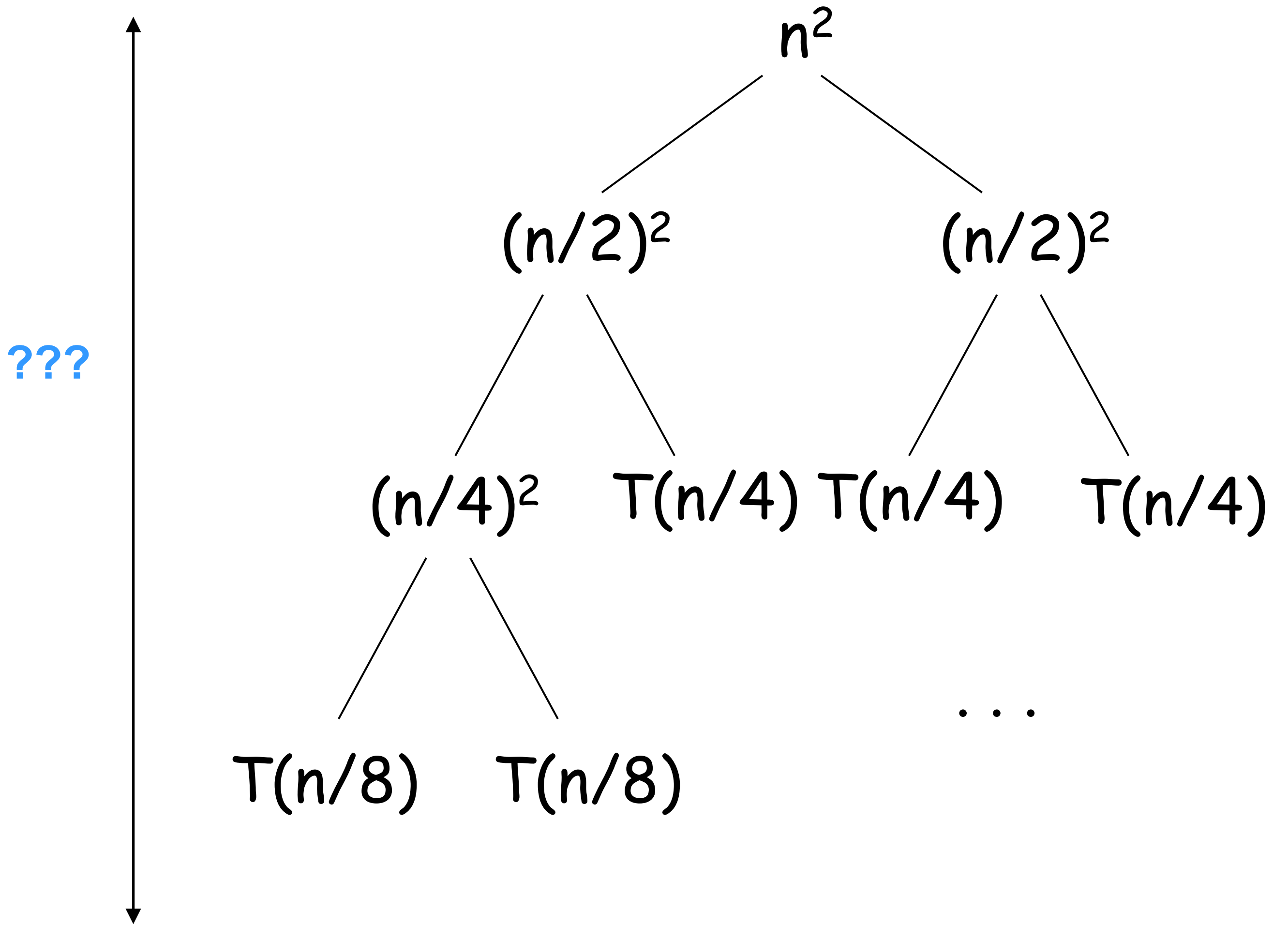


. . .

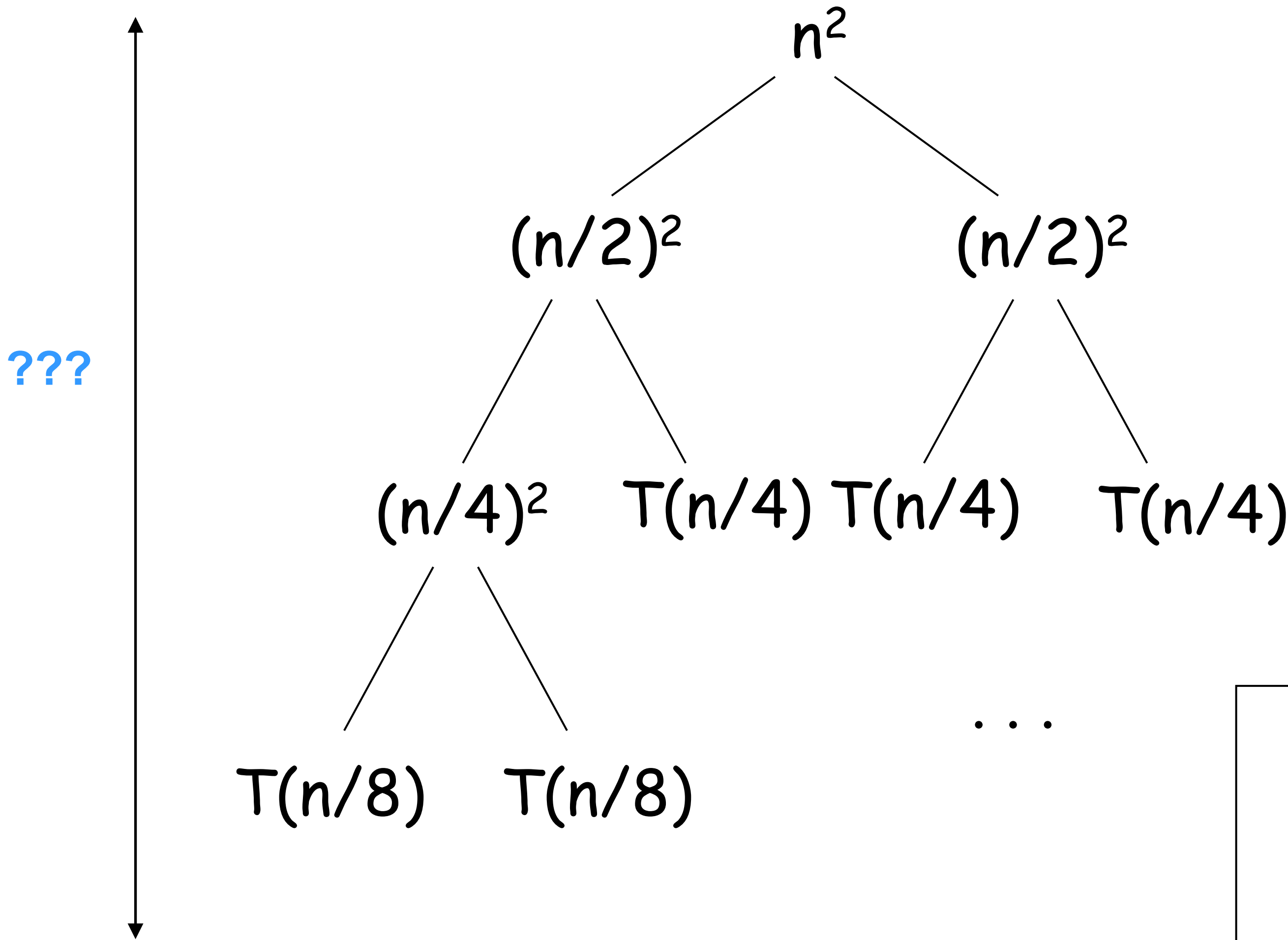
Recurrencias



Recurrencias



Recurrencias

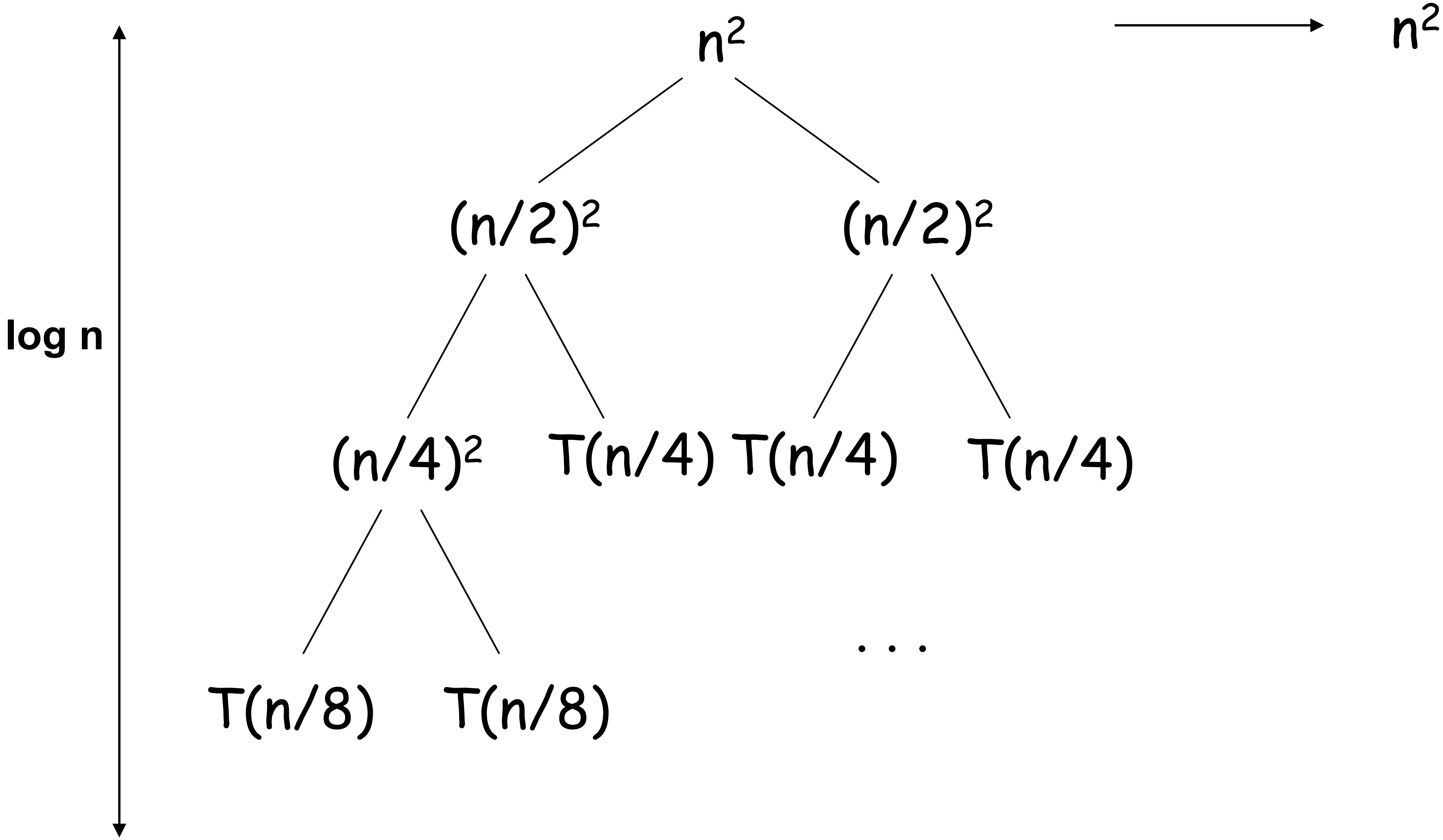


$(n/2^i)=1$

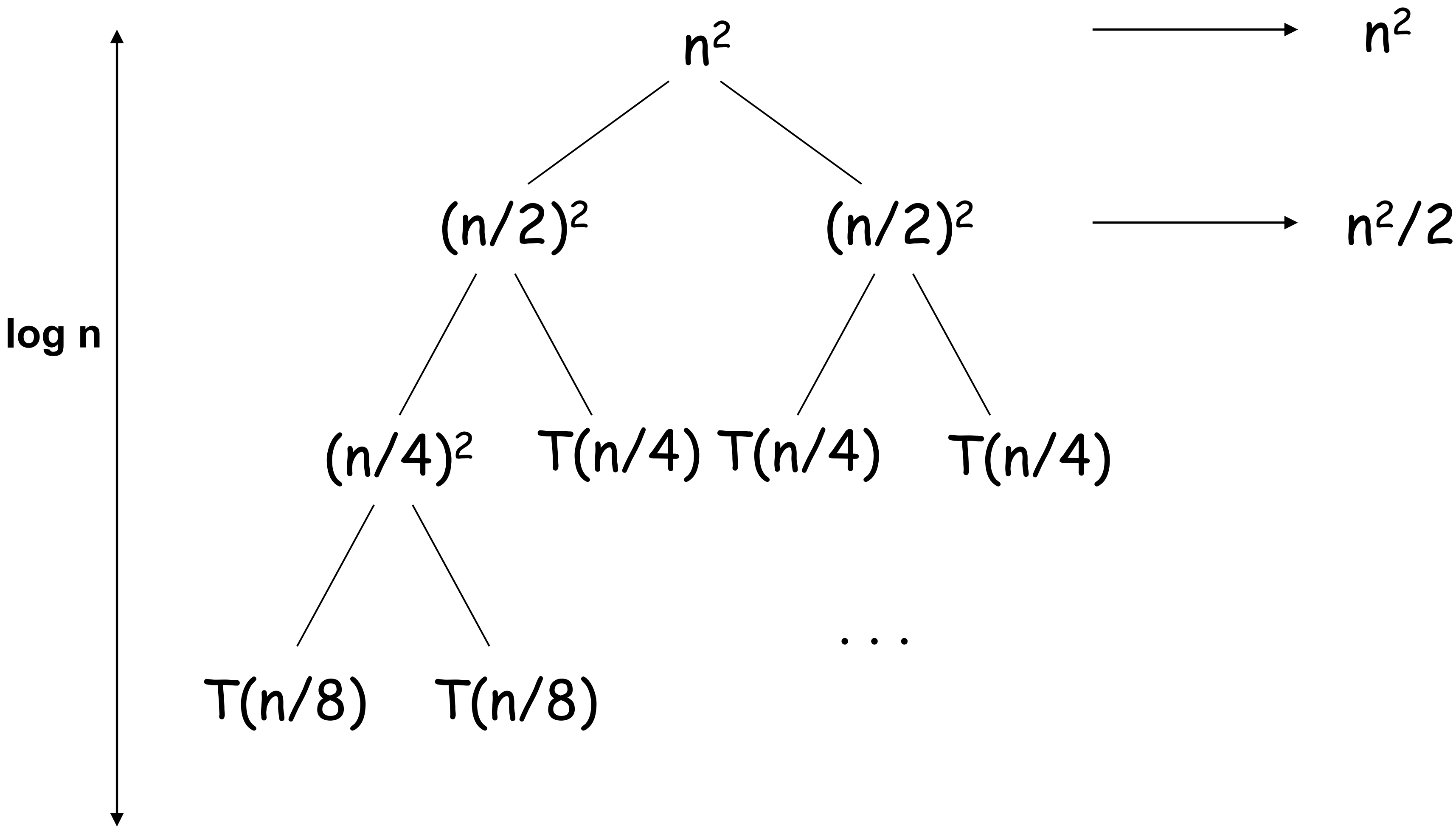
$n=2^i$

$\log n=i$

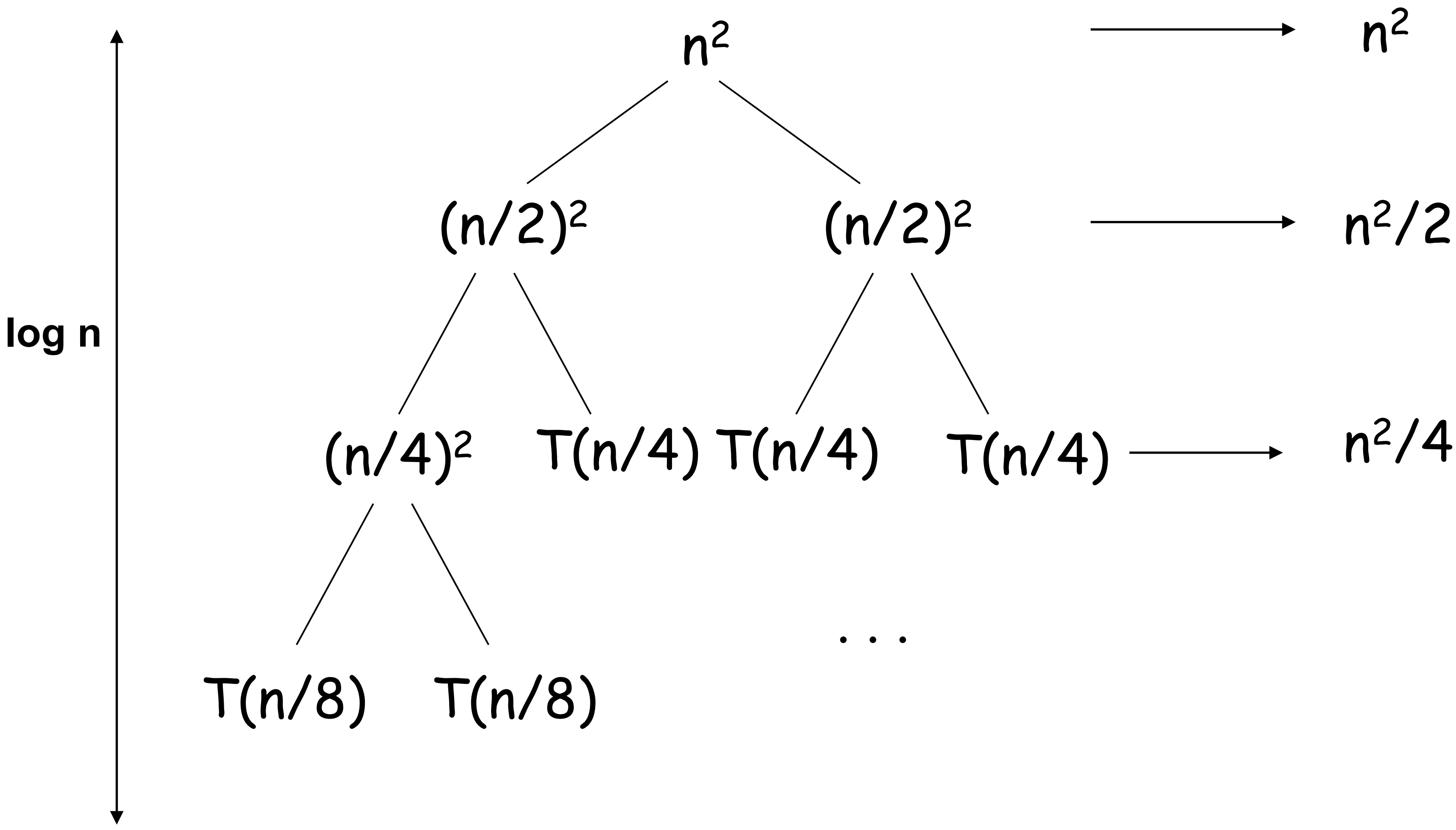
Recurrencias



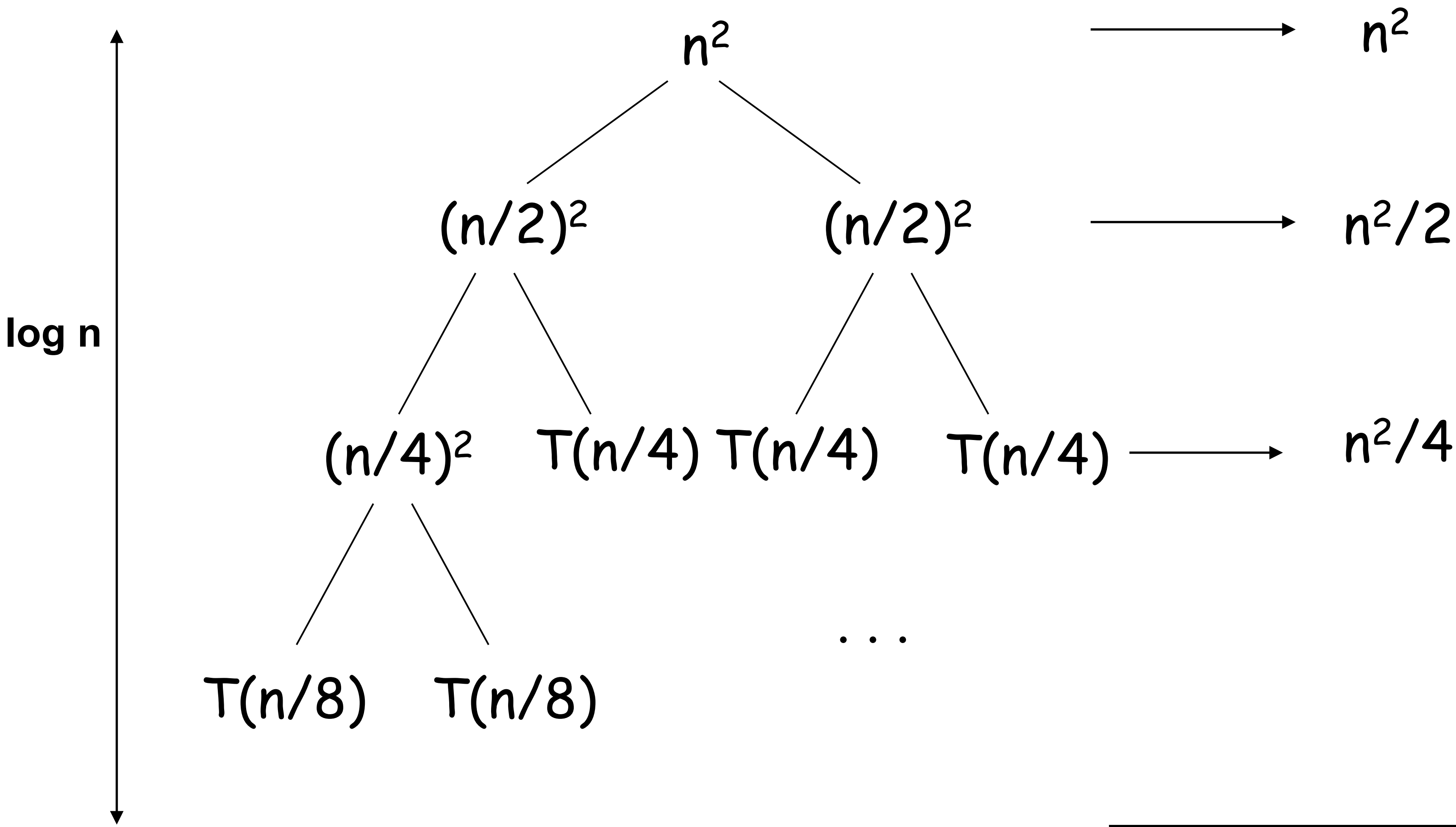
Recurrencias



Recurrencias



Recurrencias



$$\text{Total} = \left(\sum_{i=0}^{\log n-1} \frac{n^2}{2^i}\right) + n = n^2 * \sum_{i=0}^{\log n-1} \left(\frac{1}{2}\right)^i + n$$

Recurrencias

$$Total = (\sum_{i=0}^{\log n-1} \frac{n^2}{2^i}) + n = (n^2 * \sum_{i=0}^{\log n-1} \left(\frac{1}{2}\right)^i) + n$$

$$Total = n^2 * \frac{(1/2)^{(\log n)} - 1}{1/2 - 1} + n = \Theta(n^2)$$

Serie geométrica

$$\sum_{k=0}^{n} x^k = x^0 + x^1 + x^2 + x^3 + \cdots + x^n = \frac{(x^{n+1} - 1)}{(x - 1)}$$

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \quad si\ |x| < 1$$

Recurrencias

Resuelva construyendo el árbol

$T(n) = 2T(n/2) + 1$, $T(1) = \Theta(1)$

$T(n) = 2T(n/2) + n$, $T(1) = \Theta(1)$

Recurrencias

Resuelva la recurrencia $T(n) = T(n/3) + T(2n/3) + n$

Indique una cota superior y una inferior

Recurrencias

Método maestro

Permite resolver recurrencias de la forma:

$$T(n) = aT(n/b) + f(n), \text{ donde } a \geq 0,\ b>1$$

Recurrencias

Dado $T(n) = aT(n/b) + f(n)$, donde $a \geq 1$, $b>1$, se puede acotar asintóticamente como sigue:

1. $T(n) = \Theta(n^{\log_b a})$

Si $f(n) = O(n^{\log_b a-\varepsilon})$ para algún $\varepsilon > 0$

2. $T(n) = \Theta(n^{\log_b a} \lg n)$ $\qquad T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

Si $f(n) = \Theta(n^{\log_b a})$ $\qquad$ Si $f(n) = \Theta(n^{\log_b a} \lg^{k} n)$ para algún $k \geq 0$

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

Si $f(n) = \Omega(n^{\log_b a+\varepsilon})$ para algún $\varepsilon > 0$ y si $af(n/b) \leq cf(n)$ para algún $c<1$

Recurrencias

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

$$n^{\log_3 9} = n^2 \qquad \text{vs} \qquad f(n) = n$$

Es $f(n) = O(n^{\log_b a - \varepsilon})$?

Es $n = O(n^{2-\varepsilon})$?

Recurrencias

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

$$n^{\log_3 9} = n^2 \qquad \text{vs} \qquad f(n) = n$$

Es $f(n) = O(n^{\log_b a - \varepsilon})$?

Es $n = O(n^{2-\varepsilon})$?

Si $\varepsilon = 1$ se cumple que $n = O(n)$, por lo tanto, se cumple que:

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

Recurrencias

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

$$n^{\log_{3/2} 1} = n^0 = 1 \quad \text{vs} \quad f(n) = 1$$

Es $f(n) = O(n^{\log_b a - \varepsilon})$?

Es $1 = O(n^{0-\varepsilon})$?

No existe $\varepsilon > 0$

Recurrencias

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

$$n^{\log_{3/2} 1} = n^0 = 1 \quad \text{vs} \quad f(n) = 1$$

Es $f(n) = \Theta(n^{\log_b a})$?

Es $1 = \Theta(1)$?

Si, por lo tanto, se cumple que:

$$T(n) = \Theta(1 * \lg n) = \Theta(\lg n)$$

Recurrencias

T(n) =3 T(n/4) + nlgn

$$n^{\log_4 3} = n^{0.793} \quad \text{vs} \quad f(n) = n\lg n$$

Es $f(n) = O(n^{\log_b a-\varepsilon})$?

Es $f(n) = \Theta(n^{\log_b a})$?

Es $f(n) = \Omega(n^{\log_b a+\varepsilon})$?

Si, y además, af(n/b)≤cf(n)

3(n/4)lg(n/4) ≤ cnlgn

3(n/4)lgn - 3(n/4)*2 ≤ cnlgn

(3/4)nlgn ≤ cnlgn → c=3/4 y se concluye que $T(n) = \Theta(n\lg n)$

Recurrencias

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

Analice si se puede resolver por el método maestro

Recurrencias

Resuelva en los casos que sea posible usando el metodo maestro:

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

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

$T(n) = 4T(n/2) + n^3$

Recurrencias

Método de sustitución

Suponer la forma de la solución y probar por inducción matemática.

-- Se empieza conjeturando cuál puede ser la solución de la ecuación de recurrencia a partir de la similitud de esta con otra cuya solución es conocida.

-- Se demuestra por inducción matemática que la ecuación tiene dicha solución.

Recurrencias

¿Cuál es la solución de $T(n)=2T(\lfloor n/2 \rfloor)+n$, $T(1)=1$?

Para aplicar el método de sustitución, se empieza identificando una ecuación parecida para la cual se conozca la solución. Sea $T'(n)=2T'(n/2)+n$, $T(1)=1$; dado que, por método maestro, $T'(n)=O(n\log n)$ entonces la solución que se conjetura es que $T(n)=O(n\log n)$ también.

Se va a probar que siendo $T(n)=2T(\lfloor n/2 \rfloor)+n$, $T(1)=1$; $T(n) = O(n\log n)$.
En concreto se va a demostrar, por inducción, que $T(n) \leq cn\log n$.

Recurrencias

$T(n)=2T(\lfloor n/2 \rfloor)+n,\ T(1)=1$

Suponer que la solución es de la forma $T(n)=O(n\log n)$

Probar que $T(n) \le cn\log n$ (inducción matemática).

Caso inductivo:

Si se cumple la propiedad para $T(\lfloor n/2 \rfloor)$, entonces se cumple la propiedad para $T(n)$.

Se supone que se cumple para $\lfloor n/2 \rfloor$ y se prueba para n

Hipotesis inductiva: $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)$

Recurrencias

$T(n)=2T(\lfloor n/2 \rfloor)+n$

Probar que $T(n) \leq cn\log n$.

Caso inductivo:

Si $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \log (\lfloor n/2 \rfloor)$ entonces $T(n) \leq cn\log(n)$

Hipótesis inductiva: $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \log (\lfloor n/2 \rfloor)$

Paso inductivo:

$$
\begin{aligned}
T(n) &\leq 2(c\lfloor n/2 \rfloor \log (\lfloor n/2 \rfloor)) + n \\
&\leq cn \log (n/2) + n \\
&= cn \log (n) - cn + n, \text{ para } c \geq 1, \\
&\leq cn \log n
\end{aligned}
$$

Recurrencias

$T(n)=2T(\lfloor n/2 \rfloor)+n$, $T(1)=1$

Probar que $T(n) \leq cn\lg n$.

Paso base: si $c=1$, probar que $T(1)=1$ se cumple

$T(1) \leq 1{*}1\log 1$?

$1 \leq 0$?

No, se debe escoger otro valor para c

Recurrencias

$T(n)=2T(\lfloor n/2 \rfloor)+n$, $T(1)=1$

Probar que $T(n) \leq cn\lg n$.

Paso base: si $c=2$, probar que $T(1)=1$ se cumple

$T(1) \leq 2*1\lg 1$?

$1 \leq 0$?

No,

Para esto, se calcula $T(2)$ y se toma como valor inicial

Recurrencias

Probar que $T(n) \leq cn\log n$.

$T(2)=2T(0)+2=4$

Paso base: si $c=1$, probar que $T(2)=4$ se cumple

$T(2) \leq 1*2\log 2$?

$4 \leq 2$?

No, se puede variar c.

Recurrencias

Probar que $T(n) \leq cnlgn$.

$T(2)=2T(0)+2=4$

Paso base: si $c=3$, probar que $T(2)=4$ se cumple

$T(2) \leq 3*2lg\ 2$?

$4 \leq 6$?

Si, se termina la demostración.

Como se demostró que $T(n) \leq cnlgn$ para todo valor $n \geq 2$, se concluye que $T(n)$ es $O(nlog\ n)$