Fundamentos de análisis y diseño de algoritmos

Recurrencias

Método de iteración

Método maestro*

Método de sustitución

Análisis de algoritmos recursivos

Recurrencias

Método de iteración

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

Recurrencias

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

Expandir la recurrencia 2 veces

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

$T(n) = n + \frac{3n}{4} + 3^2 T(n/4^2)$ Primera expansión

$T(n/4^2) = \frac{n}{4^2} + 3T(n/4^3)$

$T(n) = n + \frac{3n}{4} + \frac{3^2 n}{4^2} + 3^3 T(n/4^3)$ Segunda expansión

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

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

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

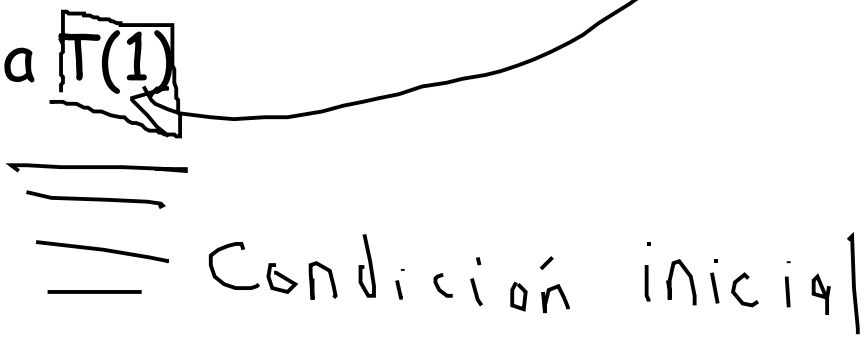
$n + 3\,(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))$

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

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

¿Cuándo se detienen las iteraciones?

Cuando se llega a $T(1)$



Recurrencias

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

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

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

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

Expansión i veces

¿Cuándo se detienen las iteraciones?

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

$\frac{n}{4^i} = 1 \qquad n = 4^i \qquad \log_4(n) = i \qquad n/4^i = 1$

Recurrencias

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

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

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

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

$n + 3\lfloor n/4^1 \rfloor + 3^2 n/4^2 + 3^3 n/4^3 + 3^4 n/4^4 + \cdots 3^i T(1)$

¿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

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

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

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

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

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

Recurrencias

$T(n) = n + 3T(n/4⌋)$

$n + 3 (n/4⌋ + 3T(n/16⌋))$

$n + 3 (n/4⌋ + 3(n/16⌋ + 3T(n/64⌋)))$

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

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

Recurrencias

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

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

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

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

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

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

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

$= O(n)$

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

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

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

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

Recurrencias

Resuelva por el método de iteración

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

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

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

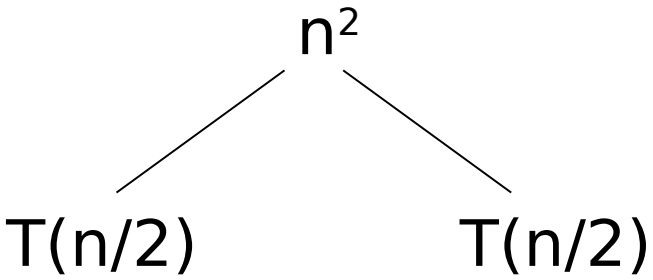
Demuestre que $T(n) = T(n/2\rfloor) + n$, es $\Omega(n\log n)$

Recurrencias

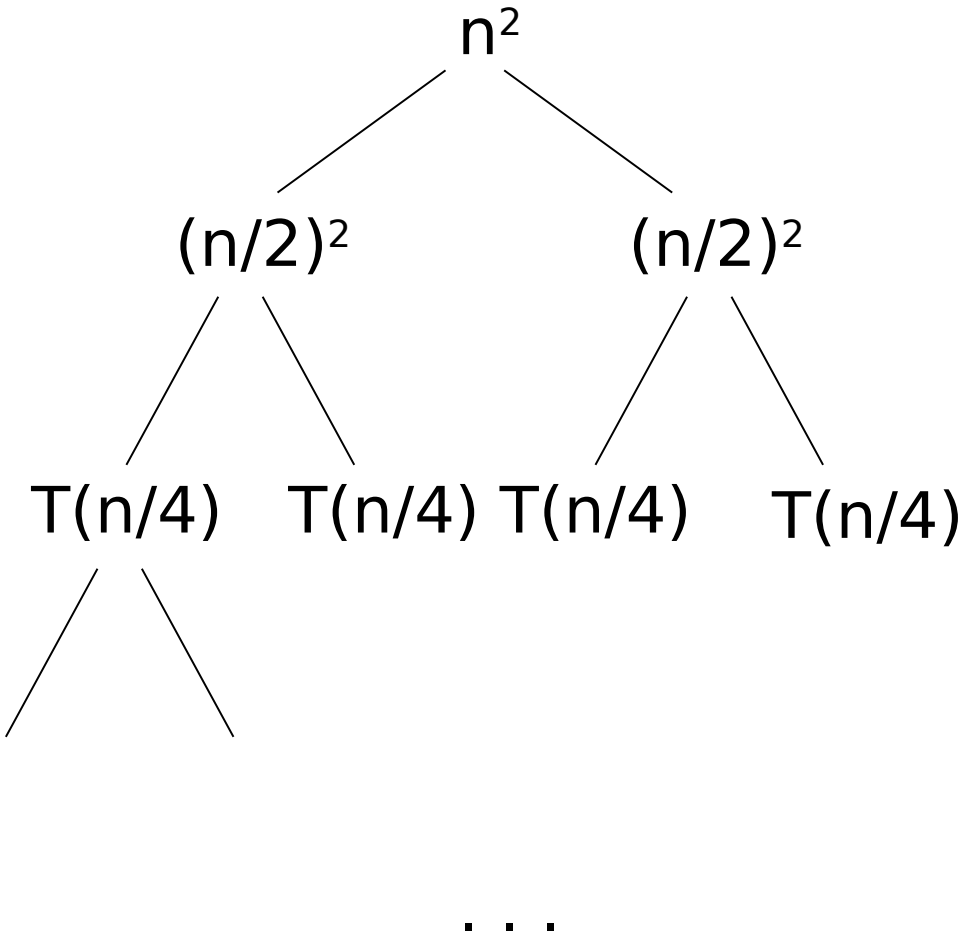
Iteración con árboles de recursión

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

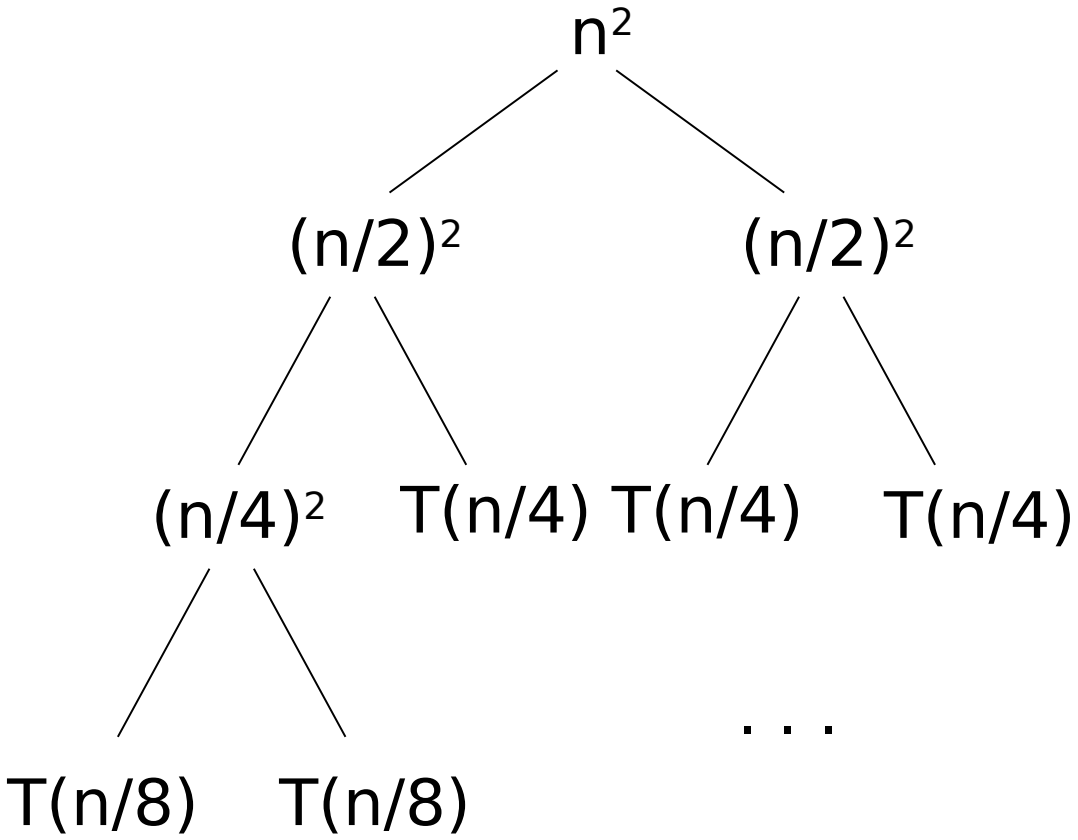
Recurrencias



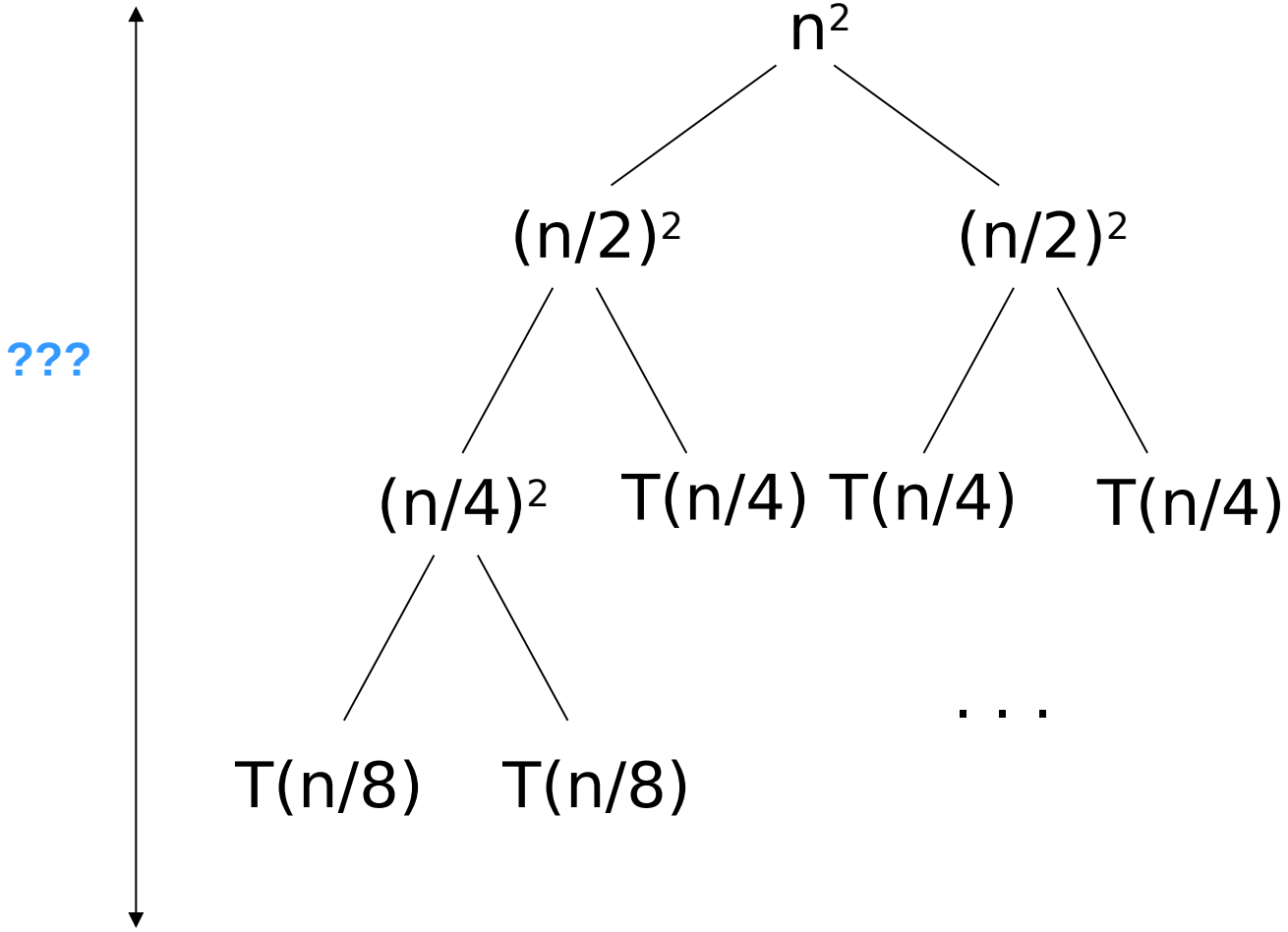
Recurrencias



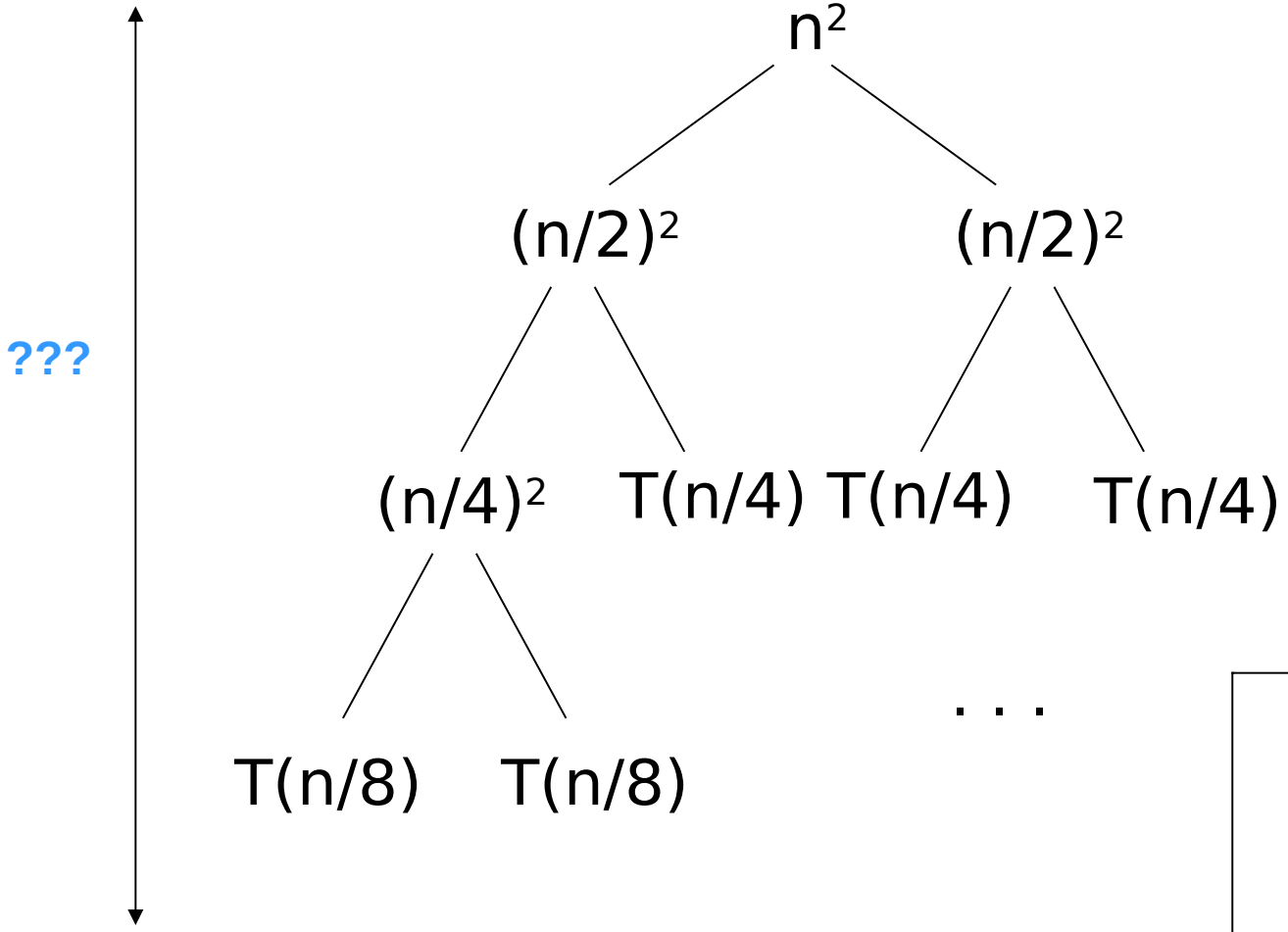
Recurrencias



Recurrencias



Recurrencias



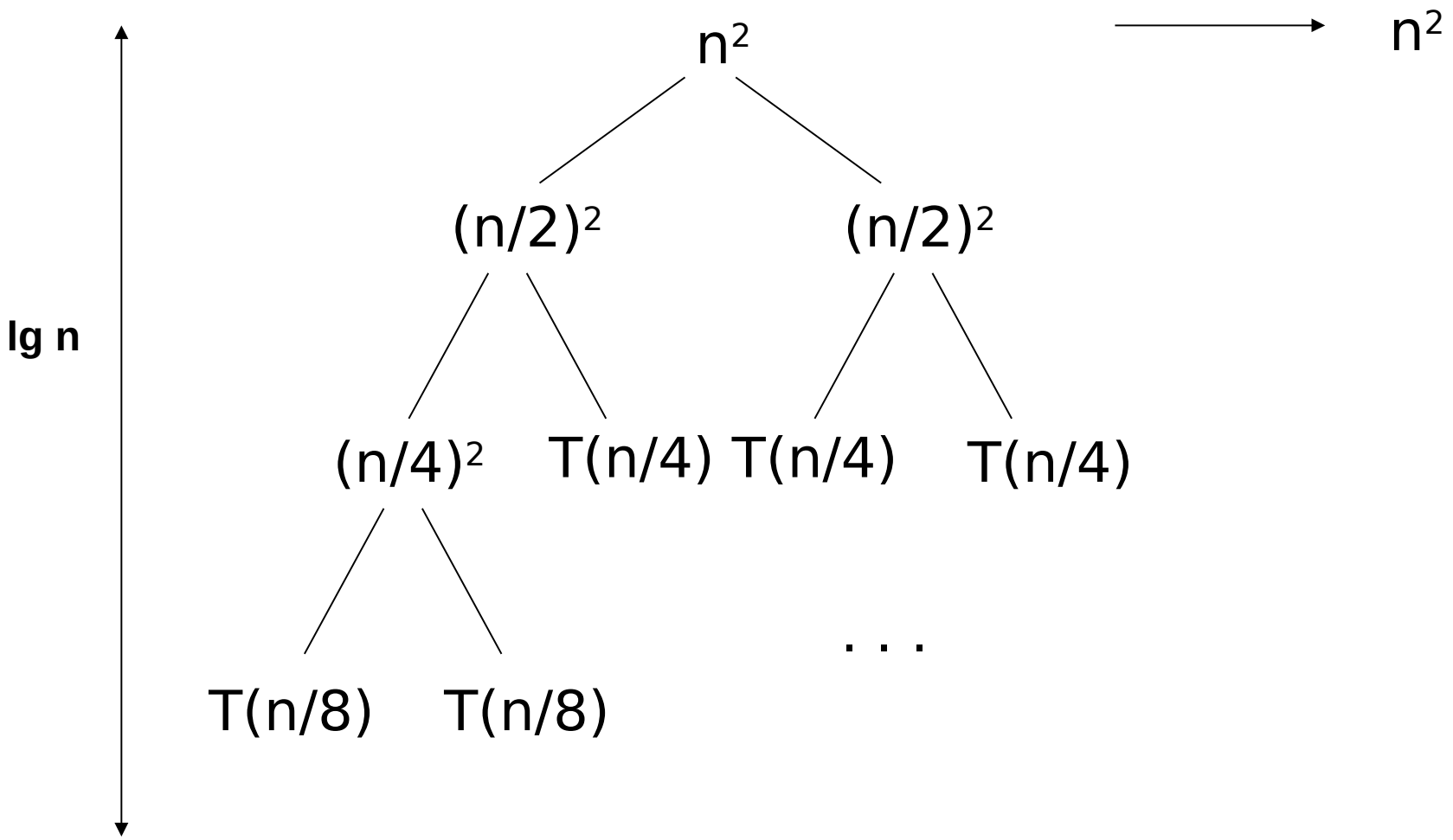
$(n/2^i)^2=1$

$(n/2^i)=1$

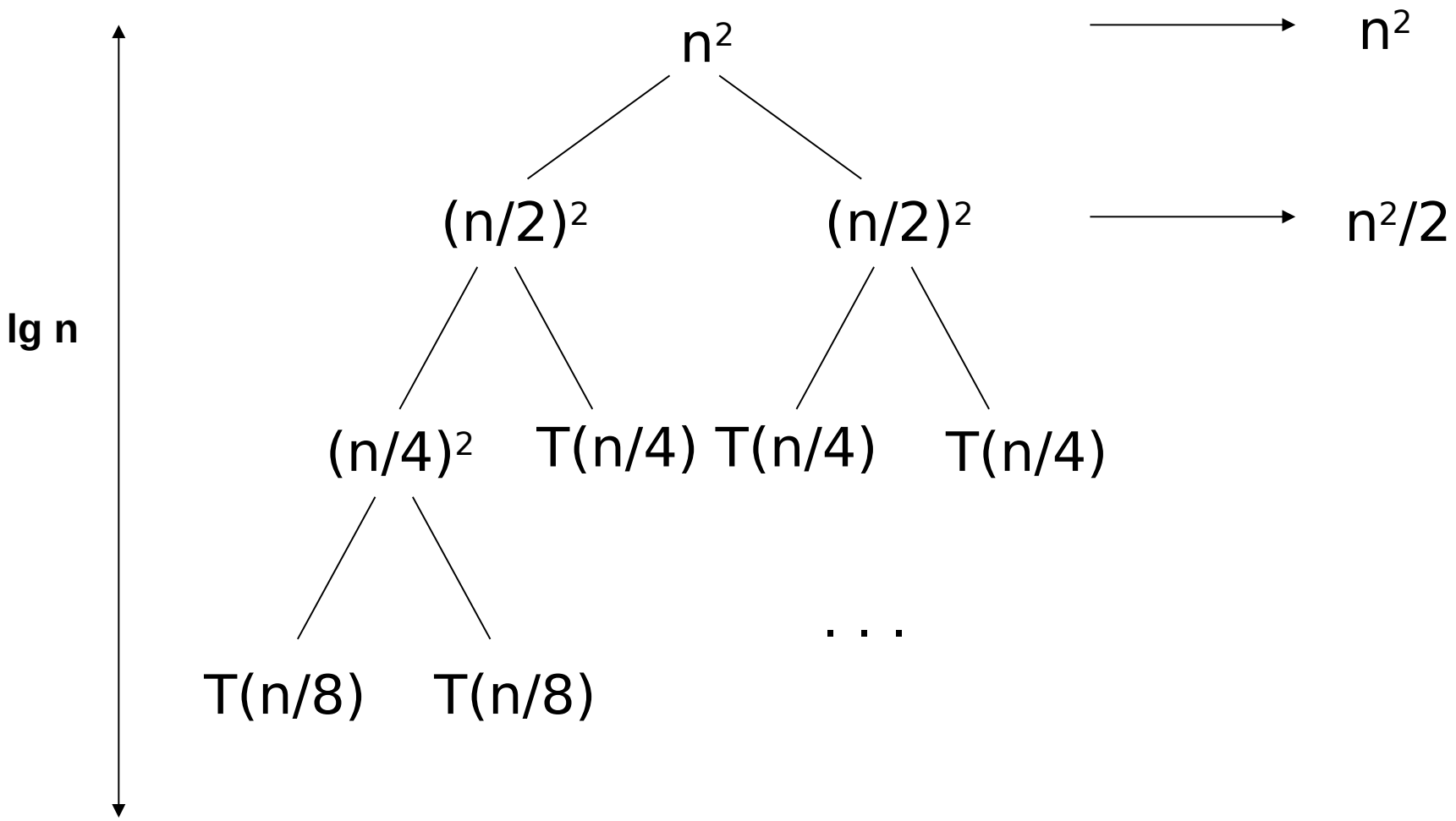
$n=2^i$

$\log n=i$

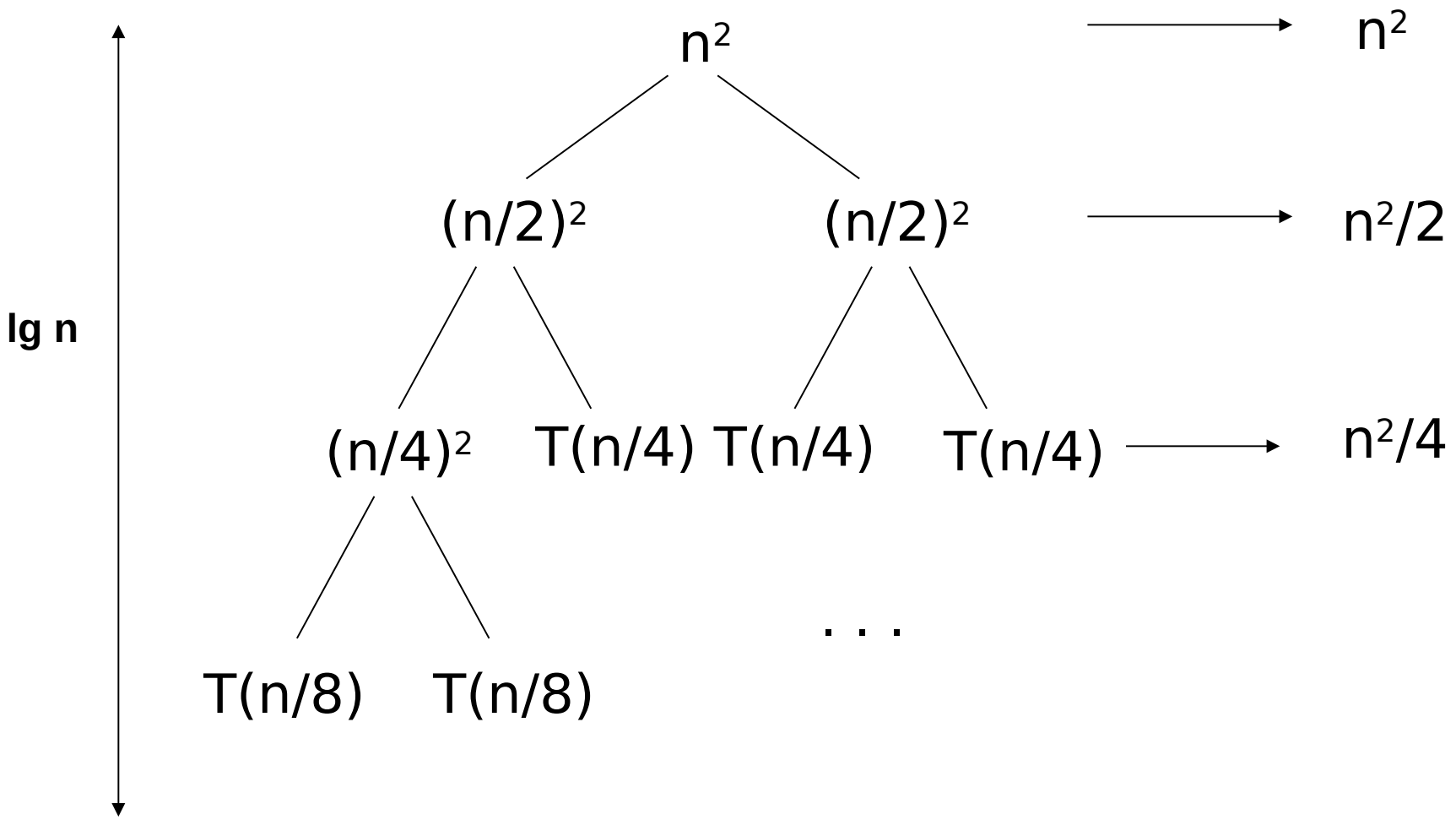
Recurrencias



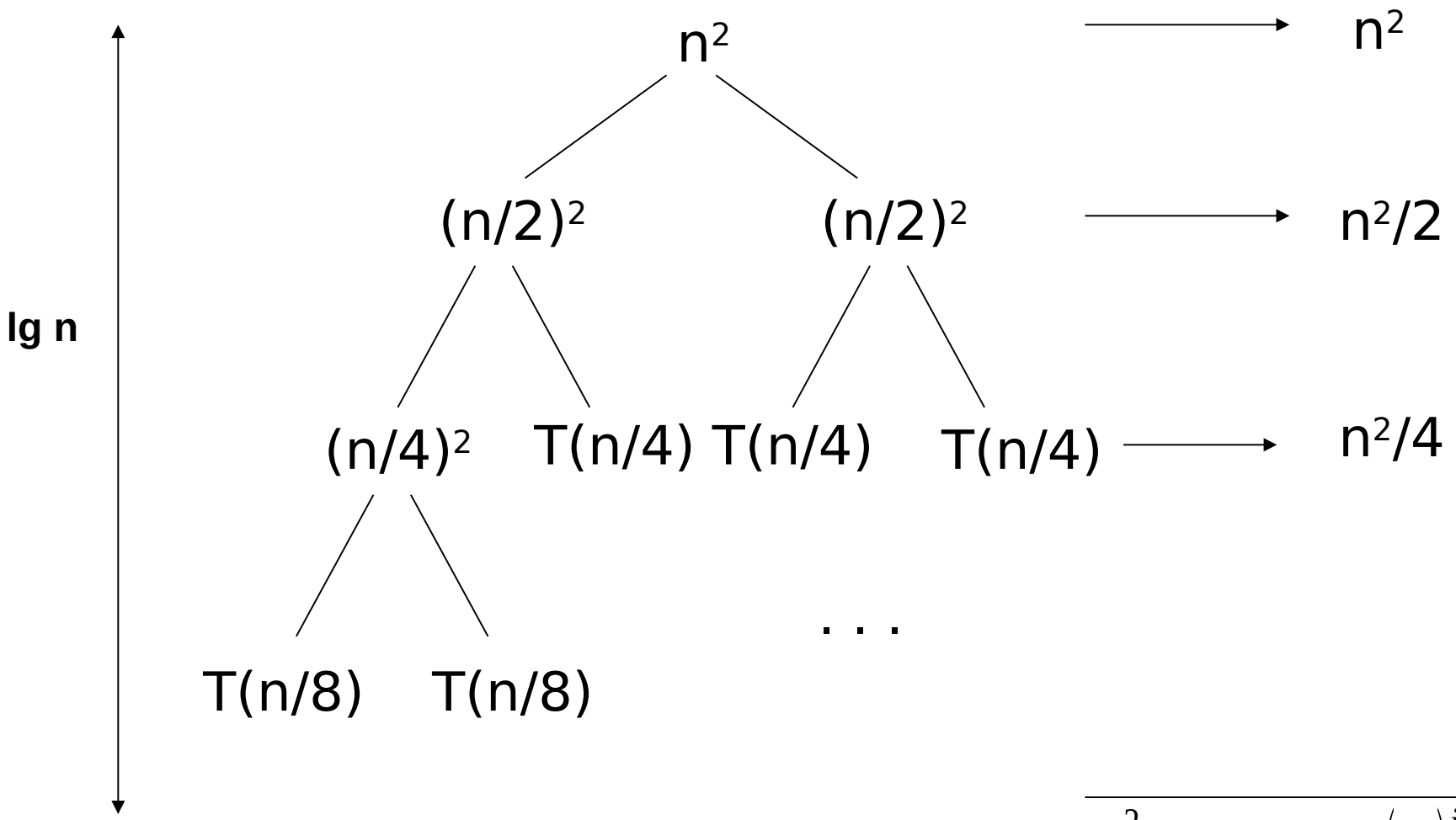
Recurrencias



Recurrencias



Recurrencias



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

Recurrencias

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

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

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:

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

Recurrencias

Dado T(n) = aT(n/b) + f(n), donde a≥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. $T(n)=\Theta(n^{\log_b a}\lg n)$

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

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

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

para algun c<1

Recurrencias

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

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

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

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

Recurrencias

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

$$n^{\log_3 9} = n^2 \qquad \textbf{Vs} \qquad f(n) = n$$

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

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

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

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

Recurrencias

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

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

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

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

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

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

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, $af(n/b) \leq cf(n)$

$3(n/4)\lg(n/4) \leq cn\lg n$

$3(n/4)\lg n - 3(n/4)*2 \leq cn\lg n$

$(3/4)n\lg n \leq cn\lg n \rightarrow c=3/4$ y se concluye que $T(n) = \Theta(n \lg n)$

Recurrencias

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

Muestre que no se puede resolver por el método maestro

Recurrencias

Resuelva

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

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

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

Recurrencias

Método de sustitución

Suponer la forma de la solución y probar por inducción matemática

Recurrencias

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

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

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

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

Hipotesis inductiva: $T(n/2) \leq cn/2\lg (n/2)$

Recurrencias

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

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

Hipótesis inductiva: $T(n/2) \leq cn/2\lg(n/2)$

Paso inductivo:

$T(n) \leq 2(cn/2\lg(n/2)) + n$

$\leq cn \lg(n/2) + n$

$= cn \lg(n) - cn + n$, para $c \geq 1$, haga $c=1$

$\leq cn \lg n$

Recurrencias

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

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

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

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

$1 \leq 0$?

No, se debe escoger otro valor para c

Recurrencias

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

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, se puede variar k.

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

Recurrencias

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

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

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

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

$4 \leq 2$?

No, se puede variar c.

Recurrencias

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

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

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

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

$4 \leq 6$?

Si, se termina la demostración

Análisis de algoritmos recursivos

Un algoritmo recursivo tiene las siguientes partes:

1) Una condición de parada

2) Un llamado recursivo

El análisis de estos algoritmos lo realizaremos analizando

1) Su complejidad en un llamado

2) Cómo es el llamado recursivo y cómo cambia la entrada a medida que se realizan los llamados

3) Cómo es la forma de la entrada para llegar a la condición de parada

Análisis de algoritmos recursivos

Ejemplo, pensemos en este algoritmo para calcular la serie de Fibunnaci para un número (n) dado

Recuerda:

$$f(n)=f(n-1)+f(n-2), f(0)=1, f(1)=1$$

fibunnaci(n)

Si n = 0 retorne 1

Sino si n = 1 retorne 2

Sino fibunnaci(n -1) + fibunnaci(n - 2)

Análisis de algoritmos recursivos

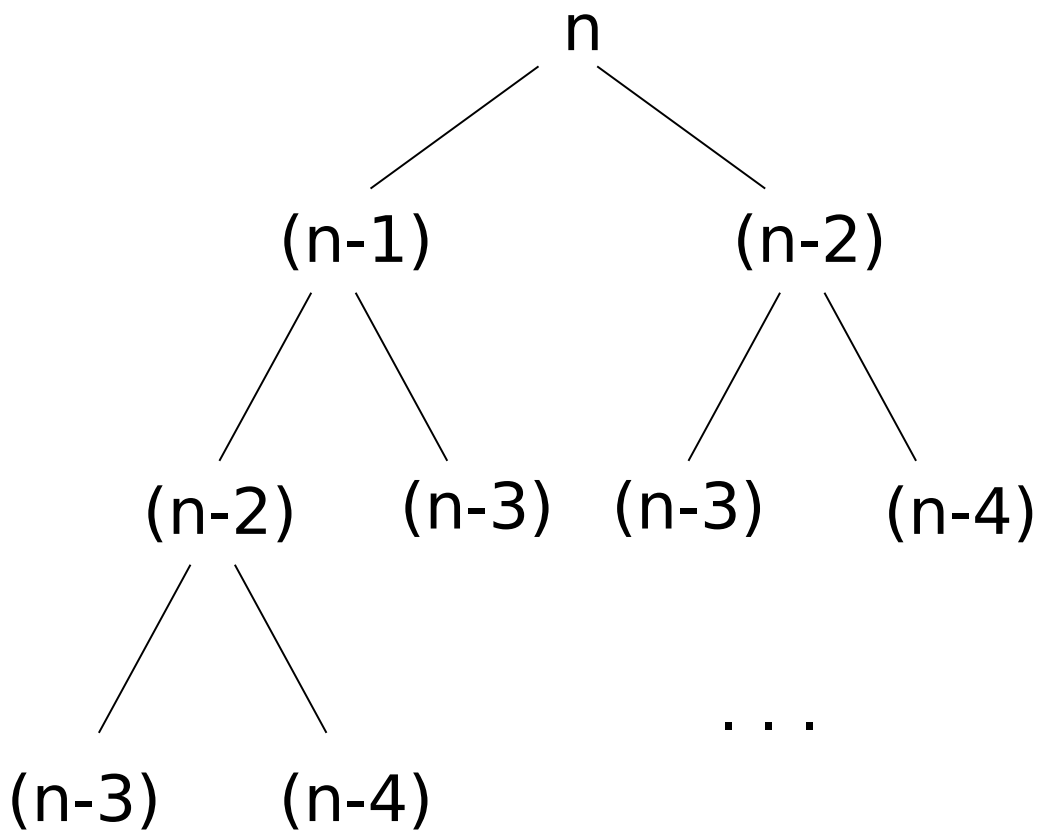
Si

Recuerda:

$$f(n)=f(n-1)+f(n-2), f(0)=1, f(1)=1$$

```
fibunnaci(n)
    Si n = 0 retorne 1
    Sino si n = 1 retorne 2
        Sino fibunnaci(n -1) + fibunnaci(n - 2)
```

Análisis de algoritmos recursivos



Análisis de algoritmos recursivos

Si observamos para cada llamado de n se realiza el llamado para n -1 y n -2, la parada se encuentra cuando n = 0 y n = 1 entonces, la complejidad de este algoritmo está dada por la relación

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

Se cuenta O(1) en cada llamado debido a la verificaciones que debe realizar, las cuales son ejecutadas en **tiempo constante**

Si observa en las condiciones de parada el tiempo también es constante, entonces:

$$T(0)=O(1), T(1)=1$$

Análisis de algoritmos recursivos

Siempre que se trabaje con algoritmos recursivos el cálculo de la complejidad va tener la forma

$$T(n)=T(x_1)+T(x_2)+...T(x_n)+f(n)$$

Donde $T(x_i)$ indica cómo cambia la entrada en cada llamado recursivo y $f(n)$ representa la complejidad de los procesos realizados en un **sólo** paso.

Análisis de algoritmos recursivos

Tarea

Analizar el algoritmo Merge Sort para estas condiciones

Mejor caso

Particiones: n/2 y n/2 complejidad ordenar O(1).

Caso promedio

Particiones: n/8, 7n/8 complejidad ordenar O(n)

Peor caso

Particiones n/2 y n/2 complejidad ordenar O(n)

Análisis de algoritmos recursivos

Tarea

Análizar algoritmos que tienen el siguiente comportamiento.

Algoritmo 1

Particiones: n -2 y n – 4, complejidad cada paso O(n)

Algoritmo 2

Particiones: n/2 y n/3, complejidad cada paso O(log(n))

Análisis de algoritmos recursivos

Tarea

Analizar el siguiente algoritmo:

```
Algoritmo(n)
        Si n = 0 retorne 1
        Si n = 1 retorne 2
        Sino Si n es par  retorne n + f( ⌊n/2⌋ )
```