```
MergeSort (A, p, r)
  Se p < r
     q ← ⌊(p+r)/2⌋
     MergeSort (A, p, q)
     MergeSort (A, q+1, r)
     Intercala (A, p, q, r)
```

Como descrever o tempo a partir das instruções do algoritmo?

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

Algoritmos recursivos:

De um modo geral, seu tempo de execução é calculado em função do próprio tempo para entradas menores

↳ Relações de recorrência

* Recorrência do merge: é suficiente para calcular o tempo?
* O que a recorrência diz sobre o tempo?

Resolução de relação de recorrência

Relação de recorrência:
Equação (ou desigualdade) que descreve uma função em termos de valores menores de seus entradas.

Ex: Fibonacci:
$F(0) = 1$
$F(1) = 1$
$F(K) = F(K-1) + F(K-2)$

Fatorial: $F(1) = 1$
$F(K) = K \cdot F(K-1)$

Resolver uma relação de recorrência:
- Determinar a função correspondente a ela, dependendo apenas do valor de n.
↳ Pode ser calculado exatamente em alguns casos:

Ex: $f(0) = 1$
$f(K) = 3 f(K-1), \forall K > 0$

$f(K) = 3^K$

Base: $f(0) = 1 = 3^0$

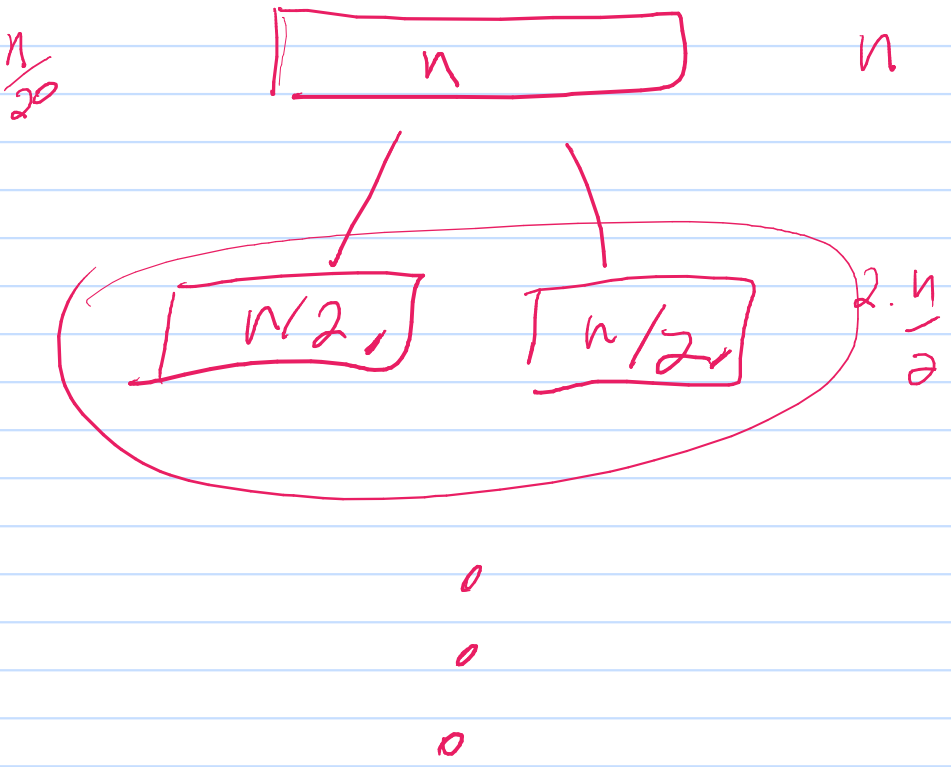
Hip: $< K$

Passo
$f(K) = 3 \cdot f(K-1)$
$= 3 \cdot 3^{K-1}$
$= 3^K$

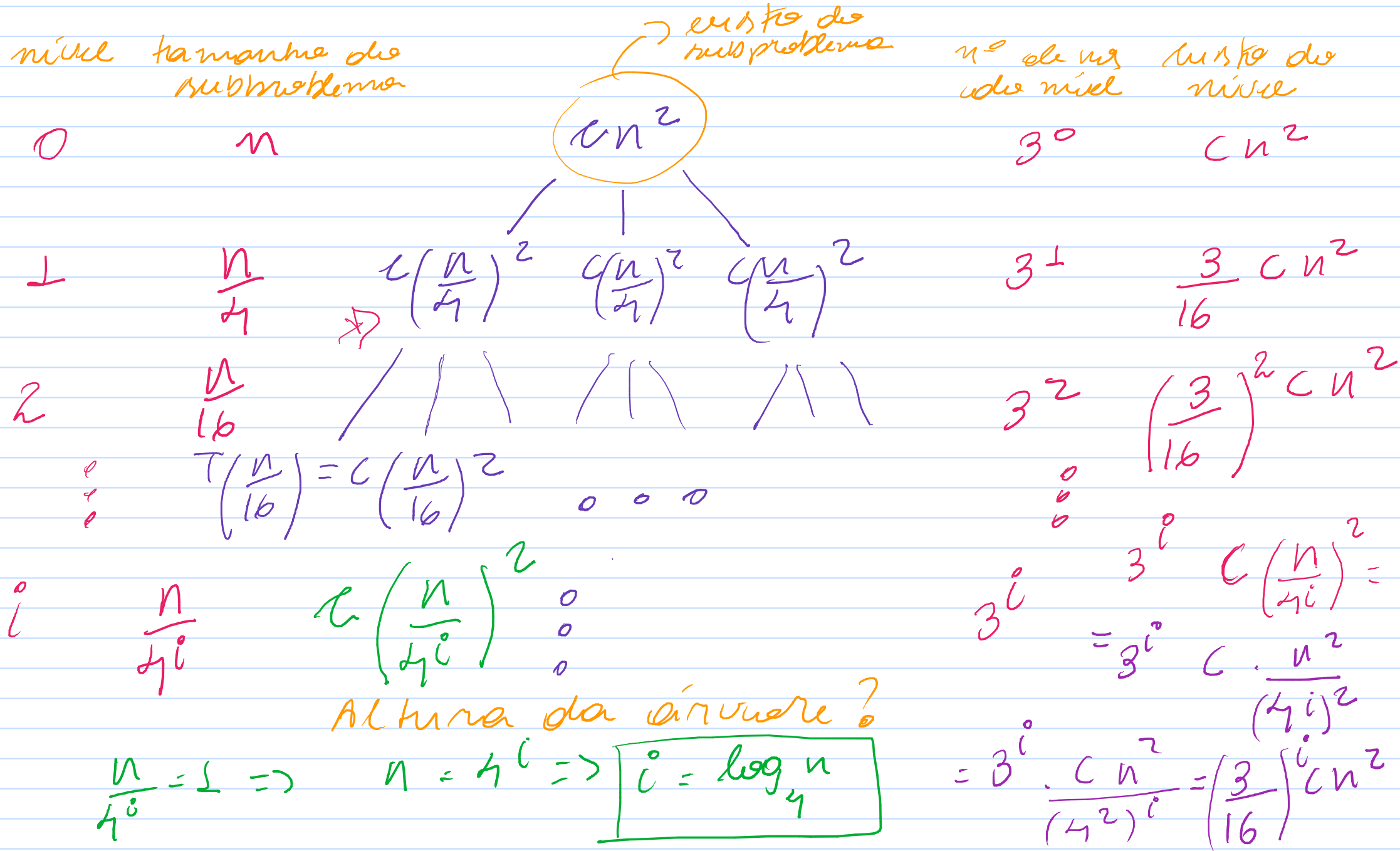
↳ Para algoritmos, calcularemos os limites assintóticos.

Relações de recorrência: Como resolver?

1) Método da árvore de recursão
   - Cada nó representa o custo de um subproblema (calcular tamanho do subproblema no nível i)
   - Somando o custo de cada nó em um nível, obtemos o custo do nível (calcular o número de nós em cada nível)
   - Somando o custo de cada nível, determinamos o custo total (calcular a altura da árvore)



Ex: $T(n) = 3T(n/4) + cn^2$ (Encontrar limite superior)



Altura da árvore?

$\frac{n}{4^i} = 1 \Rightarrow n = 4^i \Rightarrow \boxed{i = \log_4 n}$

custo total:

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n} cn^2$$

$$= cn^2 \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i$$

$$\leq cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i$$

$$= cn^2 \cdot \frac{1}{1-(3/16)}$$

$$= \frac{16}{13} cn^2$$

$$= O(n^2)$$

2) Resolução iterativa

Ex 2: Busca binária

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

$$= T(n/4) + 1 + 1 = T\left(\frac{n}{2^2}\right) + 2$$

$$= T(n/8) + 1 + 1 + 1 = T\left(\frac{n}{2^3}\right) + 3$$

$$\vdots$$

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

$$= 1 + \log n = \Theta(\log n)$$

Ex 3: $T(1) = 0$

$$T(n) = T(n-1) + \frac{1}{n}$$

$$= T(n-2) + \frac{1}{n-1} + \frac{1}{n}$$

$$= T(n-3) + \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n}$$

$$\vdots$$

$$= H_n = \ln n + O(1)$$

↳ Número harmônico

3) Método da Substituição
↳ Indução

Ex: Merge

$T(1) = \Theta(1)$

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

$$T(n) = O(n \log n) \leq c\, n \log n$$

base: $n = 2$

$T(2) = 2\,T(1) + 2 = 4$

$\leq c \cdot 2 \log 2$

$c \geq 2$

hip.: $T(n) \leq c\, n \log n$ p/ $n < k$

passo: $n = k$

$T(k) = 2\,T\left(\frac{k}{2}\right) + k$

$\leq 2\left(c\frac{k}{2} \cdot \log \frac{k}{2}\right) + k$

$= ck\left(\log k - \log 2\right) + k$

$= ck \log k - ck + k$

$\leq ck \log k$ ($-ck + k < 0$)

Ex 2: Fibonacci

$T(0) = 1$

$T(1) = 1$

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

$$T(n) = O(2^n) \leq c \cdot 2^n$$

base: $n = 2$

$$T(2) = 2 \leq c \cdot 2^2$$

hipótese: $T(n) \leq c \cdot 2^n$

$n < k$

passo: $n = k$

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

$$\leq c\,2^{k-1} + c\,2^{k-2}$$

$$\leq c\,2^{k-1} + c\,2^{k-1}$$

$$= 2 \cdot c \cdot 2^{k-1}$$

$$= c\,2^k$$

4) Teorema Mestre:

Seja $a \geq 1$, $b > 1$ constantes e $f(n)$ uma função. Seja

$$T(n) = a\,T(n/b) + f(n)$$

Então:

1. Se $f(n) = O(n^{\log_b a - \epsilon})$, $T(n) = \Theta(n^{\log_b a})$

2. Se $f(n) = \Theta(n^{\log_b a})$, $T(n) = \Theta(n^{\log_b a} \log n)$

3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ e $a f(n/b) \leq c f(n)$,

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