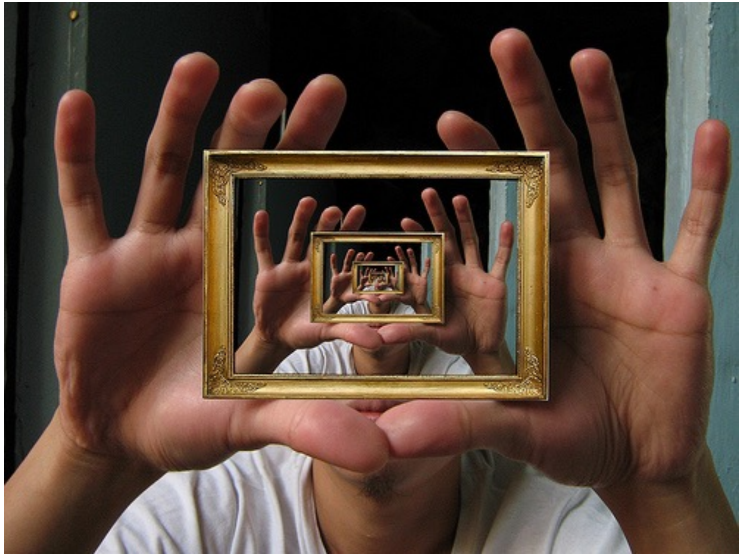
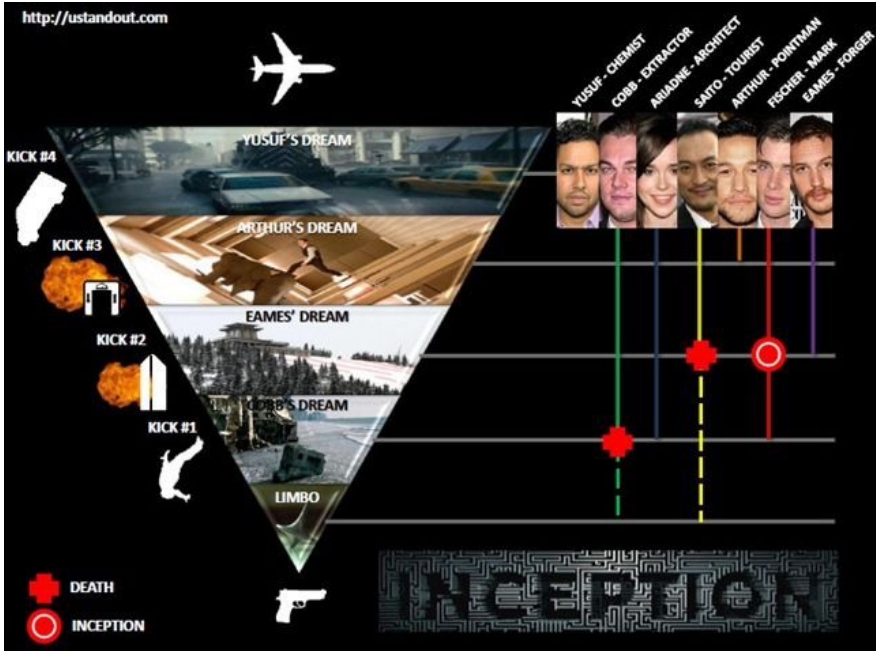
Algoritmos Recursivos & Recorrências

DEC0006 - Prof. Martín Vigil

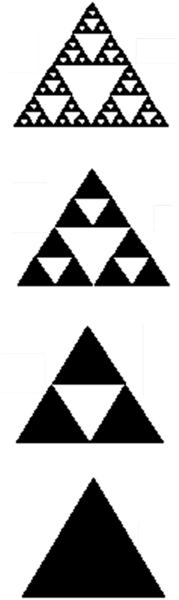
Recursividade: propriedade do que pode ser *repetido*

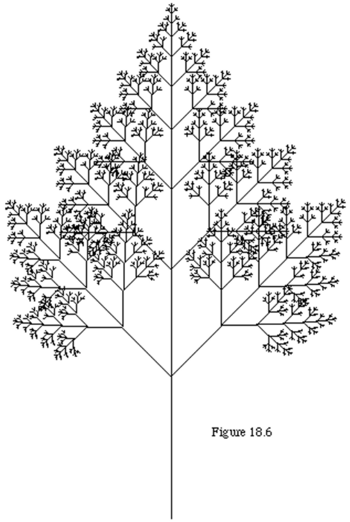




Recursão

- É definir um objeto em termos de outros objetos do *mesmo* tipo
- Os outros objetos são instâncias *mais simples ou menores*





Algoritmos recursivos

- Algoritmos que fazem chamadas a si mesmos

```
função Fib(n)
    se n == 0 OU n == 1
        retorne n
    senão
        retorne Fib(n-1) + Fib(n-2)
```

n	0	1	2	3	4	5	6	7
Fib(n)	0	1	1	2	3	5	8	13

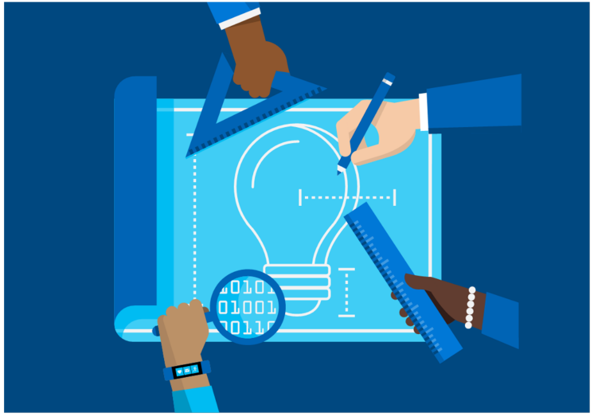
Algoritmos recursivos para calcular fatorial

n! = n*(n-1)!

```
função fatorialRec(n):
    se n == 0 OU n == 1:
        retorne 1
    se n > 1 :
        retorne n * fatorialRec(n-1)
    senão :
        retorne -1;
```

Projetando algoritmos recursivos: passos

1. **Encontrar método geral:** como reduzir o problema a ser resolvido?
2. **Identificar condição de parada:** quando resolver o problema diretamente?
3. **Delinear algoritmo:** qual é sequência instruções envolvendo 1. e 2.
4. **Verificar terminação:** o algoritmo delineado *sempre* termina?
5. **Analisar complexidade:** encontrar os limites da função de complexidade



Projetando algoritmo recursive de Fibonacci

1. **Encontrar método geral:** como *reduzir* o problema a ser resolvido?

$$Fib(n) = Fib(n-1)+Fib(n-2) = Fib(n-2)+Fib(n-3)+Fib(n-2)+Fib(n-1) = ...+Fib(0)+Fib(1)$$

Projetando algoritmos recursivos: passos

2. Identificar condição de parada: quando resolver o problema *diretamente*?

$$Fib(0) = 1$$
$$Fib(1) = 1$$

Projetando algoritmos recursivos: passos

3. Delinear algoritmo: qual é sequência instruções envolvendo 1. e 2.

```
função Fib(n)
    se n == 0 OU n == 1
        retorne n
    senão
        retorne Fib(n-1) + Fib(n-2)
```

Projetando algoritmos recursivos: passos

4. Verificar terminação: o algoritmo delineado *sempre* termina?

```
função Fib(n)
      se n == 0 OU n == 1
            retorne n
      senão
            retorne Fib(n-1) + Fib(n-2)
```

Projetando algoritmos recursivos: passos

5. Analisar complexidade: encontrar os limites da função de complexidade

```
1.  função Fib(n)
2.      se n == 0 OU n == 1
3.          retorne n
4.      senão
5.          retorne Fib(n-1) + Fib(n-2)
```

$$T(n) = \begin{cases} \Theta(1) + T(n-1) + T(n-2), & n > 1 \\ \Theta(1), & n < 2 \end{cases}$$

Recorrências &
Métodos para resolvê-las

Recorrências

- Equação ou inequação que descreve uma função em termos do seu valor para entradas pequenas
- Característica recursiva

$$T(n) = \begin{cases} \Theta(1) + T(n-1) + T(n-2), & n > 1 \\ \Theta(1) & , \ n < 2 \end{cases}$$

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

Como descobrir g(n) tal que T(n) ∈ O(g(n))?

Método da Árvore Recursiva: passos

Seja $T(n) = a * T\left(\frac{n}{b}\right) + f(n)$ uma recorrência onde a,b>0.

1 Desenvolva uma árvore de chamadas recursivas para T(n) do seguinte modo:

1a) Cria uma árvore onde há somente a raiz rotulada T(n)

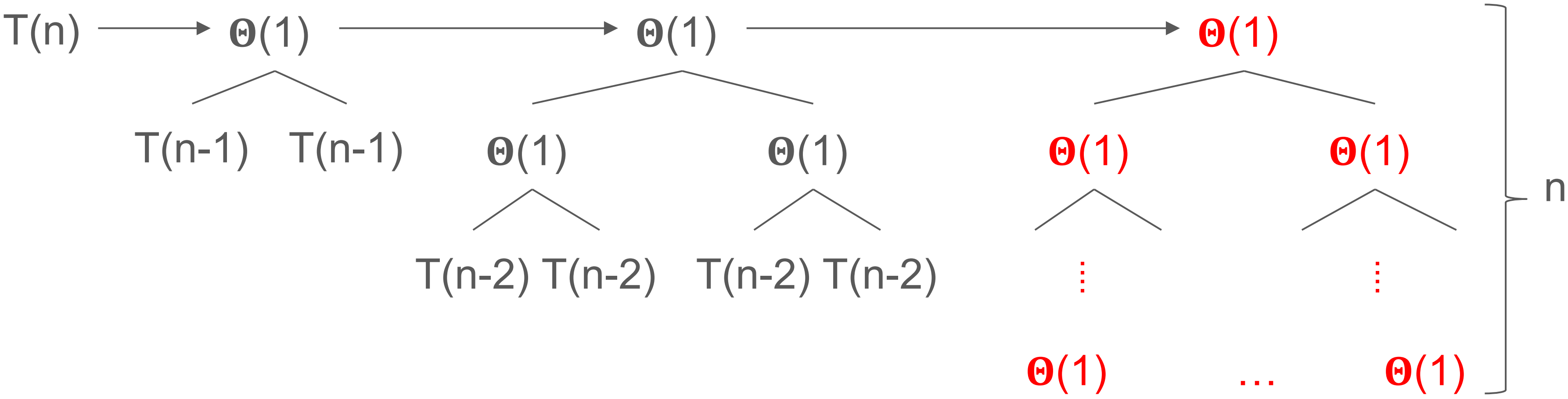
1b) Expanda a raiz, criando uma nova árvore cuja raiz é f(n) e a>0 subárvores com rótulo $T(\frac{n}{b})$

1c) Repita 1b) para cada subárvore até que $\frac{n}{b} = 1$

2 Some os custos de cada nível

Método da Árvore Recursiva: exemplo #1

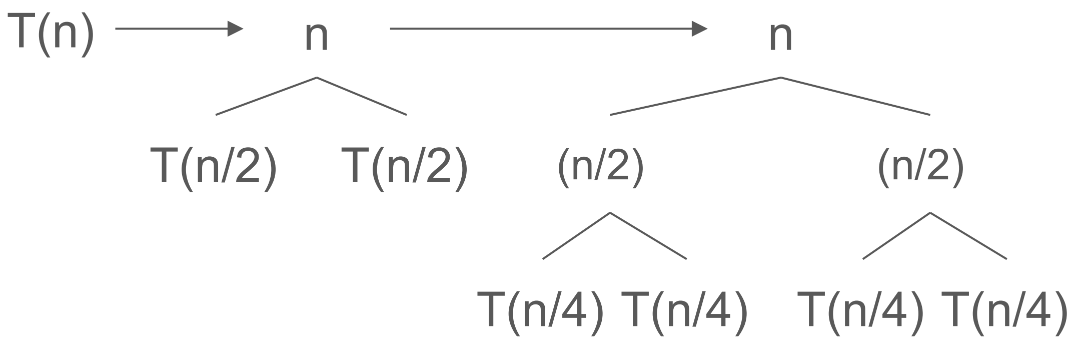
Seja $T(n) = 2T(n-1) + \Theta(1)$ uma recorrência



$T(n) = 2T(n-1) + \theta(1) = 2^n - 1 \in O(2^n)$

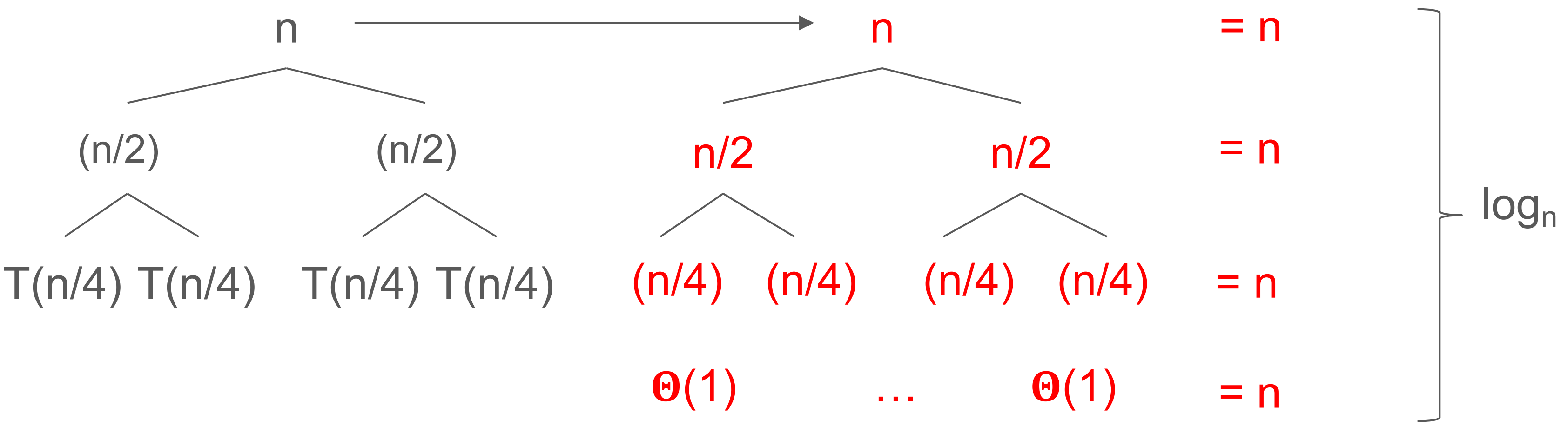
Método da Árvore Recursiva: exemplo #2

Seja $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ uma recorrência, $f(n) = cn \in \Theta(n)$



Método da Árvore Recursiva: exemplo #2

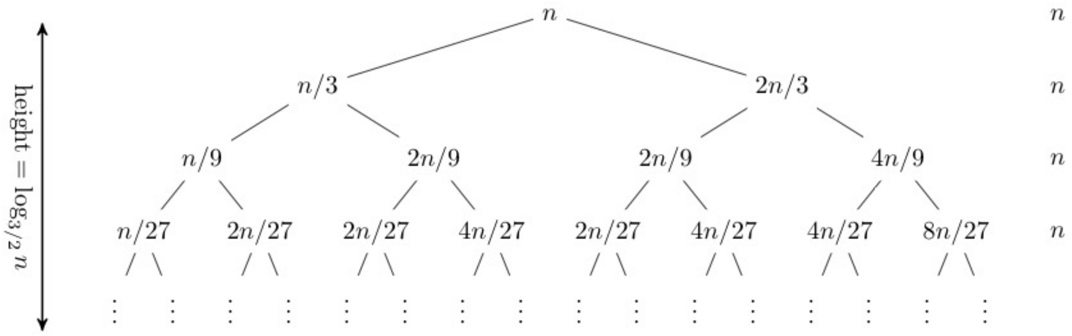
Seja $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ uma recorrência, $f(n) = cn \in \Theta(n)$



$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) = nlog_2n \in O(nlog\ n)$

Método da Árvore Recursiva: complicações #1

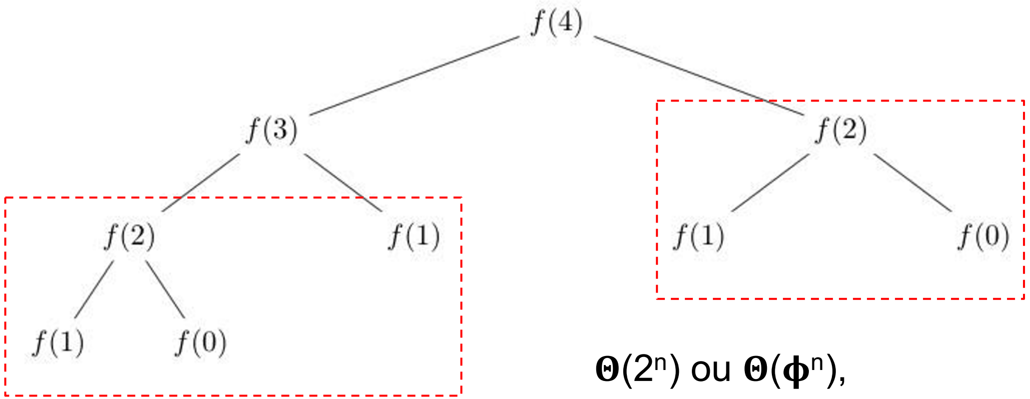
Seja $T(n) = T\left(\frac{n}{3}\right) + T(\frac{2n}{3}) + \Theta(n)$ uma recorrência.



Solução complexa para árvores desbalanceadas

Método da Árvore Recursiva: complicações #2

```
função f(n):
    se n == 0 OU n == 1:
        retorne n
    se n >1 :
        retorne f(n-1) + f(n-2)
    senão :
        retorne -1;
```



$\Theta(2^n)$ ou $\Theta(\phi^n)$,
ϕ=1.61803398

Programação dinâmica: [cap. 15, 3] na Bibliografia

Método do Teorema Mestre

- Seja $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ uma recorrência onde a,b≥1 são constantes e f(n) é uma função assintoticamente positiva
- Compare f(n) a $n^{\log_b a}$ assintoticamente para escolher uma opção:

1. se $f(n) \in O(n^{\log_b a - w})$, onde w>0 é uma constante, então $T(n) \in \Theta\left(n^{\log_b a}\right)$
2. se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta\left(n^{\log_b a}\, log\, n\right)$
3. se $f(n) \in \Omega(n^{\log_b a + w})$, onde w>0 é uma constante, e se $\mathrm{af}\left(\frac{n}{b}\right) \leq cf(n)$ para alguma constante c<1 e todos n suficientemente grandes, então $\mathrm{T(n)} \in \Theta(f(n))$.

Método do Teorema Mestre: exemplo #1

- Descubra o limite assintótico restrito de $T(n) = 9T\left(\frac{n}{3}\right) + n$
- Se $T(n) = aT\left(\frac{n}{b}\right) + f(n),$ a=9, b=3, f(n) = n e $n^{\log_3 9} = n^2$, w=?

Caso1: se $f(n) \in \Theta\left(n^{\log_b a - w}\right)$ e w>0 é constante, então $T(n) \in \Theta\left(n^{\log_3 9}\right) = \Theta(n^2)$

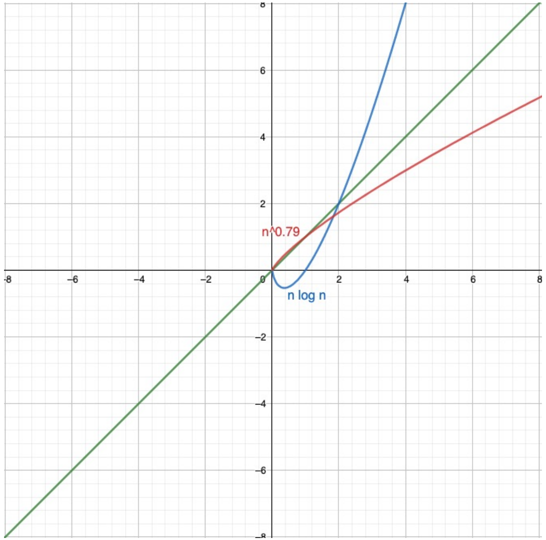
Método do Teorema Mestre: exemplo #2

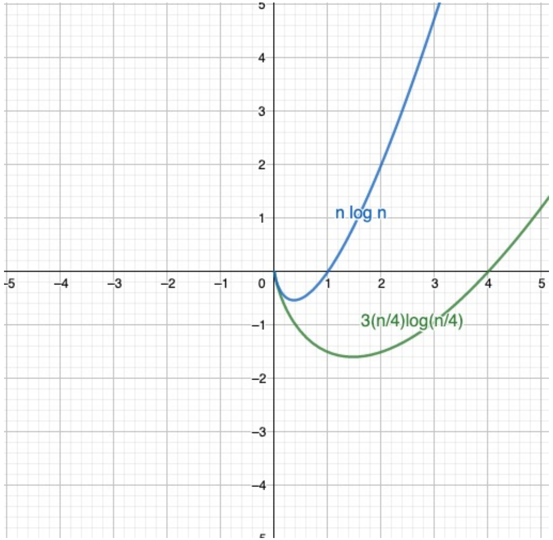
- Descubra o limite assintótico restrito de $T(n) = T\left(\frac{2n}{3}\right) + 1$
- Se $T(n) = aT\left(\frac{n}{b}\right) + f(n),$ a=1, b=3/2, f(n) = 1 e $n^{\log_{3/2} 1} = n^0 = 1$

Caso 2: se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(\mathrm{f(n)logn}) = \Theta\left(n^{\log_b a}\, log\, n\right)$

Método do Teorema Mestre: exemplo #3

- Descubra o limite assintótico restrito de $T(n) = 3T\left(\frac{n}{4}\right) + nlogn$
- Se $T(n) = aT\left(\frac{n}{b}\right) + f(n),$ a=3, b=4, f(n) = nlog n, $n^{\log_4 3} = n^{0.79}$, w=?

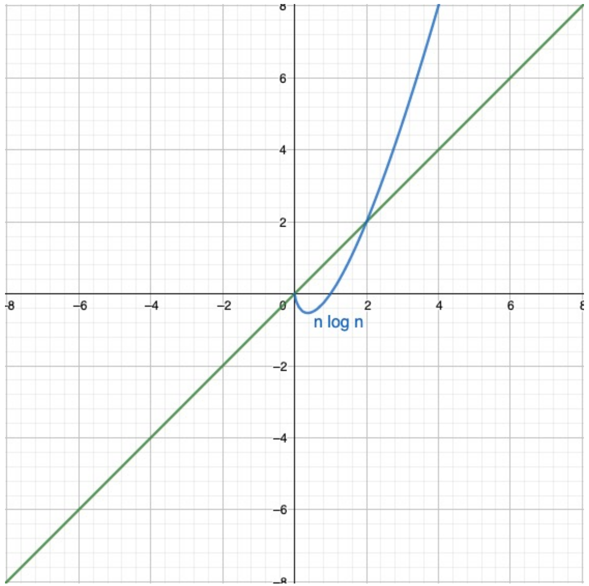




Caso 3: se $f(n) \in \Omega(n^{\log_b a+w})$, w>0 é uma constante, $\mathrm{af}\left(\frac{n}{b}\right) \leq cf(n)$ para alguma constante c<1 e todos n suficientemente grandes, então $\mathrm{T(n)} \in \Theta(f(n))$.

Método do Teorema Mestre: exemplo #4

- Descubra o limite assintótico restrito de $T(n) = 2T\left(\frac{n}{2}\right) + nlogn$
- Se $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, a=2, b=2, f(n) = nlog n, $n^{\log_2 2} = n$, nlogn > n?



Caso 3: se $f(n) \in \Omega(n^{\log_b a+w})$, w>0 é uma constante, af$\left(\frac{n}{b}\right) \leq cf(n)$ para alguma constante c<1 e todos n suficientemente grandes, então $\mathrm{T(n)} \in \Theta(f(n))$.

Síntese da aula

- Recursão: definir um objeto em termos de objetos **menores** do mesmo tipo
- Útil para reduzir a solução de um problema à solução de **subproblemas**
  - O custo de algoritmo recursos é geralmente definido por uma recorrência
  - Método da Árvore Recursiva: contabiliza operações a cada recursão
  - Método do Teorema Mestre: 3 casos cuja escolha é feita com base em f(n)
  - Pode ser não trivial aplicar os métodos em qualquer recorrência.

Bibliografia

[1] COOKE, D. John. **Constructing Correct Software.** 2. ed. Londres: Springer-verlag, 2005. 509 p.

[2] KRUSE, Robert L.; RYBA, Alexander J.. **Data Structures and Program Design in C++.** Upper Saddle River, New Jersey: Prentice-hall, 2000. 717 p.

[3] CORMEN, Thomas H. et al. **Introduction to Algorithms.** 3. ed. [S.I.]: Mit Press, 2009. 1312 p.

[4] GOODRICH, Michael T.; TAMASSIA, Roberto. **Data structures and algorithms in Java.** 5. ed. [S. I.]: Wiley, 2010. 714 p.