Aulas 1 & 2

1. Calcular os nºs de Fibonacci
2. Notação Assimptótica
3. Dividir-para-Conquistar
4. Merge Sort
5. Teorema Mestre

Exemplo 1 - N^{os} de Fibonacci

$$fib(n) = \begin{cases} n & \text{se } n = 0 \text{ ou } n = 1 \\ fib(n-1) + fib(n-2) & \text{c.c.} \end{cases}$$

Implementação 3

```
Fib(n)
    if n == 0 || n == 1
        return 1
    else
        fib-cur := 1
        fib-prev := 0
        for i := 2 to n
            temp := fib-cur
            fib-cur := fib-prev + fib-cur
            fib-prev :=
        return cur
```

Esta implementação é eficiente ?

$T(n) = O(n)$

$S(n) = O(1)$

Invariante:

Notação Assimptótica

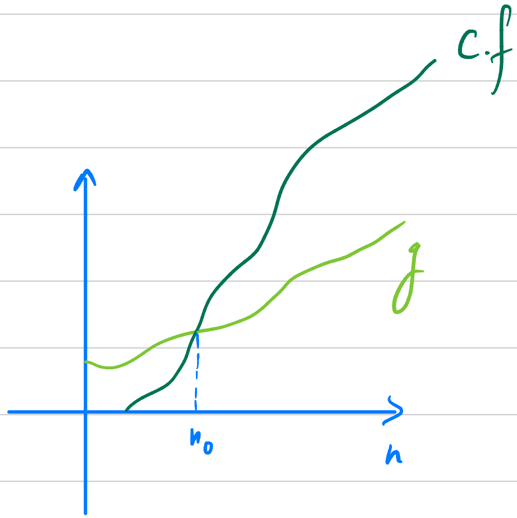
Definição 1 [Majorante / Minorante Assimptótica]

Majorante:

$g \in O(f)$ sse $\exists c.\ \exists n_0.\ \forall n \geq n_0.\ g(n) \leq c.f(n)$

Notação Assimptótica

Definição 1 [Majorante / Minorante Assimptótica]

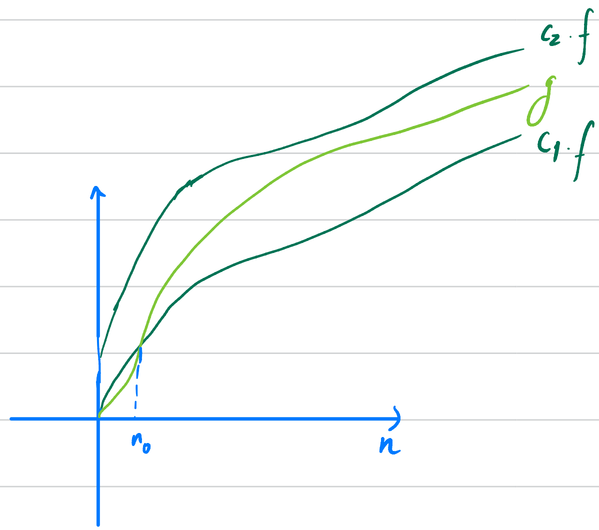


Majorante:

$$g \in O(f) \quad \text{sse} \quad \exists c . \exists n_0 . \forall n \geq n_0 . \; g(n) \leq c . f(n)$$

Minorante:

$$g \in \Omega(f) \quad \text{sse} \quad \exists c . \exists n_0 . \forall n \geq n_0 . \; g(n) \geq c . f(n)$$



Tight-Bound:

$$g \in \Theta(f) \quad \text{sse} \quad \exists c_1, c_2 . \exists n_0 . \forall n \geq n_0 . \; c_1 . f(n) \leq g(n) \leq c_2 . f(n)$$

Notação Assimptótica

Lema 1

$g \in \Theta(f)$ sse $g \in O(f) \wedge g \in \Omega(f)$

Prova

$\boxed{\Rightarrow}$

Notação Assimptótica

Lema 1

$g \in \Theta(f)$ sse $g \in O(f) \wedge g \in \Omega(f)$

Prova

$\boxed{\Rightarrow}$

$g \in \Theta(f) \Rightarrow g \in O(f) \wedge g \in \Omega(f)$

- $g \in \Theta(f)$ (hyp)
- $\exists n_0, c_1, c_2 . \forall n \geq n_0 . c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$

$\hookrightarrow \exists n_0, c . \forall n \geq n_0 . g(n) \geq c \cdot f(n) \iff g \in \Omega(f)$

$\hookrightarrow \exists n_0, c . \forall n \geq n_0 . g(n) \leq c \cdot f(n) \iff g \in O(n)$

Notação Assimptótica

Lema 1

$g \in \Theta(f)$ sse $g \in O(f) \wedge g \in \Omega(f)$

Prova

$\boxed{\Leftarrow}$

$g \in O(f) \wedge g \in \Omega(f) \Rightarrow g \in \Theta(f)$

- $g \in O(f) \Leftrightarrow \exists c_1, n_0 .\ \forall n \geq n_0 .\ g(n) \leq c_1 \cdot f(n)$
- $g \in \Omega(f) \Leftrightarrow \exists c_2, n_0' .\ \forall n \geq n_0' .\ g(n) \geq c_2 \cdot f(n)$
- $\exists n_0'', c_1, c_2 .\ \forall n \geq n_0'' .\ c_2 \cdot f(n) \leq g(n) \leq c_1 \cdot f(n)$

$\hookrightarrow \max(n_0, n_0')$

Notação Assimptótica

Lema 1 [Transitividade]

i) $f \in O(g) \wedge g \in O(h) \Rightarrow f \in O(h)$

ii) $f \in \Omega(g) \wedge g \in \Omega(h) \Rightarrow f \in \Omega(h)$

iii) $f \in \Theta(g) \wedge g \in \Theta(h) \Rightarrow f \in \Theta(h)$

Prova i)

Notação Assimptótica

Lema 1 [Transitividade]

i) $f \in O(g) \wedge g \in O(h) \Rightarrow f \in O(h)$

ii) $f \in \Omega(g) \wedge g \in \Omega(h) \Rightarrow f \in \Omega(h)$

iii) $f \in \Theta(g) \wedge g \in \Theta(h) \Rightarrow f \in \Theta(h)$

Prova i)

- Hipóteses: $\underbrace{f \in O(g)}_{(*)} \wedge \underbrace{g \in O(h)}_{(**)}$

$(*)$ $\exists c, n_0.\ \forall n \geq n_0.\ f(n) \leq c \cdot g(n)$

$(**)$ $\exists c', n_0'.\ \forall n \geq n_0'.\ g(n) \leq c' \cdot h(n)$

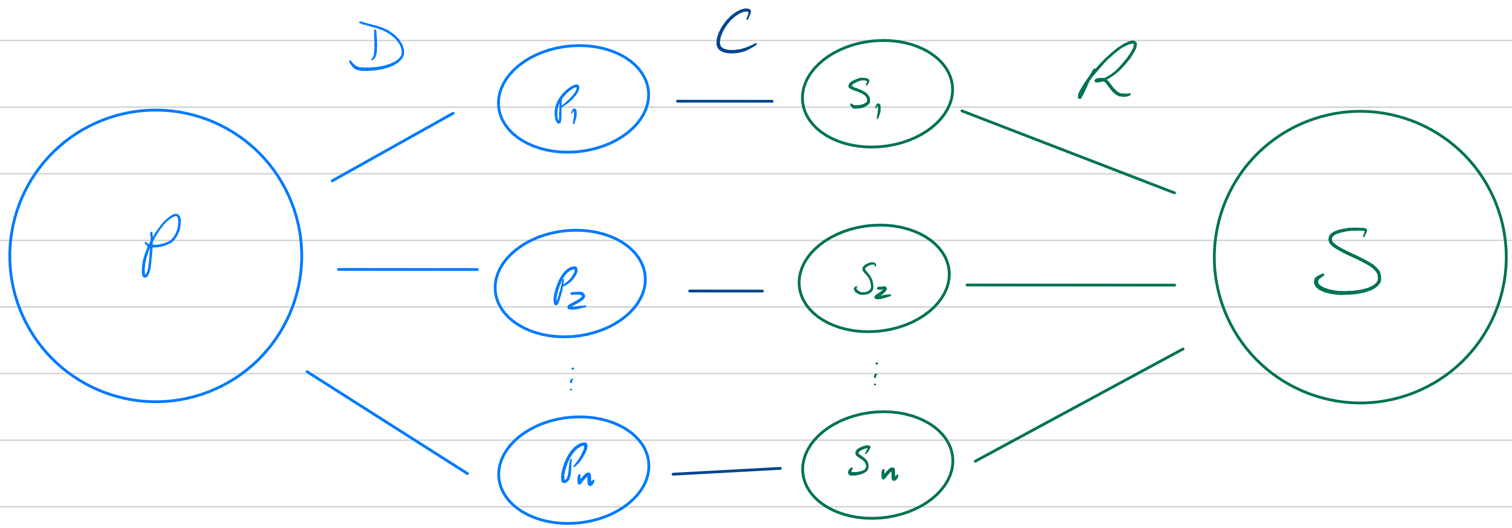
$\forall n \geq \max(n_0, n_0').\ f(n) \leq c \cdot c' \cdot h(n) \rightarrow f \in O(h)$

Metodologia Dividir-para-Conquistar

1. Dividir o problema a resolver num conjunto de subproblemas

2. Resolver (recursivamente) cada um dos subproblemas

3. Combinar as soluções dos subproblemas para obter a solução do problema original



Merge Sort

```
MergeSort(A, l, r)
  if l < r
     m = ⌊(l+r)/2⌋
     MergeSort(A, l, m)
     MergeSort(A, m+1, r)
     Merge(A, l, m, r)
```

Dividir: $D(n) = O(1)$

Resolver: $R(n) = 2 \cdot T(n/2)$

Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Exemplo:

$\langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$ $\quad l = 1, \ r = 8$

MergeSort

```
MergeSort(A, l, r)
  if l < r
     m = ⌊(l+r)/2⌋
     MergeSort(A, l, m)
     MergeSort(A, m+1, r)
     Merge(A, l, m, r)
```

Dividir: $D(n) = O(1)$

Resolver: $R(n) = 2 \cdot T(n/2)$

Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

⟨3, 41, 52, 26, 38, 57, 9, 49⟩ $l = 1, \ r = 8$

⟨3, 9, 26, 38, 41, 49, 52, 57⟩

⟨3, 41, 52, 26⟩ ⟨38, 57, 9, 49⟩

⟨3, 26, 41, 52⟩ ⟨9, 38, 49, 57⟩

⟨3, 41⟩ ⟨52, 26⟩ ⟨38, 57⟩ ⟨9, 49⟩

⟨3, 41⟩ ⟨26, 52⟩ ⟨38, 57⟩ ⟨9, 49⟩

⟨3⟩ ⟨41⟩ ⟨52⟩ ⟨26⟩ ⟨38⟩ ⟨57⟩ ⟨9⟩ ⟨49⟩

Merge Sort

MergeSort (A, l, r)
  if l < r
    $m = \lfloor l + r / 2 \rfloor$
    MergeSort (A, l, m)
    MergeSort (A, m+1, r)
    Merge (A, l, m, r)

Dividir: $D(n) = O(1)$
Resolver: $R(n) = 2 \cdot T(n/2)$
Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Merge (A, l, m, r)

Setup

Main Loop

Merge (A, l, m, r)

Setup
Main Loop

Setup:

	l			m				r
A:	1	3	5	6	2	4	9	10

L:	1	3	5	6	∞
R:	2	4	9	10	∞

Main Loop:

	i				
L:	1	3	5	6	∞

	j				
R:	2	4	9	10	∞

A:	1	3	5	6	2	4	9	10

Merge (A, l, m, r)

$\boxed{\text{Setup}}$

$\boxed{\text{Main Loop}}$

Setup:

	l			m				r
A:	1	3	5	6	2	4	9	10

L:	1	3	5	6	∞
R:	2	4	9	10	∞

Main Loop:

L:	1	3	5	6	∞
R:	2	4	9	10	∞

A:	1	~~3~~	~~5~~	~~6~~	~~2~~	~~4~~	9	10
	1	2	3	4	5	6	9	10

MergeSort

MergeSort(A, l, R)
  if l < R
    m = $\lfloor l + R / 2 \rfloor$
    MergeSort(A, l, m)
    MergeSort(A, m+1, R)
    Merge(A, l, m, R)

Dividir: $D(n) = O(1)$
Resolver: $R(n) = 2 \cdot T(n/2)$
Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Merge(A, l, m, R)

$\boxed{\text{Setup}}$

$\boxed{\text{Main Loop}}$

Merge Sort

```
MergeSort(A, l, r)
  if l < r
    m = ⌊l+r/2⌋
    MergeSort(A, l, m)
    MergeSort(A, m+1, r)
    Merge(A, l, m, r)
```

Dividir: $D(n) = O(1)$

Resolver: $R(n) = 2 \cdot T(n/2)$

Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Merge(A, l, m, r)

Merge(A, l, m, r)

Setup

Main Loop

Merge Sort

```
MergeSort(A, l, r)
  if l < r
    m = ⌊l+r/2⌋
    MergeSort(A, l, m)
    MergeSort(A, m+1, r)
    Merge(A, l, m, r)
```

Dividir: $D(n) = O(1)$

Resolver: $R(n) = 2 \cdot T(n/2)$

Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Merge(A, l, m, r)

Merge(A, l, m, r)

Setup

Main Loop

Merge Sort

MergeSort(A, l, r)
  if l < r
    m = ⌊l + r / 2⌋
    MergeSort(A, l, m)
    MergeSort(A, m+1, r)
    Merge(A, l, m, r)

Dividir: $D(n) = O(1)$
Resolver: $R(n) = 2 \cdot T(n/2)$
Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Merge(A, l, m, r)

Setup

Main Loop

Set Up: Alocar os arrays L e R
e preencher c/ os valores de A

let L[1 .. (m - l) + 2] be a new array
let R[1 .. (r - m) + 1] be a new array
for i = 1 to (m - l) + 1
  L[i] = A[l + i - 1]
for j = 1 to (r - m)
  R[j] = A[l + j]
L[m - l + 2] = ∞
R[m - l + 2] = ∞

Merge Sort

```
MergeSort(A, l, r)
    if l < r
        m = ⌊l + r / 2⌋
        MergeSort(A, l, m)
        MergeSort(A, m+1, r)
        Merge(A, l, m, r)
```

Merge(A, l, m, r)

Setup

Main Loop

Dividir: $D(n) = O(1)$

Resolver: $R(n) = 2 \cdot T(n/2)$

Combinar: $C(n) = \underbrace{?}_{\text{merge}}$

Set Up: Main Loop

```
i = 1; j = 1

for k = l to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i++
    else
        A[k] = R[j]
        j++
```

Merge - Complexidade

Merge (A, l, m, r)

```
let L[1..(m-l)+2] be a new array
let R[1..(r-m)+1] be a new array
for i=1 to (m-l)+1
    L[i] = A[l+i-1]
for j=1 to (r-m)
    R[j] = A[m+j]
L[m-l+2] = ∞
R[m-l+2] = ∞

i = 1; j = 1
for k=l to r
    if L[i] ≤ R[j]
        A[k] = L[i]; i++
    else
        A[k] = R[j]; j++
```

Merge Sort

TPC: $\underbrace{O(n) + O(n)}$ $= O(n)$

$f \in O(n) \land g \in O(n) \Rightarrow f+g \in O(n)$

Merge (A, l, m, r)

let L[1.. (m-l)+2] be a new array
let R[1.. (r-m)+1] be a new array
for i = 1 to (m-l)+1
  L[i] = A[l+i-1]
for j = 1 to (r-m)
  R[j] = A[l+j]
L[m-l+2] = ∞
R[m-l+2] = ∞

$\rightarrow (m-l)+1$

$\rightarrow (r-m)$

$\underbrace{r - \cancel{m} + \cancel{m} - l}\ +1$

$\underbrace{(r-l)}_{n}+1$

$O(n)$

i = 1; j = 1
for k = l to r
  if L[i] ≤ R[j]
    A[k] = L[i]; i++
  else
    A[k] = R[j]; j++

$\underbrace{(r-l)+1}_{n}$

$O(n)$

$O(n)$

MergeSort

```
MergeSort(A, l, r)
  if l < r
    m = ⌊l + r / 2⌋
    MergeSort(A, l, m)
    MergeSort(A, m+1, r)
    Merge(A, l, m, r)
```

Dividir: $D(n) = O(1)$

Resolver: $R(n) = 2 \cdot T(n/2)$

Combinar: $C(n) = O(n)$

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

0: $O(n)$ —— $1 \times O(n/2^0) = O(n)$

1: $O(n/2)$ $O(n/2)$ —— $2 \times O(n/2^1) = O(n)$

2: $O(n/4)$ $O(n/4)$ $O(n/4)$ $O(n/4)$ —— $2^2 \times O(n/2^2) = O(n)$

k: $2^k \times O(n/2^k) = O(n)$

Quem é k?

$$n/2^k = 1$$

$$k = \log_2 n$$

$$\underline{T(n) = O(n \cdot \log n)}$$

Método de Substituição

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

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

Base Case

$\boxed{n=2}$

$$T(2) = 2\,T(2/2) + O(2)$$
$$= 2 \cdot T(1) + O(1)$$
$$= O(1)$$

$\boxed{n>2}$

$$T(n+1) = 2 \cdot T\left(\frac{n+1}{2}\right) + O(n+1)$$

$$= 2 \cdot O\left(\left(\frac{n+1}{2}\right) \cdot \log_2\left(\frac{n+1}{2}\right)\right) + O(n+1)$$

$$= \cancel{2} \cdot \frac{n+1}{\cancel{2}} \cdot O\left(\log_2\left(\frac{n+1}{2}\right)\right) + O(n+1)$$

$$= (n+1) \cdot O(\log_2(n+1)) \cancel{- O((n+1) \cdot \log_2 2)} \cancel{+ O(n+1)}$$

$$= (n+1) \cdot O(\log_2(n+1))$$

Teorema Mestre (Simplificado)

Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

então:

$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a \\ O(n^d \log n) & \text{se } d = \log_b a \\ O(n^{\log_b a}) & \text{se } d < \log_b a \end{cases}$$

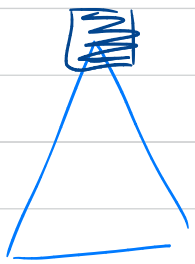
Teorema Mestre (Simplificado)

Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

então:

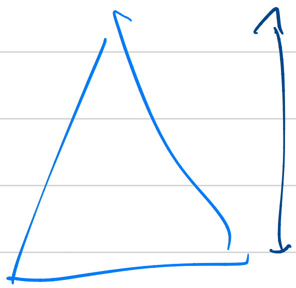
$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a & \text{I} \\ O(n^d \log n) & \text{se } d = \log_b a & \text{II} \\ O(n^{\log_b a}) & \text{se } d < \log_b a & \text{III} \end{cases}$$

I



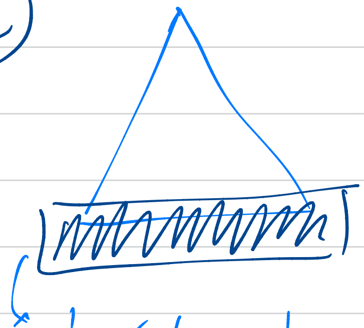
Custo da raiz domina o custo do problema

II



Custo distribuído

III



custo é dominado pelo nº de folhas

Teorema Mestre (Simplificado)

Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

então:

$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a & \text{I} \\ O(n^d \log n) & \text{se } d = \log_b a & \text{II} \\ O(n^{\log_b a}) & \text{se } d < \log_b a & \text{III} \end{cases}$$

Merge Sort

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

$a : 2$

$b : 2$

$d : 2$

$\log_b a = \log_2 2 = 1 = d$

Caso II $\rightarrow$ $T \in O(n \log n)$

Teorema Mestre Generalizado

Se $T(n) = a\,T(n/b) + f(n)$ p/ constantes $a \geq 1$ e $b > 1$
então:

(I) Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ p/ algum $\varepsilon > 0$, e se $a\,f(n/b) \leq c.f(n)$ p/ algum $c < 1$ e n soficientemente grande,

Condição de Regularidade

então: $T(n) = \Theta(f(n))$

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

(III) Se $f(n) = O(n^{\log_b a - \varepsilon})$ p/ algum $\varepsilon > 0$, então: $T(n) = \Theta(n^{\log_b a})$

Teorema Mestre - Exemplos

1. $T(n) = 9\,T(n/3) + n$

- Simplificado:

- Generalizado:

Teorema Mestre - Exemplos

1. $T(n) = 9\,T(n/3) + n$

- Simplificado: $a : 9$, $b : 3$, $d : 1$ $\rightarrow \log_b a = 2 > 1 \Rightarrow T(n) = O(n^2)$

- Generalizado: $f(n) = n \rightarrow$ Relação entre $f(n)$ e $n^{\log_b a} = n^2$

$$f(n) \in O(n^{2-\varepsilon})$$
$$\Downarrow$$
$$\underline{T(n) = \Theta(n^2)}$$

Teorema Mestre - Exemplos

1. $T(n) = 3T(n/4) + n \log n$

- Simplificado:

- Generalizado:

Teorema Mestre - Exemplos

1. $T(n) = 3T(n/4) + n \log n$

- Simplificado:

- Generalizado:

$f(n) = n \log n \qquad n^{\log_b a} = n^{\log_4 3} \qquad \log_4 3 < 1$

$n \log n > n^{\log_4 3}$

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

Condição de regularidade: //

$a\, f(n/b) \leq c \cdot f(n)$

$3 \frac{n}{4} \log(n/4) \leq c \cdot n \log n$

Seja $c = 3/4$

$3/4\, \cancel{n} \log(n/4) \leq 3/4\, \cancel{n} \log n$

$\log(n/4) \leq \log(n)$

$\log n - \log(n/4) \geq 0$

$\log\left(\frac{n}{n/4}\right) \geq 0$

$\log 4 \geq 0$ T

Teorema Mestre - Exemplo Código

I

```
int f(int n)
{
  int j, i;

  j = 0;
  i = 0;
  while(i < n)
  {
    j++;
    i+= 2;
  }

  if(n > 1)
    i = 2*f(j) + f(j);

  return i;
}
```

Teorema Mestre - Exemplo Código

I

```
int f(int n)
{
  int j, i;

  j = 0;
  i = 0;
  while(i < n)
  {
    j++;
    i+= 2;
  }

  if(n > 1)
    i = 2*f(j) + f(j);

  return i;
}
```

$O(n)$

k	i	j
0	0	0
1	2	1
2	4	2
3	6	3
⋮		
k	2k	k

Condição Limite: $i = n$

$2k = n \Leftrightarrow k = n/2$

- $T(n) = 2 \cdot T(n/2) + O(n)$

$a = 2 \quad b = 2 \quad d = 1$

$T(n) = O(n \log n)$

Teorema Mestre - Exemplo Código

II

```
int f(int n)
{
  int i = 0;
  while(i*i < n)
    i++;
  if(n > 1)
    i = f(n/4) + f(n/4) + f(n/4);

  return i;
}
```

Teorema Mestre - Exemplo Código

II

```
int f(int n)
{
  int i = 0;
  while(i*i < n)
    i++;
  if(n > 1)
    i = f(n/4) + f(n/4) + f(n/4);

  return i;
}
```

k	i
0	0
1	1
2	2
$\vdots$	$\vdots$
k	k

Condição de paragem:

$i * i = n$

$k^2 = n \Leftrightarrow$

$k = \sqrt{n}$

- $T(n) = 3\,T(n/4) + O(\sqrt{n})$

$a = 3 \quad b = 4 \quad d = 1/2$

$T(n) = O(n^{\log_4 3})$

$4^{1/2} = 2 < 3 \Rightarrow 1/2 < \log_4 3$

Teorema Mestre - Exemplo Código

III

```
int f(int n)
{
  int i = 0, j = 0;
  while(n*n > i) {
    i = i + 2;
    j++;
  }

  if(n > 1)
    i = 5*f(n/2) + f(n/2) + f(n/2) + f(n/2);

  while (j > 0) {
    i = i + 2;
    j--;
  }
  return i;
}
```

Teorema Mestre - Exemplo Código

III

```
int f(int n)
{
  int i = 0, j = 0;
  while(n*n > i) {
    i = i + 2;
    j++;
  }

  if(n > 1)
    i = 5*f(n/2) + f(n/2) + f(n/2) + f(n/2);

  while (j > 0) {
    i = i + 2;
    j--;
  }
  return i;
}
```

$O(n^2)$

$O(n^2)$

k	i	j
0	0	0
1	2	1
2	4	2
3	6	3
⋮		
k	2k	k

Condição de Paragem:

$i = n^2$

$2k = n^2$

$k = n^2/2$

$$T(n) = 4 \cdot T(n/2) + O(n^2)$$

$a = 4 \quad b = 2 \quad d = 2$

$$T(n) = O(n^2 \cdot \log n)$$

Teorema Mestre - Exemplo Código

TV

```
int f(int n)
{
  int i = 0, j = 0;
  while (j < 10) {
    i = i + 2;
    j++;
  }

  if(n > 1)
    i += f(n/2) + 3*f(n/2);

  while (j > 0) {
    i--;
    j = j - 2;
  }
  return i;
}
```

Teorema Mestre - Exemplo Código

TV

```
int f(int n)
{
  int i = 0, j = 0;
  while (j < 10) {
    i = i + 2;
    j++;
  }

  if(n > 1)
    i += f(n/2) + 3*f(n/2);

  while (j > 0) {
    i--;
    j = j - 2;
  }
  return i;
}
```

$O(1)$

$O(1)$

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

$$T(n) = O(n)$$

Teorema Mestre - Exemplo Código

V

```
int f(int n) {
  int i = 0, j = n;

  if (n <= 1) return 1;

  while(j > 0) {
    i++;
    j = j / 2;
  }

  for (int k = 0; k < 4; k++)
    j += f(n/2);

  while (i > 0) {
    j = j + 2;
    i--;
  }
  return j;
}
```

Teorema Mestre - Exemplo Código

V

```
int f(int n) {
  int i = 0, j = n;

  if (n <= 1) return 1;

  while(j > 0) {
    i++;
    j = j / 2;
  }

  for (int k = 0; k < 4; k++)
    j += f(n/2);

  while (i > 0) {
    j = j + 2;
    i--;
  }
  return j;
}
```

$O(\log n)$

$O(\log n)$

k	i	j
0	0	n
1	1	$n/2$
2	2	$n/4$
⋮		
k	k	$n/2^k$

Cond. de paragem:

$$n/2^k = 1 \iff n = 2^k \iff k = \log_2 n$$

$$T(n) = 4 \cdot T(n/2) + O(\log n)$$

$$\log(n) \in O(n^{2-\varepsilon})$$

$$T(n) = O(n^2)$$