

ANÁLISE DE ALGORITMOS RECURSIVOS

MO417 - Complexidade de
Algoritmos I

Santiago Valdés Ravelo
[https://ic.unicamp.br/~santiago/
ravelo@unicamp.br](https://ic.unicamp.br/~santiago/ravelo@unicamp.br)

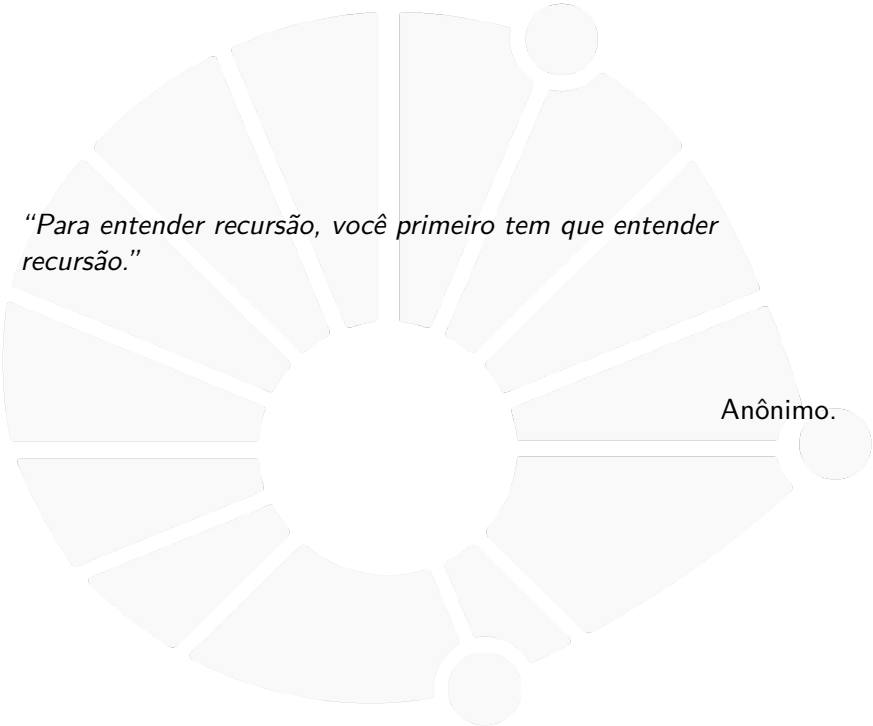
03/24

5



UNICAMP





“Para entender recursão, você primeiro tem que entender recursão.”

Anônimo.



PROJETANDO ALGORITMOS



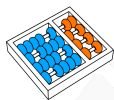
Entendendo e melhorando

Até agora:

- ▶ Ordenamos **incrementalmente** com o INSERTION-SORT.
- ▶ Vimos que sua complexidade de pior caso é $\Theta(n^2)$.

Vamos estudar uma maneira alternativa de ordenar números:

- ▶ Utilizaremos uma técnica recursiva chamada de **divisão e conquista**.
- ▶ Muitas vezes, obtemos algoritmos mais rápidos do que os incrementais.



Algoritmos recursivos

- ▶ Um **algoritmo recursivo** resolve um problema:
 - ▶ Diretamente, se a instância for pequena.
 - ▶ Chamando a si mesmo uma ou mais vezes, se a instância não for pequena.
- ▶ As chamadas recursivas devem receber **instâncias menores**.



Divisão e conquista

Um algoritmo de **divisão e conquista** tem três etapas:

1. **DIVISÃO:** Dividir o problema em subproblemas semelhantes, mas com instâncias menores.
2. **CONQUISTA:** Cada subproblema é resolvido recursivamente, ou diretamente se os subproblemas forem pequenos.
3. **COMBINAÇÃO:** As soluções dos subproblemas são combinadas para obter uma solução da instância original.



Exemplo: ordenando usando divisão e conquista

MERGESORT é um exemplo clássico de divisão e conquista.

Ideia:

1. **DIVISÃO:** Divida um vetor de tamanho n em dois subvetores de tamanhos $\lfloor n/2 \rfloor$ e $\lceil n/2 \rceil$.
2. **CONQUISTA:** Ordene os dois subvetores recursivamente.
3. **COMBINAÇÃO:** Intercale os dois subvetores obtendo um vetor ordenado.

Vejamos um exemplo:





Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: MERGESORT(A, p, r)

```

1 se  $p < r$ 
2    $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3   MERGESORT( $A, p, q$ )
4   MERGESORT( $A, q + 1, r$ )
5   INTERCALA( $A, p, q, r$ )

```

A	p	q	r						
	66	33	55	44	99	11	77	22	88



Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: MERGESORT(A, p, r)

```

1 se  $p < r$ 
2    $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3   MERGESORT( $A, p, q$ )
4   MERGESORT( $A, q + 1, r$ )
5   INTERCALA( $A, p, q, r$ )

```

	p				q				r
A	33	44	55	66	99	11	77	22	88



Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: MERGESORT(A, p, r)

```

1 se  $p < r$ 
2    $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3   MERGESORT( $A, p, q$ )
4   MERGESORT( $A, q + 1, r$ )
5   INTERCALA( $A, p, q, r$ )

```

	p				q			r	
A	33	44	55	66	99	11	22	77	88



Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: MERGESORT(A, p, r)

```

1 se  $p < r$ 
2    $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3   MERGESORT( $A, p, q$ )
4   MERGESORT( $A, q + 1, r$ )
5   INTERCALA( $A, p, q, r$ )
    
```

A	p	11	22	33	44	q	55	66	77	88	99	r
-----	-----	----	----	----	----	-----	----	----	----	----	----	-----



Combinando soluções dos subproblemas

Problema (Intercalar dois subvetores)

- ▶ **Entrada:** Um vetor $A[p \dots r]$ tal que os subvetores $A[p \dots q]$ e $A[q + 1 \dots r]$ estão ordenados.
- ▶ **Saída:** Um rearranjo de $A[p \dots r]$ ordenado.

Entrada:

	p		q		r				
A	22	33	55	77	99	11	44	66	88

Saída:

	p		q		r				
A	11	22	33	44	55	66	77	88	99



Pseudocódigo de INTERCALA

Algoritmo: INTERCALA(A, p, q, r)

```

1  para  $i \leftarrow p$  até  $q$ 
2  |    $B[i] \leftarrow A[i]$ 
3  para  $j \leftarrow q + 1$  até  $r$ 
4  |    $B[r + q + 1 - j] \leftarrow A[j]$ 
5   $i \leftarrow p$ 
6   $j \leftarrow r$ 
7  para  $k \leftarrow p$  até  $r$ 
8  |   se  $B[i] \leq B[j]$ 
9  |   |    $A[k] \leftarrow B[i]$ 
10  |   |    $i \leftarrow i + 1$ 
11  |   senão
12  |   |    $A[k] \leftarrow B[j]$ 
13  |   |    $j \leftarrow j - 1$ 

```



Complexidade de INTERCALA

Entrada:

	p				q				r
A	22	33	55	77	99	11	44	66	88

Saída:

	p				q				r
A	11	22	33	44	55	66	77	88	99

Tamanho da entrada: $n = r - p + 1$.

Consumo de tempo: $\Theta(n)$.



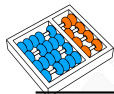
Complexidade de MERGESORT

Algoritmo: MERGESORT(A, p, r)

```

1 se  $p < r$ 
2    $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
3   MERGESORT( $A, p, q$ )
4   MERGESORT( $A, q + 1, r$ )
5   INTERCALA( $A, p, q, r$ )
    
```

- ▶ Tamanho da entrada: $n = r - p + 1$.
- ▶ Seja $T(n)$ o número de instruções executadas no pior caso.



Complexidade de MERGESORT

Algoritmo: MERGESORT(A, p, r)

```

1 se  $p < r$ 
2    $q \leftarrow \lfloor (p+r)/2 \rfloor$ 
3   MERGESORT( $A, p, q$ )
4   MERGESORT( $A, q+1, r$ )
5   INTERCALA( $A, p, q, r$ )
    
```

Linha	Tempo
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) + \Theta(2).$$



Recorrência

O tempo de MERGESORT é dado pela fórmula

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{se } n = 2, 3, 4, \dots \end{cases}$$

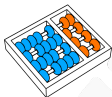
Obtemos uma **fórmula de recorrência**:

- ▶ É a descrição de uma função em termos de si mesma.
- ▶ O tempo de um algoritmo recursivo costuma se descrito por uma recorrência.

Mas queremos uma **fórmula fechada!**



RECORRÊNCIAS



Resolução de recorrências

Considere a recorrência:

$$T(n) = \begin{cases} d & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b & \text{se } n \geq 2, \end{cases}$$

- ▶ Queremos uma **FÓRMULA FECHADA** para $T(n)$.
- ▶ Não é necessária a solução exata.
- ▶ Basta encontrar uma função $f(n)$ tal que $T(n) \in \Theta(f(n))$.



Resolução de recorrências

Existem alguns métodos comuns para resolver recorrências:

- ▶ Substituição.
- ▶ Iteração.
- ▶ Árvore de recorrência.

Veremos também o chamado **Teorema Master**:

- ▶ Aplicável a uma família comum de recorrências.
- ▶ Fornece uma fórmula fechada diretamente.



Método da substituição

Ideia:

1. **ADIVINHAR** uma solução.
2. Demonstrar que é válida usando indução.

Nem sempre é fácil chutar a solução:

- ▶ É necessário ter experiência.
- ▶ Mas vamos obter sugestões com os métodos estudados.



Exemplo

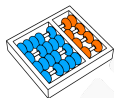
Exemplo

Encontre uma fórmula fechada para:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n & \text{se } n \geq 2 \end{cases}$$

Chutamos que $T(n) \in O(n \log_2 n)$.

- ▶ Observe que há uma constante escondida na notação O .
- ▶ Para usar indução, precisamos conhecer essa constante.
- ▶ Então chutamos que $T(n) \leq 3n \log_2 n$.



Exemplo: passo da indução

Suponha que a desigualdade vale para $k < n$.

$$\begin{aligned}
 T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\
 &\leq 3 \left\lceil \frac{n}{2} \right\rceil \log_2 \left\lceil \frac{n}{2} \right\rceil + 3 \left\lfloor \frac{n}{2} \right\rfloor \log_2 \left\lfloor \frac{n}{2} \right\rfloor + n \quad (\text{pela h.i.}) \\
 &\leq 3 \left\lceil \frac{n}{2} \right\rceil \log_2 n + 3 \left\lfloor \frac{n}{2} \right\rfloor (\log_2 n - 1) + n \\
 &= 3 \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \log_2 n - 3 \left\lfloor \frac{n}{2} \right\rfloor + n \\
 &= 3n \log_2 n - 3 \left\lfloor \frac{n}{2} \right\rfloor + n \\
 &\leq 3n \log_2 n.
 \end{aligned}$$

Mostramos o passo da indução!



Exemplo: base da indução

Ainda falta a base.

- ▶ Temos $T(1) = 1$, mas $3 \cdot 1 \cdot \log_2 1 = 0$.
- ▶ A desigualdade não vale para $n = 1$.
- ▶ Não temos uma base da indução.

Ok, queremos mostrar apenas $T(n) \in O(n \log_2 n)$.

- ▶ Basta mostrar que a desigualdade vale para $n \geq n_0$.
- ▶ Vamos escolher $n_0 = 2$.

Base da indução:

- ▶ Para $n = 2$ e $n = 3$, temos:

$$T(2) = T(1) + T(1) + 2 = 4 \leq 3 \cdot 2 \cdot \log_2 2 = 6$$

$$T(3) = T(2) + T(1) + 3 = 8 \leq 3 \cdot 3 \cdot \log_2 3 \approx 14,26$$

- ▶ Por que precisamos de **DOIS CASOS BÁSICOS?**



Modificando um pouco o exemplo

Mas se tivéssemos $T(1) = 8$?

- ▶ A desigualdade não vale para $n = 2$
- ▶ Temos $T(2) = 8 + 8 + 2 = 18$, mas $3 \cdot 2 \cdot \log_2 2 = 6$
- ▶ Podemos escolher uma **constante multiplicativa** maior.
- ▶ Tentando $T(n) \leq 10n \log_2 n$, obtemos

$$T(2) = 18 \leq 10 \cdot 2 \cdot \log_2 2 = 20$$

$$T(3) = 29 \leq 10 \cdot 3 \cdot \log_2 3 \approx 47,55$$

Conclusão:

- ▶ Se o passo da indução vale, então podemos escolher valores adequados para as constantes c e n_0 .



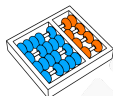
Encontrando as constantes

Suponha que já adivinhamos que $T(n) \in O(n \log_2 n)$:

- ▶ Chutamos que $T(n) \leq cn \log_2 n$ para $c = 3$.
- ▶ Depois escolhemos $n_0 = 2$.
- ▶ Como podemos encontrar essas constantes?

Uma maneira possível:

1. Suponha $T(n) \leq cn \log_2 n$ para algum c genérico.
2. Substitua e desenvolva a expressão para $T(n)$.
3. Determine c e n_0 de forma a provar o passo da indução.



Primeira tentativa

$$\begin{aligned}T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\ &\leq c \left\lceil \frac{n}{2} \right\rceil \log_2 \left\lceil \frac{n}{2} \right\rceil + c \left\lfloor \frac{n}{2} \right\rfloor \log_2 \left\lfloor \frac{n}{2} \right\rfloor + n \\ &\leq c \left\lceil \frac{n}{2} \right\rceil \log_2 n + c \left\lfloor \frac{n}{2} \right\rfloor \log_2 n + n \\ &= c \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \log_2 n + n \\ &= cn \log_2 n + n\end{aligned}$$

Não deu certo...



Segunda tentativa

$$\begin{aligned}
 T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\
 &\leq c \left\lceil \frac{n}{2} \right\rceil \log_2 \left\lceil \frac{n}{2} \right\rceil + c \left\lfloor \frac{n}{2} \right\rfloor \log_2 \left\lfloor \frac{n}{2} \right\rfloor + n \\
 &\leq c \left\lceil \frac{n}{2} \right\rceil \log_2 n + c \left\lfloor \frac{n}{2} \right\rfloor (\log_2 n - 1) + n \\
 &= c \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \log_2 n - c \left\lfloor \frac{n}{2} \right\rfloor + n \\
 &= cn \log_2 n - c \left\lfloor \frac{n}{2} \right\rfloor + n \\
 &\leq cn \log_2 n. \quad (\text{escolhendo } n \geq n_0 \text{ adequado})
 \end{aligned}$$

- ▶ Para a última desigualdade, queremos $-c \lfloor n/2 \rfloor + n \leq 0$.
- ▶ Basta que $c = 3$ e $n_0 \geq 2$.



Completando o exemplo

- ▶ Já mostramos que $T(n) \in O(n \log_2 n)$.
- ▶ Mas queremos mostrar que $T(n) \in \Theta(n \log_2 n)$.
 - ▶ Resta mostrar $T(n) \in \Omega(n \log_2 n)$.
 - ▶ A prova é similar.
 - ▶ Faça como exercício.

ANÁLISE DE ALGORITMOS RECURSIVOS

MO417 - Complexidade de
Algoritmos I

Santiago Valdés Ravelo
[https://ic.unicamp.br/~santiago/
ravelo@unicamp.br](https://ic.unicamp.br/~santiago/ravelo@unicamp.br)

03/24

5



UNICAMP

