

Melhores momentos

AULA PASSADAS

# Resumo

função	consumo de tempo	observação
bubble	$O(n^2)$	todos os casos
insercao	$O(n^2)$ $O(n)$	pior caso melhor caso
insercao_binaria	$O(n^2)$ $O(n \lg n)$	pior caso melhor caso
selecao	$O(n^2)$	todos os casos
merge_sort	$O(n \lg n)$	todos os casos
quick_sort	$O(n^2)$ $O(n \lg n)$	pior caso melhor caso

## Divisão e conquista

Algoritmos por **divisão-e-conquista** têm três passos em cada nível da recursão:

**Dividir:** o problema é dividido em subproblemas de tamanho menor;

**Conquistar:** os subproblemas são resolvidos **recursivamente** e subproblemas “pequenos” são resolvidos diretamente;

**Combinar:** as soluções dos subproblemas são combinadas para obter uma solução do problema original.

**Exemplo:** ordenação por intercalação (**merge\_sort**).

# AULA 13

# Ordenação: algoritmo Mergesort



Fonte: [https://www.youtube.com/watch?v=XaqR3G\\_NVoo](https://www.youtube.com/watch?v=XaqR3G_NVoo)

PF 9

<http://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html>

# Ordenação

$v[0 : n]$  é **crecente** se  $v[0] \leq \dots \leq v[n-1]$ .

**Problema:** Rearranjar um vetor  $v[0 : n]$  de modo que ele fique **crecente**.

Entra:

0											<b>n</b>
33	55	33	44	33	22	11	99	22	55	77	

Sai:

0											<b>n</b>
11	22	22	33	33	33	44	55	55	77	99	

## merge\_sort

Rearranja  $v[p : r]$  em ordem crescente.

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

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

## merge\_sort

Rearranja  $v[p : r]$  em ordem crescente.

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
-----  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

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



## merge\_sort

Rearranja  $v[p : r]$  em ordem crescente.

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

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

## merge\_sort

Rearranja  $v[p : r]$  em ordem crescente.

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

---

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

## merge\_sort

Rearranja  $v[p : r]$  em ordem crescente.

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

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

# Mergesort

*v*

<i>p</i>				<i>q</i>				<i>r</i>
55	33	66	44	99	11	77	22	88

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	55	33	66	44	99	11	77	22	88

v

	<i>p</i>		<i>q</i>		<i>r</i>				
	55	33	66	44					

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	55	33	66	44	99	11	77	22	88

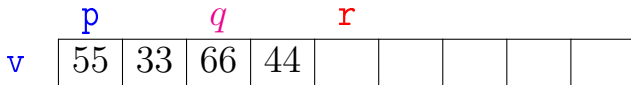
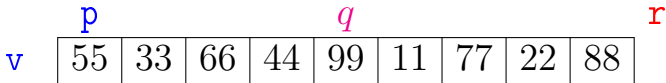
v

	<i>p</i>		<i>q</i>		<i>r</i>				
	55	33	66	44					

v

	<i>p</i>	<i>q</i>	<i>r</i>						
	55	33							

# Mergesort



# Mergesort

v

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

v

$p$		$q$		$r$				
55	33	66	44					

v

$p$	$q$	$r$						
55	33							

v

$p$	$r$							
55								



# Mergesort

v

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

v

$p$		$q$		$r$				
55	33	66	44					

v

$p$	$q$	$r$						
55	33							

v

	$p$	$r$						
55	33							

# Mergesort

v

	<i>p</i>				<i>q</i>				<i>r</i>
	55	33	66	44	99	11	77	22	88

v

	<i>p</i>		<i>q</i>		<i>r</i>				
	55	33	66	44					

v

	<i>p</i>	<i>q</i>	<i>r</i>						
	55	33							

v

	<i>p</i>	<i>r</i>							
	55	33							

# Mergesort

v

	<i>p</i>				<i>q</i>				<i>r</i>
	33	55	66	44	99	11	77	22	88

v

	<i>p</i>		<i>q</i>		<i>r</i>				
	33	55	66	44					

v

	<i>p</i>	<i>q</i>	<i>r</i>						
	33	55							

# Mergesort

v

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

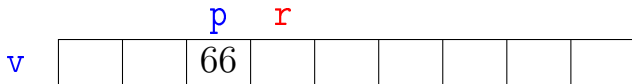
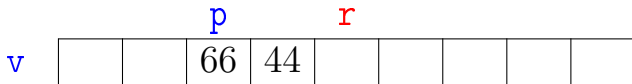
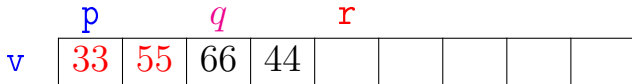
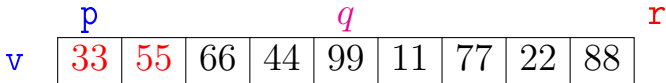
v

	p		q		r				
	33	55	66	44					

v

		p		r					
		66	44						

# Mergesort



# Mergesort

v

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

v

	p		q		r				
	33	55	66	44					

v

		p		r					
		66	44						

v

		p		r					
		66							

# Mergesort

v

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

v

	p		q		r				
	33	55	66	44					

v

			p		r				
			66	44					

v

				p		r			
				44					

# Mergesort

v

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

v

	p		q		r				
	33	55	66	44					

v

			p		r				
			66	44					

v

				p	r				
				44					



# Mergesort

v

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

v

	p		q		r				
	33	55	66	44					

v

			p		r				
			66	44					

# Mergesort

v

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

v

	p		q		r				
	33	55	44	66					

v

			p		r				
			44	66					

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	55	44	66	99	11	77	22	88

v

	<i>p</i>		<i>q</i>		<i>r</i>				
	33	55	44	66					

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

	<i>p</i>		<i>q</i>		<i>r</i>				
	33	44	55	66					

# Mergesort

*v*

<i>p</i>				<i>q</i>				<i>r</i>
33	44	55	66	99	11	77	22	88

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

# Mergesort

*v*

	<i>p</i>			<i>q</i>					<i>r</i>
	33	44	55	66	99	11	77	22	88

*v*

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

*v*

				<i>p</i>	<i>q</i>	<i>r</i>		
				99	11			

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

v

				<i>p</i>	<i>q</i>	<i>r</i>		
				99	11			

v

				<i>p</i>	<i>r</i>			
				99				



# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

v

				<i>p</i>	<i>q</i>	<i>r</i>		
				99	11			

v

				<i>p</i>	<i>r</i>			
				99				

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

v

				<i>p</i>	<i>q</i>	<i>r</i>		
				99	11			

v

					<i>p</i>	<i>r</i>		
					11			

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

v

				<i>p</i>	<i>q</i>	<i>r</i>		
				99	11			

v

					<i>p</i>	<i>r</i>		
					11			

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	99	11	77	22	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				99	11	77	22	88

v

				<i>p</i>	<i>q</i>	<i>r</i>		
				99	11			

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	11	99	77	22	88

v

				<i>p</i>	<i>q</i>			<i>r</i>
				11	99	77	22	88

v

				<i>p</i>	<i>q</i>	<i>r</i>		
				11	99			

# Mergesort

*v*

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	11	99	77	22	88

*v*

				<i>p</i>		<i>q</i>		<i>r</i>
				11	99	77	22	88

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	11	99	77	22	88

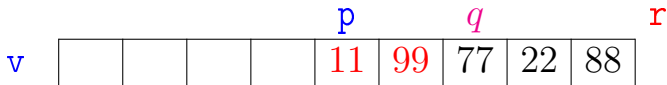
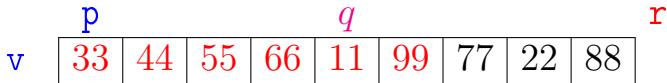
v

				<i>p</i>	<i>q</i>			<i>r</i>
				11	99	77	22	88

v

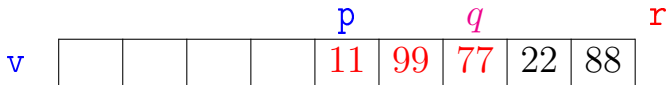
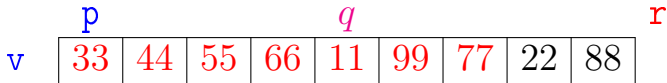
						<i>p</i>	<i>q</i>	<i>r</i>
						77	22	88

# Mergesort

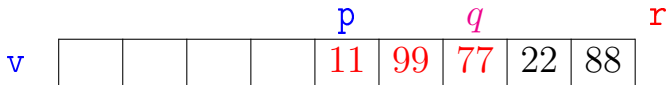
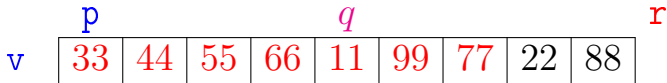




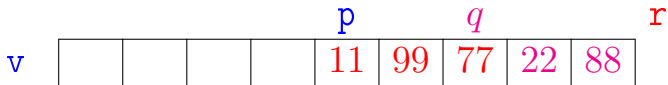
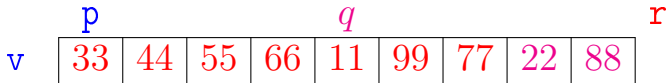
# Mergesort



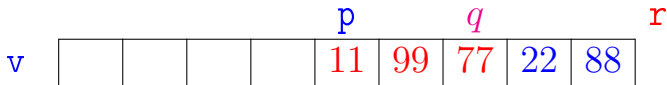
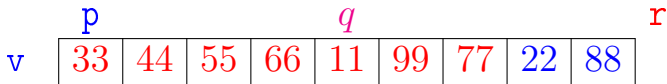
# Mergesort



# Mergesort



# Mergesort



# Mergesort

v

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

v

				p		q		r
				11	99	77	22	88

v

						p	q	r
						77	22	88

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	11	99	22	77	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				11	99	22	77	88

v

						<i>p</i>	<i>q</i>	<i>r</i>
						22	77	88

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	11	99	22	77	88

v

				<i>p</i>		<i>q</i>		<i>r</i>
				11	99	22	77	88

# Mergesort

v

	<i>p</i>			<i>q</i>				<i>r</i>	
	33	44	55	66	11	22	77	88	99

v

				<i>p</i>		<i>q</i>		<i>r</i>
				11	22	77	88	99



# Mergesort

*v*

<i>p</i>				<i>q</i>				<i>r</i>
33	44	55	66	11	22	77	88	99

# Mergesort

$v$

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

# Mergesort

$v$

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

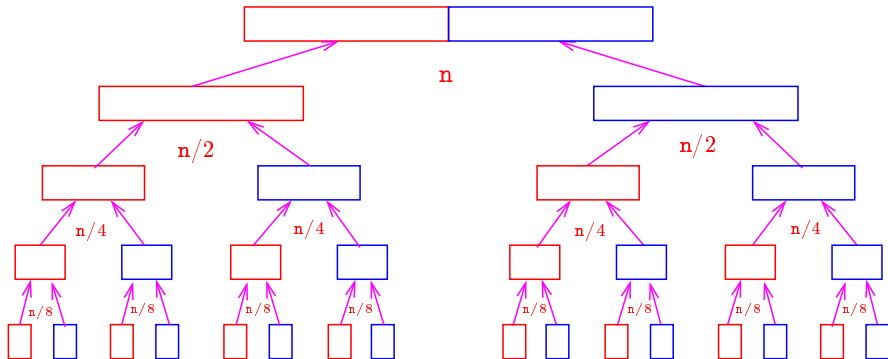
## Correção

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

A função está **correta**?

A **correção** da função, que se apóia na correção do **intercale**, pode ser demonstrada por indução em  $n := r - p$ .

# Consumo de tempo: versão MAC0122



## Consumo de tempo: versão MAC0122

O consumo de tempo em cada nível da recursão é proporcional a  $n$ .

Há cerca de  $\lg n$  níveis de recursão.

nível	consumo de tempo (proporcional a)	
1	$\approx n$	$= n$
2	$\approx n/2 + n/2$	$= n$
3	$\approx n/4 + n/4 + n/4 + n/4$	$= n$
...	...	
$\lg n$	$\approx 1 + 1 + 1 + 1 \cdots + 1 + 1$	$= n$
Total	$\approx n \lg n = O(n \lg n)$	

## Consumo de tempo: outra versão

```
def merge_sort (p, r, v) {  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

Consumo de tempo?

$T(n)$  := consumo de tempo quando  $n = r - p$

## Consumo de tempo: outra versão

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

linha    consumo na linha (proporcional a)

---

1	?
2	?
3	?
4	?
5	?

---

$T(n) = ?$



## Consumo de tempo: outra versão

```
def merge_sort (p, r, v):  
1   if p < r-1:  
2       q = (p + r) // 2  
3       merge_sort(p, q, v)  
4       merge_sort(q, r, v)  
5       intercale(p, q, r, v)
```

linha    consumo na linha (proporcional a)

---

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

---

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

## Consumo de tempo: outra versão

$T(n)$  := consumo de tempo quando  $n = r - p$

$$T(1) = 1$$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \quad \text{para } n = 2, 3, 4, \dots$$

Solução:  $T(n)$  é  $O(n \log n)$ .

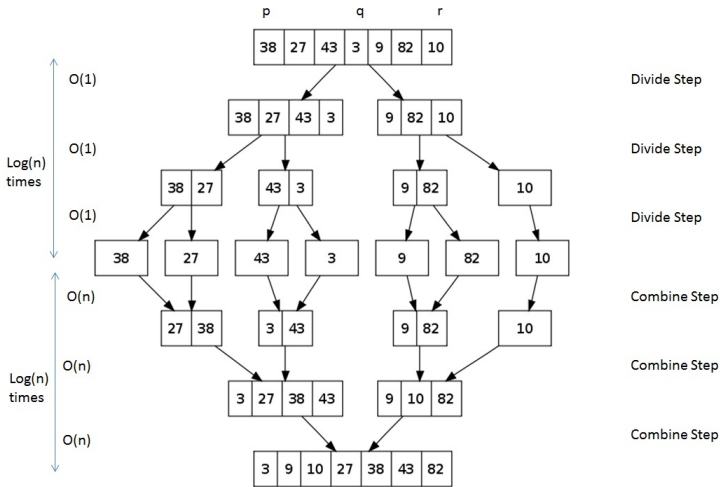
Demonstração: ...

## Conclusão

O consumo de tempo da função `merge_sort` é proporcional a  $n \lg n$ .

O consumo de tempo da função `merge_sort` é  $O(n \lg n)$ .

# Consumo de tempo



$$\begin{aligned} \text{Total Runtime} &= \text{Total time required in Divide} + \text{Total time required in Combine} \\ &= 1 * \text{Log}(n) + n * \text{Log}(n) = n \text{Log}(n). \end{aligned}$$

## Divisão e conquista

Algoritmos por **divisão-e-conquista** têm três passos em cada nível da recursão:

**Dividir:** o problema é dividido em subproblemas de tamanho menor;

**Conquistar:** os subproblemas são resolvidos **recursivamente** e subproblemas “pequenos” são resolvidos diretamente;

**Combinar:** as soluções dos subproblemas são combinadas para obter uma solução do problema original.

**Exemplo:** ordenação por intercalação (**merge\_sort**).

## merge\_sort: versão iterativa

```
def merge_sort (n, v):  
    b = 1  
    while b < n:  
        p = 0  
        while p + b < n:  
            r = p + 2*b  
            if r > n: r = n  
            intercale(p, p+b, r, v)  
            p = p + 2*b  
        b = 2*b
```