

Melhores momentos

AULA 19

Resumo

função	consumo de tempo	observação
bubble	$O(n^2)$	todos os casos
insercao	$O(n^2)$ $O(n)$	piores caso melhor caso
insercaoBinaria	$O(n^2)$ $O(n \lg n)$	piores caso melhor caso
selecao	$O(n^2)$	todos os casos

Intercalação

PF 9

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

Ordenação

$v[0..n-1]$ é crescente se $v[0] \leq \dots \leq v[n-1]$.

Problema: Rearranjar um vetor $v[0..n-1]$ de modo que ele fique crescente.

Entra:

1											$n-1$
33	55	33	44	33	22	11	99	22	55	77	

Sai:

0											$n-1$
11	22	22	33	33	33	44	55	55	77	99	

AULA 20

Intercalação

Problema: Dados $v[p..q-1]$ e $v[q..r-1]$ crescentes, rearranjar $v[p..r-1]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

Entra:

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

Sai:

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

Intercalação

v

22	33	55	77	11	44	66	88	99
----	----	----	----	----	----	----	----	----

p q r

w

--	--	--	--	--	--	--	--	--

Navigation icons

Intercalação

v

--	--	--	--	--	--	--	--	--

k

w

22	33	55	77	99	88	66	44	11
----	----	----	----	----	----	----	----	----

i j

Navigation icons

Intercalação

v

11								
----	--	--	--	--	--	--	--	--

k

w

22	33	55	77	99	88	66	44	11
----	----	----	----	----	----	----	----	----

i j

Navigation icons

Intercalação

v

11	22							
----	----	--	--	--	--	--	--	--

k

w

22	33	55	77	99	88	66	44	11
----	----	----	----	----	----	----	----	----

i j

Navigation icons

Intercalação

v

11	22	33						
----	----	----	--	--	--	--	--	--

k

w

22	33	55	77	99	88	66	44	11
----	----	----	----	----	----	----	----	----

i j

Navigation icons

Intercalação

v

11	22	33	44					
----	----	----	----	--	--	--	--	--

k

w

22	33	55	77	99	88	66	44	11
----	----	----	----	----	----	----	----	----

i j

Navigation icons

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha	execuções da linha
1	?
2	?
3	?
4	?
5	?
6	?
7-9	?
total	?

Navigation icons

Conclusão

A função `intercala` consome $5n$ unidades de tempo.

O algoritmo `intercala` consome $O(n)$ unidades de tempo.

Também escreve-se

O algoritmo `intercala` consome tempo $O(n)$.

Navigation icons

Ordenação

$v[0..n-1]$ é crescente se $v[0] \leq \dots \leq v[n-1]$.

Problema: Rearranjar um vetor $v[0..n-1]$ de modo que ele fique crescente.

Entra:

0												$n-1$
33	55	33	44	33	22	11	99	22	55	77		

Sai:

0												$n-1$
11	22	22	33	33	33	44	55	55	77	99		

Navigation icons

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é ($n := r - p$):

linha	execuções da linha
1	$= q - p = n - r + q$
2	$= q - p - 1 = n - r + q - 1$
3	$= r - q = n - q + p$
4	$= r - q - 1 = n - q + p - 1$
5	$= 1$
6	$= r - p + 1 = n + 1$
7-9	$= 2(r - p) = 2n$
total	$= 7n - 2(r - p) = 5n$

Navigation icons

Ordenação: algoritmo Mergesort

PF 9

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

Navigation icons

Função mergeSort

Rearranja $v[p..r-1]$ em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {  
1  if (p < r-1) {  
2      int q = (p + r)/2;  
3      mergeSort(p, q, v);  
4      mergeSort(q, r, v);  
5      intercala(p, q, r, v);  
    }  
}
```

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

Navigation icons

função mergeSort

Rearranja $v[p..r-1]$ em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
}
}
```

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

Navigation icons

função mergeSort

Rearranja $v[p..r-1]$ em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
}
}
```

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

Navigation icons

função mergeSort

Rearranja $v[p..r-1]$ em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
}
}
```

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

Navigation icons

função mergeSort

Rearranja $v[p..r-1]$ em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
}
}
```

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

Navigation icons

Mergesort

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

Navigation icons

Mergesort

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

	p		q		r				
v	55	33	66	44					

Navigation icons

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								

Navigation icons

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									

Navigation icons

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									

Navigation icons

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								

Navigation icons

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								

Navigation icons

Mergesort

v

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

v

	p		q		r				
33	55	66	44						

v

	p	q	r						
33	55								

Navigation icons

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					

Navigation icons

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						

Navigation icons

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						

Navigation icons

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					

Navigation icons

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					

Navigation icons

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					

Navigation icons

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						

Navigation icons

Mergesort

v

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

v

	p		q		r				
33	55	44	66						

Navigation icons

Mergesort

v

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

v

	p		q		r				
33	44	55	66						

Navigation icons

Mergesort

v

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

Navigation icons

Mergesort

v

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

v

				p		q		r	
				99	11	77	22	88	

Navigation icons

Mergesort

v

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

v

				p		q		r	
				99	11	77	22	88	

Navigation icons

v

				p	q	r			
				99	11				

Mergesort

v

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

v

				^p 99	11	^q 77	22	88	^r
--	--	--	--	-----------------	----	-----------------	----	----	--------------

v

				^p 99	^q 11					^r
--	--	--	--	-----------------	-----------------	--	--	--	--	--------------

v

				^p 99	^r					
--	--	--	--	-----------------	--------------	--	--	--	--	--

Navigation icons

Mergesort

v

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

v

				^p 99	11	^q 77	22	88	^r
--	--	--	--	-----------------	----	-----------------	----	----	--------------

v

				^p 99	^q 11					^r
--	--	--	--	-----------------	-----------------	--	--	--	--	--------------

v

				^p 99	^r					
--	--	--	--	-----------------	--------------	--	--	--	--	--

Navigation icons

Mergesort

v

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

v

				^p 99	11	^q 77	22	88	^r
--	--	--	--	-----------------	----	-----------------	----	----	--------------

v

				^p 99	^q 11					^r
--	--	--	--	-----------------	-----------------	--	--	--	--	--------------

v

					^p 11	^r				
--	--	--	--	--	-----------------	--------------	--	--	--	--

Navigation icons

Mergesort

v

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

v

				^p 99	11	^q 77	22	88	^r
--	--	--	--	-----------------	----	-----------------	----	----	--------------

v

				^p 99	^q 11					^r
--	--	--	--	-----------------	-----------------	--	--	--	--	--------------

v

					^p 11	^r				
--	--	--	--	--	-----------------	--------------	--	--	--	--

Navigation icons

Mergesort

v

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

v

				^p 99	11	^q 77	22	88	^r
--	--	--	--	-----------------	----	-----------------	----	----	--------------

v

				^p 99	^q 11					^r
--	--	--	--	-----------------	-----------------	--	--	--	--	--------------

Navigation icons

Mergesort

v

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

v

				^p 11	99	^q 77	22	88	^r
--	--	--	--	-----------------	----	-----------------	----	----	--------------

v

				^p 11	^q 99					^r
--	--	--	--	-----------------	-----------------	--	--	--	--	--------------

Navigation icons

Mergesort

v

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

v

				p		q		r
				11	99	77	22	88

Navigation icons

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

Navigation icons

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

v

						p	r	
						77		

Navigation icons

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

v

						p	r	
						77		

Navigation icons

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

v

							p	q	r
							22	88	

Navigation icons

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

v

							p	q	r
							22	88	

Navigation icons

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	

v

							p	q	r
							22	88	

Navigation icons

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	

Navigation icons

Mergesort

v

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

v

				p		q			r
				11	99	22	77	88	

v

						p	q		r
						22	77	88	

Navigation icons

Mergesort

v

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

v

				p		q			r
				11	99	22	77	88	

Navigation icons

Mergesort

v

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

v

				p		q			r
				11	22	77	88	99	

Navigation icons

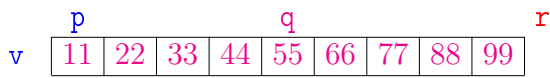
Mergesort

v

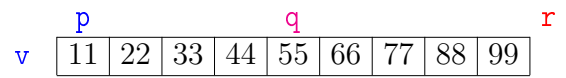
	p				q				r
33	44	55	66	11	22	77	88	99	

Navigation icons

Mergesort



Mergesort



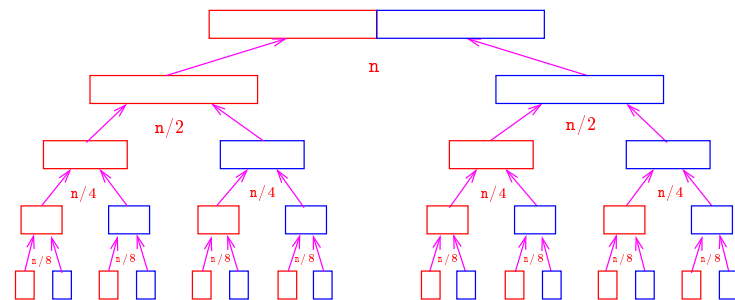
Correção

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
}
}
```

A função está **correta**?

A **correção** da função, que se apóia na correção do **intercala**, 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: versão MAC0338

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
}
}
```

Consumo de tempo?

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

Consumo de tempo: versão MAC0338

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(p, q, r, v);
    }
}
```

linha	consumo na linha (proporcional a)
1	?
2	?
3	?
4	?
5	?

$$T(n) = ?$$

◀ ▶ ↺ ↻ 🔍

Consumo de tempo: versão MAC0338

```
void mergeSort (int p, int r, int v[]) {
1  if (p < r-1) {
2      int q = (p + r)/2;
3      mergeSort(p, q, v);
4      mergeSort(q, r, v);
5      intercala(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: versão MAC0338

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

$$T(1) = 1$$

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

Solução: $T(n)$ é $O(n \log n)$.

Demonstração: ...

Em MAC0338 vocês verão como estimar “a ordem” de recorrências.

◀ ▶ ↺ ↻ 🔍

Conclusão

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

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

◀ ▶ ↺ ↻ 🔍

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 (`mergeSort`).

◀ ▶ ↺ ↻ 🔍

mergeSort: versão iterativa

```
void mergeSort (int n, int v[]){
    int p, r;
    int b = 1;
    while (b < n) {
        p = 0;
        while (p + b < n) {
            r = p + 2*b;
            if (r > n) r = n;
            intercala(p, p+b, r, v);
            p = p + 2*b;
        }
        b = 2*b;
    }
}
```

◀ ▶ ↺ ↻ 🔍