Intercalação

Fonte: http://csunplugged.org/sorting-algorithms

PF 9

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

Intercalação

Problema: Dados $v[p:q]$ e $v[q:r]$ crescentes, rearranjar $v[p:r]$ 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

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

w									

Intercalação

	k								
v									

	i								j
w	22	33	55	77	99	88	66	44	11

Intercalação

		k							
v	11								

	i							j	
w	22	33	55	77	99	88	66	44	11

Intercalação

			k						
v	11	22							

		i						j	
w	22	33	55	77	99	88	66	44	11

Intercalação

				k					
v	11	22	33						

			i					j	
w	22	33	55	77	99	88	66	44	11

Intercalação

					k				
v	11	22	33	44					

			i				j		
w	22	33	55	77	99	88	66	44	11

Intercalação

						k			
v	11	22	33	44	55				

				i			j		
w	22	33	55	77	99	88	66	44	11

Intercalação

							k		
v	11	22	33	44	55	66			

				i		j			
w	22	33	55	77	99	88	66	44	11

Intercalação

								k	
v	11	22	33	44	55	66	77		

					i	j			
w	22	33	55	77	99	88	66	44	11

Intercalação

									k
v	11	22	33	44	55	66	77	88	

					ij				
w	22	33	55	77	99	88	66	44	11

Intercalação

										k
v	11	22	33	44	55	66	77	88	99	

					j	i			
w	22	33	55	77	99	88	66	44	11

Intercalação

```
def intercale(p, q, r, v):
    e = v[p:q] # clone
    d = v[q:r] # clone
    d.reverse() # método mutador
    w = e + d
    i = 0
    j = r-p-1
    for k in range(p,r):
        if w[i] <= w[j]):
            v[k] = w[i]
            i += 1
        else:
            v[k] = w[j]
            j -= 1
```

Consumo de tempo

$n := r - p$

linha	proporcional a
1–2	?
3	?
4	?
5–6	?
7	?
8–13	?
total	?

Consumo de tempo

$\mathtt{n} := \mathtt{r} - \mathtt{p}$

linha		proporcional a		
1–2	$=$	$\mathtt{n}$		
3	$\leq$	$\mathtt{n}$		
4	$=$	$\mathtt{n}$		
5–6	$=$	1		
7	$=$	$\mathtt{r} - \mathtt{p} + 1$	$=$	$\mathtt{n} + 1$
8–13	$=$	$\mathtt{r} - \mathtt{p}$	$=$	$\mathtt{n}$
total	$\approx$	$5\mathtt{n} + 2$		

Conclusão

A função `intercale` consome $5\mathtt{n} + 2$ unidades de tempo.

O algoritmo `intercale` consome $O(\mathtt{n})$ unidades de tempo.

Também escreve-se

O algoritmo `intercale` consome tempo $O(\mathtt{n})$.

Ordenação: algoritmo Mergesort

Fonte: 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 crescente se $v[0] \leq \cdots \leq v[n-1]$.

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

Entra:

0 n

33	55	33	44	33	22	11	99	22	55	77

Sai:

0 n

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

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

Mergesort

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

Mergesort

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

	p		q		r				
v	55	33	66	44					

Mergesort

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

	p		q		r				
v	55	33	66	44					

	p	q	r						
v	55	33							

Mergesort

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

	p		q		r				
v	55	33	66	44					

	p	q	r						
v	55	33							

	p	r							
v	55								

Mergesort

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

	p		q		r				
v	55	33	66	44					

	p	q	r						
v	55	33							

	p	r							
v	55								

Mergesort

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

	p		q		r				
v	55	33	66	44					

	p	q	r						
v	55	33							

		p	r						
v	55	33							

Mergesort

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

	p		q		r				
v	55	33	66	44					

	p	q	r						
v	55	33							

		p	r						
v	55	33							

Mergesort

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

	p		q		r				
v	33	55	66	44					

	p	q	r						
v	33	55							

Mergesort

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

	p		q		r				
v	33	55	66	44					

			p		r				
v			66	44					

Mergesort

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

	p		q		r					
v	33	55	66	44						

			p		r					
v			66	44						

			p	r						
v			66							

Mergesort

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

	p		q		r				
v	33	55	66	44					

			p		r				
v			66	44					

			p	r					
v			66						

Mergesort

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

	p		q		r				
v	33	55	66	44					

			p		r				
v			66	44					

				p	r				
v				44					

Mergesort

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

	p		q		r				
v	33	55	66	44					

			p		r				
v			66	44					

				p	r				
v				44					

Mergesort

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

	p		q		r				
v	33	55	66	44					

			p		r				
v			66	44					

Mergesort

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

	p		q		r				
v	33	55	44	66					

			p		r				
v			44	66					

Mergesort

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

	p		q		r				
v	33	55	44	66					

Mergesort

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

	p		q		r				
v	33	44	55	66					

Mergesort

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

Mergesort

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

					p		q			r
v					99	11	77	22	88	

Mergesort

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

					p		q			r
v					99	11	77	22	88	

					p	q	r		
v					99	11			

Mergesort

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

					p		q			r
v					99	11	77	22	88	

					p	q	r		
v					99	11			

					p	r			
v					99				

Mergesort

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

					p		q			r
v					99	11	77	22	88	

					p	q	r			
v					99	11				

					p	r				
v					99					

Mergesort

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

					p		q			r
v					99	11	77	22	88	

					p	q	r		
v					99	11			

						p	r		
v						11			

Mergesort

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

					p		q			r
v					99	11	77	22	88	

					p	q	r			
v					99	11				

						p	r			
v						11				

Mergesort

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

					p		q			r
v					99	11	77	22	88	

					p	q	r		
v					99	11			

Mergesort

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

					p		q			r
v					11	99	77	22	88	

					p	q	r			
v					11	99				

Mergesort

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

					p		q			r
v					11	99	77	22	88	

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

							p	r		
v							77			

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

							p	r		
v							77			

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

								p	q	r
v								22	88	

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

								p	q	r
v								22	88	

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

								p	q	r
v								22	88	

Mergesort

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

					p		q			r
v					11	99	77	22	88	

							p	q		r
v							77	22	88	

Mergesort

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

					p		q			r
v					11	99	22	77	88	

							p	q		r
v							22	77	88	

Mergesort

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

					p		q			r
v					11	99	22	77	88	

Mergesort

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

					p		q			r
v					11	22	77	88	99	

Mergesort

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

Mergesort

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

Mergesort

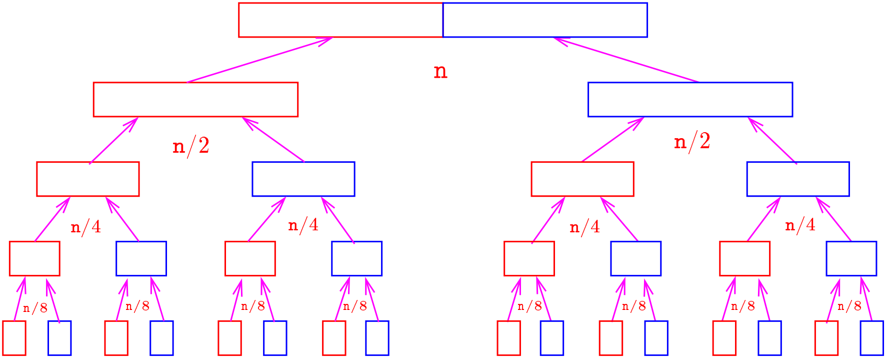
	p				q					r
v	11	22	33	44	55	66	77	88	99	

Correção

```
def merge_sort (p, r, v):
    if p < r-1:
        q = (p + r) // 2
        merge_sort(p, q, v)
        merge_sort(q, r, v)
        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	$\approx n/4 + n/4 + n/4 + n/4$	$= n$
$\ldots$	$\ldots$	
$\lg n$	$\approx 1 + 1 + 1 + 1 \cdots + 1 + 1$	$= n$
Total	$\approx n \lg n = O(n \lg n)$	

Consumo de tempo: outra versão

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

$$T(n) \quad = \; ?$$

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

$\mathtt{T}(\mathtt{n}) :=$ consumo de tempo quando $\mathtt{n} = \mathtt{r} - \mathtt{p}$

$$\mathtt{T}(1) = 1$$
$$\mathtt{T}(\mathtt{n}) = \mathtt{T}(\lceil \mathtt{n}/2 \rceil) + \mathtt{T}(\lfloor \mathtt{n}/2 \rfloor) + \mathtt{n} \quad \text{para } \mathtt{n} = 2, 3, 4, \ldots$$

Solução: $\mathtt{T}(\mathtt{n})$ é $O(\mathtt{n} \log \mathtt{n})$.

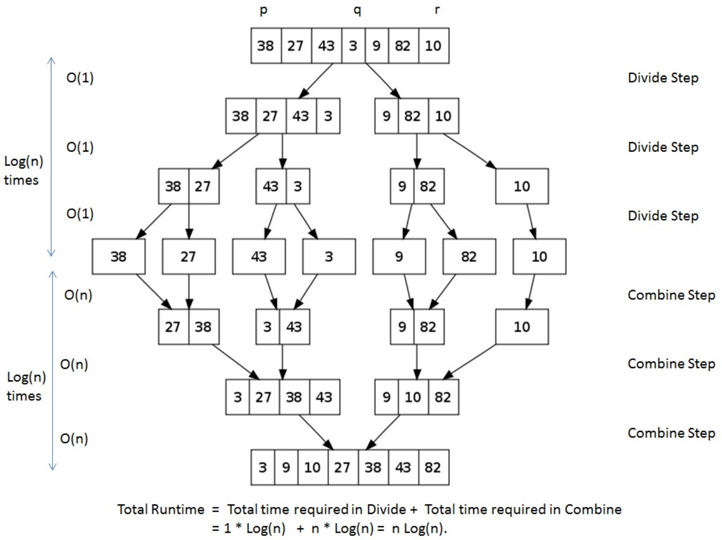
Demonstração: ...

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



Fonte: http://images.1233.tw/in-place-quicksort-algorithm/

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
```