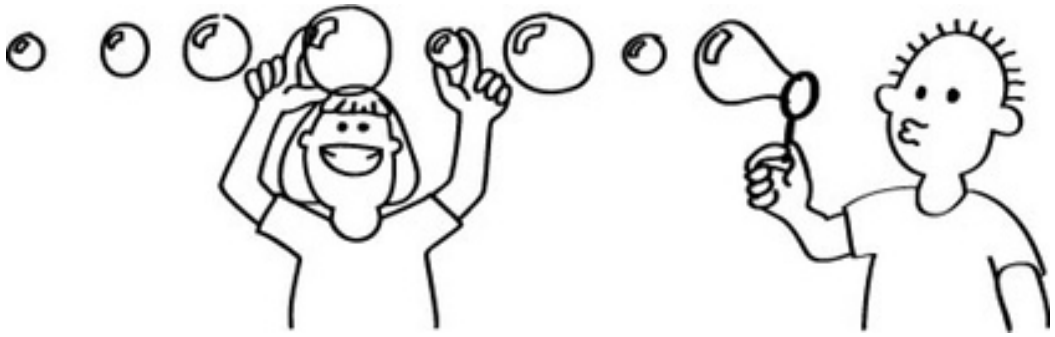
AULA 16

Intercalação



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

PF 9

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

Intercalação

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

Entra:

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

Intercalação

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

Entra:

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

Sai:

	p								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

```
void intercala(int p,int q,int r,int v[]) {
    int i, j, k, *w;
    w = mallocSafe((r-p)*sizeof(int));
1   for (i = 0, k = p; k < q; i++, k++)
2      w[i] = v[k];
3   for (j = r-p-1; k < r; j--, k++)
4      w[j] = v[k];
```

Intercalação

```
void intercala(int p,int q,int r,int v[]) {
    int i, j, k, *w;
    w = mallocSafe((r-p)*sizeof(int));
1   for (i = 0, k = p; k < q; i++, k++)
2      w[i] = v[k];
3   for (j = r-p-1; k < r; j--, k++)
4      w[j] = v[k];
5   i = 0; j = r-p-1;
6   for (k = p; k < r; k++)
7      if (w[i] <= w[j])
8         v[k] = w[i++];
9      else v[k] = w[j--];
    free (w);
}
```

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	?
8–9	?
total	?

Consumo de tempo

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

linha	execuções da linha			
1	$=$	$\mathtt{q} - \mathtt{p} + 1$		
2	$=$	$\mathtt{q} - \mathtt{p}$		
3	$=$	$\mathtt{r} - \mathtt{q} + 1$		
4	$=$	$\mathtt{r} - \mathtt{q}$		
5	$=$	1		
6	$=$	$\mathtt{r} - \mathtt{p} + 1$	$=$	$\mathtt{n} + 1$
7	$=$	$\mathtt{r} - \mathtt{p}$	$=$	$\mathtt{n}$
8–9	$=$	$\mathtt{r} - \mathtt{p}$	$=$	$\mathtt{n}$
total	$=$	$5\mathtt{n} + 4$		

Conclusão

A função `intercala` consome
$5\mathtt{n} + 4$ unidades de tempo.

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

Também escreve-se

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

Ordenação: algoritmo Mergesort

Fonte: https://www.youtube.com/watch?v=dENca26N6V4

PF 9

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

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}{-}1]$ é crescente se $\mathtt{v}[0] \leq \cdots \leq \mathtt{v}[\mathtt{n}{-}1]$.

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

Entra:

0										n−1
33	55	33	44	33	22	11	99	22	55	77

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \leq \cdots \leq \mathtt{v}[\mathtt{n}-1]$.

Problema: Rearranjar um vetor $\mathtt{v}[0\,..\,\mathtt{n}-1]$ 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

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \leq \cdots \leq \mathtt{v}[\mathtt{n}-1]$.

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

Ideia:

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

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \leq \cdots \leq \mathtt{v}[\mathtt{n}-1]$.

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

Ideia:

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

Ordena metade esquerda...

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \leq \cdots \leq \mathtt{v}[\mathtt{n}-1]$.

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

Ideia:

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

Ordena metade esquerda... Ordena metade direita...

Ordenação

$\mathtt{v}[0\,..\,\mathtt{n}-1]$ é crescente se $\mathtt{v}[0] \leq \cdots \leq \mathtt{v}[\mathtt{n}-1]$.

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

Ideia:

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

Ordena metade esquerda... Ordena metade direita... Intercala!

Função mergeSort

Rearranja v[p . . r−1] em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {  /* pelo menos dois elementos */
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

função mergeSort

Rearranja v[p . . r−1] em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {   /* pelo menos dois elementos */
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

função mergeSort

Rearranja v[p . . r−1] em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {   /* pelo menos dois elementos */
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

função mergeSort

Rearranja v[p . . r−1] em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {   /* pelo menos dois elementos */
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

função mergeSort

Rearranja v[p . . r−1] em ordem crescente.

```
void mergeSort (int p, int r, int v[]) {
1   if (p < r-1) {  /* pelo menos dois elementos */
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

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

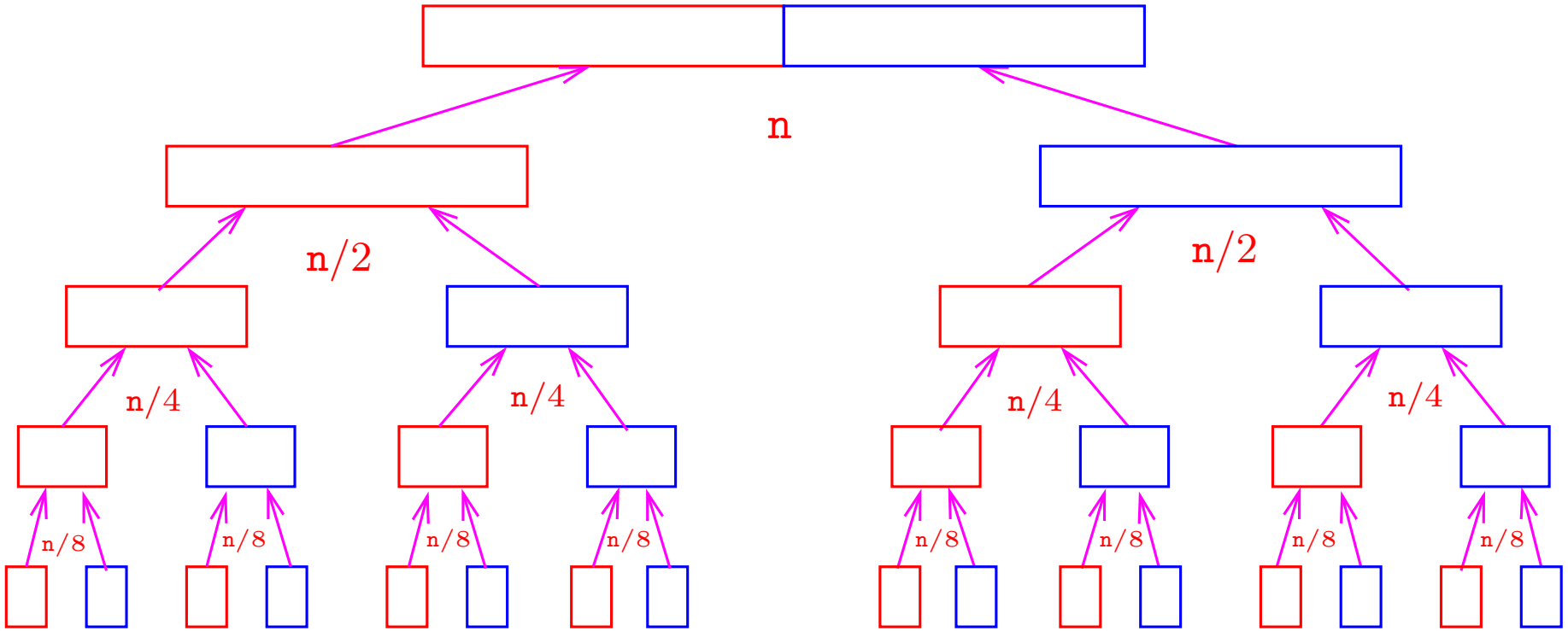
Correção

```
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 apoia 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 $\mathtt{n}$.

Há cerca de $\lg \mathtt{n}$ níveis de recursão.

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

Consumo de tempo: outra versão

```
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?

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

Consumo de tempo: outra versão

```
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: outra versão

```
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: 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 $\mathrm{O}(\mathtt{n} \log \mathtt{n})$.

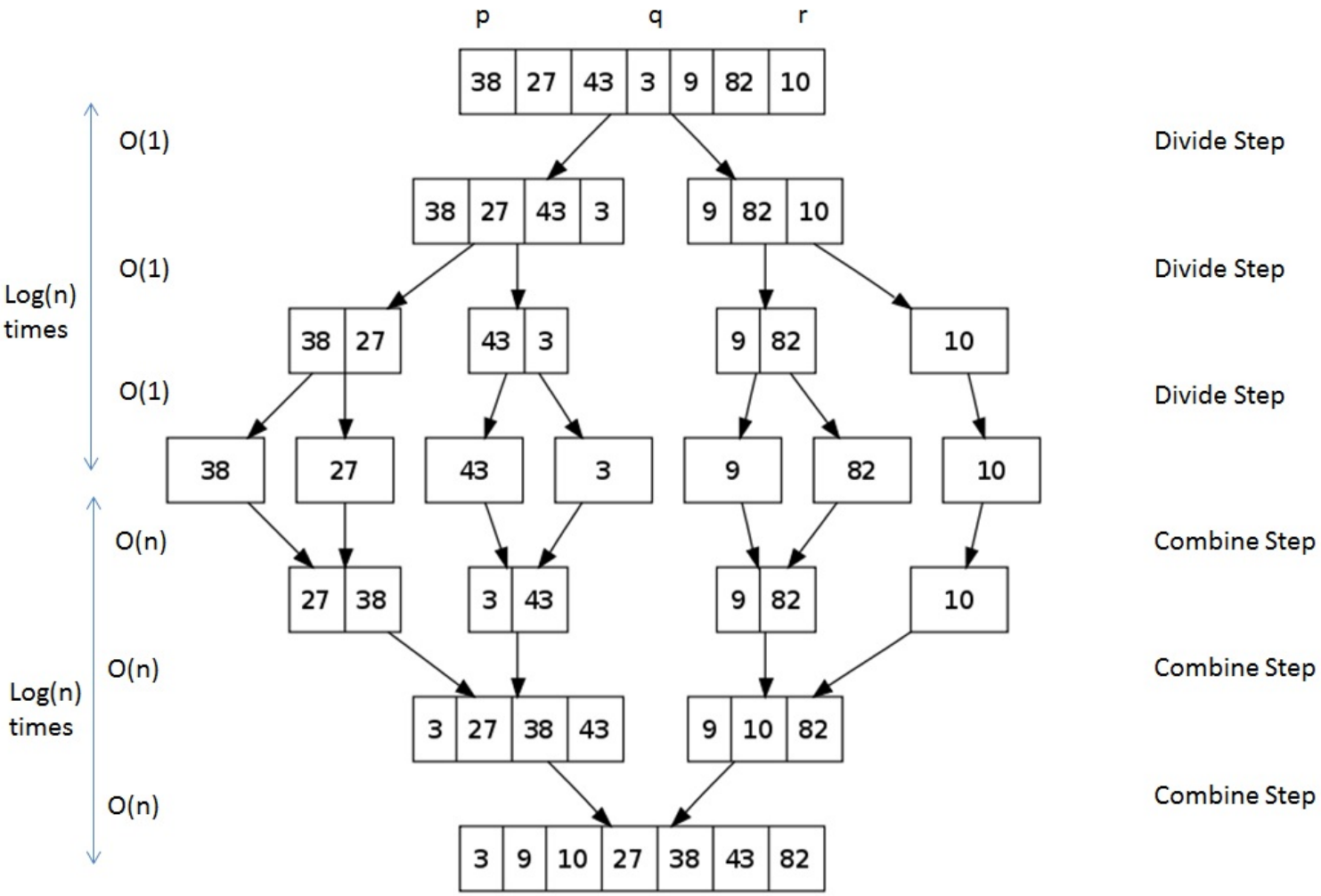
Demonstração: ...

Conclusão

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

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

Consumo de tempo



Total Runtime = Total time required in Divide + Total time required in Combine
= 1 * Log(n) + n * Log(n) = n Log(n).

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 (`mergeSort`).

mergeSort: versão iterativa

```
void mergeSort (int n, int v[]) {
   int p, r, 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;
   }
}
```

mergeSort: versão iterativa

```
void mergeSort (int n, int v[]) {
   int p, r, 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;
   }
}
```

Consumo de tempo: $\mathrm{O}(n \lg n)$