Melhores momentos

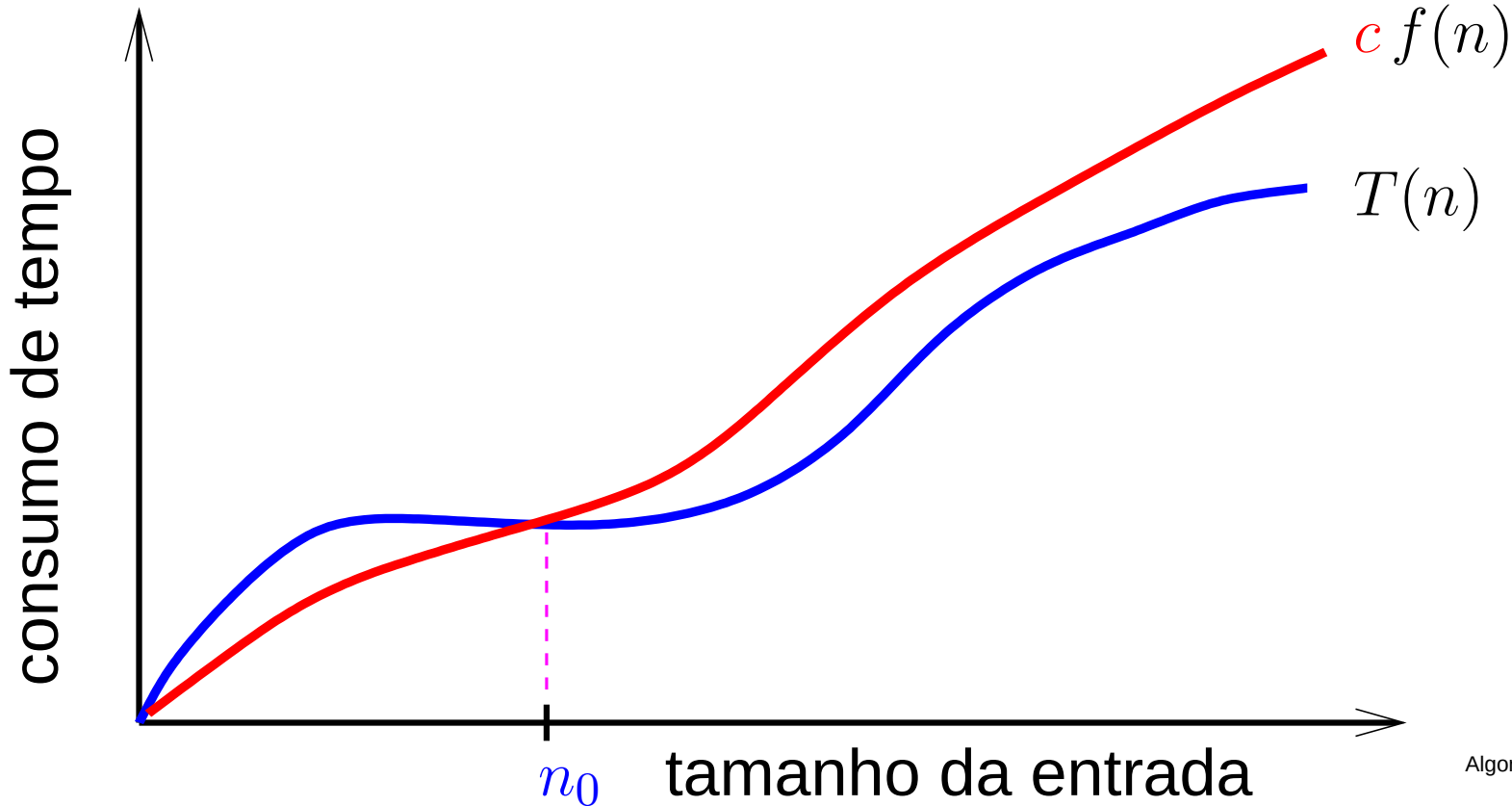
AULA 2

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros no reais. Dizemos que $T(n)$ é $\mathrm{O}(f(n))$ se existem constantes positivas c e n_0 tais que

$$T(n) \leq c\, f(n)$$

para todo $n \geq n_0$.



Notação O

Intuitivamente...

$\mathrm{O}(f(n))$ $\approx$ funções q não crescem mais rápido que $f(n)$

$\approx$ funções menores ou iguais a um múltiplo de $f(n)$

n^2 $(3/2)n^2$ $9999n^2$ $n^2/1000$ etc.

crescem todas com a mesma velocidade, são todas $\mathrm{O}(n^2)$.

- $n^2 + 99n$ é $\mathrm{O}(n^2)$
- $33n^2$ é $\mathrm{O}(n^2)$
- $9n + 2$ é $\mathrm{O}(n^2)$
- $0{,}00001n^3 - 200n^2$ não é $\mathrm{O}(n^2)$

Uso da notação O

$\mathrm{O}(f(n)) = \{T(n) :$ existem c e n_0 tq $T(n) \leq cf(n), n \geq n_0\}$

"$T(n)$ é $\mathrm{O}(f(n))$" deve ser entendido como "$T(n) \in \mathrm{O}(f(n))$".

"$T(n) = \mathrm{O}(f(n))$" deve ser entendido como "$T(n) \in \mathrm{O}(f(n))$".

"$T(n) \leq \mathrm{O}(f(n))$" é feio.

"$T(n) \geq \mathrm{O}(f(n))$" não faz sentido!

"$T(n)$ é $g(n) + \mathrm{O}(f(n))$" significa que existem constantes positivas c e n_0 tais que

$$T(n) \leq g(n) + c\, f(n)$$

para todo $n \geq n_0$.

Ordenação por inserção

Algoritmo rearranja $A[p\,..\,r]$ em ordem crescente

```
ORDENA-POR-INSERÇÃO (A, p, r)
1    para j ← p + 1 até r faça
2         chave ← A[j]

3         i ← j − 1
4         enquanto i ≥ p e A[i] > chave faça
5              A[i + 1] ← A[i]      ▷ desloca
6              i ← i − 1

7         A[i + 1] ← chave      ▷ insere
```

Quanto tempo o algoritmo consome?
"Tamanho" do problema: $n := r - p + 1$

Consumo de tempo

linha	consumo de todas as execuções da linha
1	$\mathrm{O}(n)$
2	$\mathrm{O}(n)$
3	$\mathrm{O}(n)$
4	$n\mathrm{O}(n) = \mathrm{O}(n^2)$
5	$n\mathrm{O}(n) = \mathrm{O}(n^2)$
6	$n\mathrm{O}(n) = \mathrm{O}(n^2)$
7	$\mathrm{O}(n)$
total	$\mathrm{O}(3n^2 + 4n) = \mathrm{O}(n^2)$

Justificativa

Bloco de linhas 4–6 é executado $\leq n$ vezes;
cada execução consome $\mathrm{O}(n)$;
todas juntas consomem $n\mathrm{O}(n)$.

Êpa!

Quem garante que $n\mathrm{O}(n) = \mathrm{O}(n^2)$?
Quem garante que $\mathrm{O}(n^2) + \mathrm{O}(n^2) + \mathrm{O}(n^2) = \mathrm{O}(3n^2)$?
Quem garante que $\mathrm{O}(3n^2 + 4n) = \mathrm{O}(n^2)$?
Veja exercícios de Mais notação O.

Conclusão:

O algoritmo consome $\mathrm{O}(n^2)$ unidades de tempo.

Notação O cai como uma luva!

$n\mathrm{O}(n) = \mathrm{O}(n^2)$

"$n\mathrm{O}(n) = \mathrm{O}(n^2)$" significa "$nO(n) \subset \mathrm{O}(n^2)$".
Ou seja, se $T(n)$ é $\mathrm{O}(n)$, então $nT(n)$ é $\mathrm{O}(n^2)$.

De fato, se $T(n)$ é $\mathrm{O}(n)$ então existem constantes, digamos 10 e 10^{100}, tais que

$$T(n) \leq 10\,n$$

para todo $n \geq 10^{100}$. Desta forma,

$$n\,T(n) \leq n\,10\,n \leq 10\,n^2$$

para todo $n \geq 10^{100}$. Logo $n\,T(n)$ é $\mathrm{O}(n^2)$.

$n\mathrm{O}(n) = \mathrm{O}(n^2)$

"$n\mathrm{O}(n) = \mathrm{O}(n^2)$" significa "$nO(n) \subset \mathrm{O}(n^2)$".
Ou seja, se $T(n)$ é $\mathrm{O}(n)$, então $nT(n)$ é $\mathrm{O}(n^2)$.

De fato, se $T(n)$ é $\mathrm{O}(n)$ então existem constantes positivas c e n_0, tais que

$$T(n) \leq c\,n$$

para todo $n \geq n_0$. Desta forma,

$$n\,T(n) \leq n\,c\,n \leq c\,n^2$$

para todo $n \geq n_0$. Logo $n\,T(n)$ é $\mathrm{O}(n^2)$.

AULA 3

Mais análise com notação O

CLRS 2.1–2.2

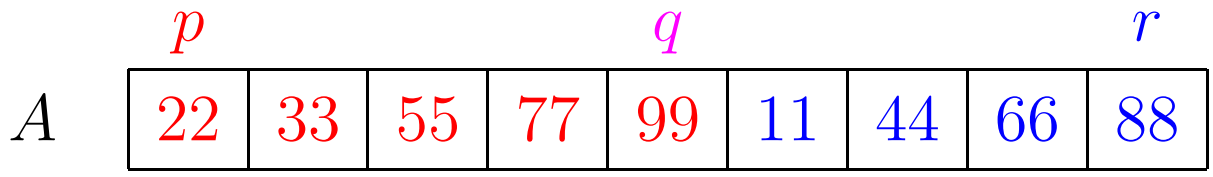
AU 3.3, 3.6 (muito bom)

Análise da intercalação

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

Para que valores de q o problema faz sentido?

Entra:

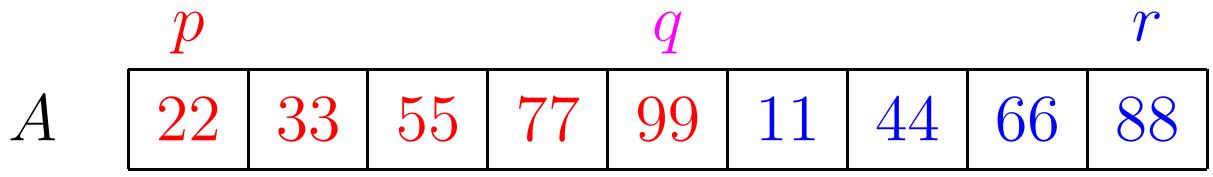


Análise da intercalação

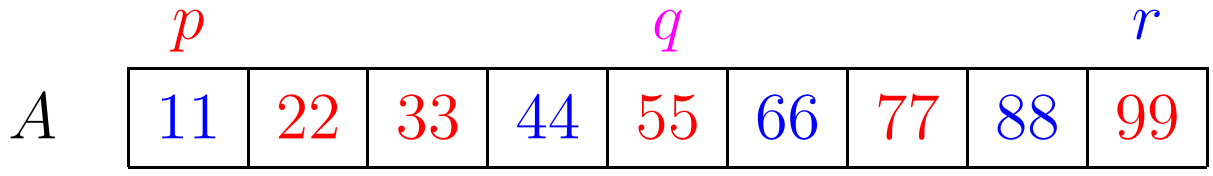
Problema: Dados $A[p\,..\,q]$ e $A[q{+}1\,..\,r]$ crescentes, rearranjar $A[p\,..\,r]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

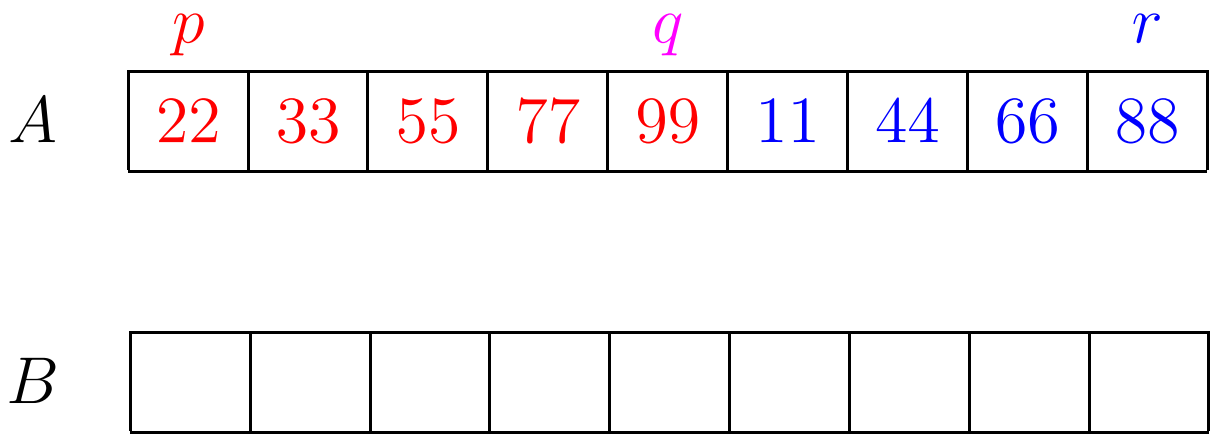
Entra:



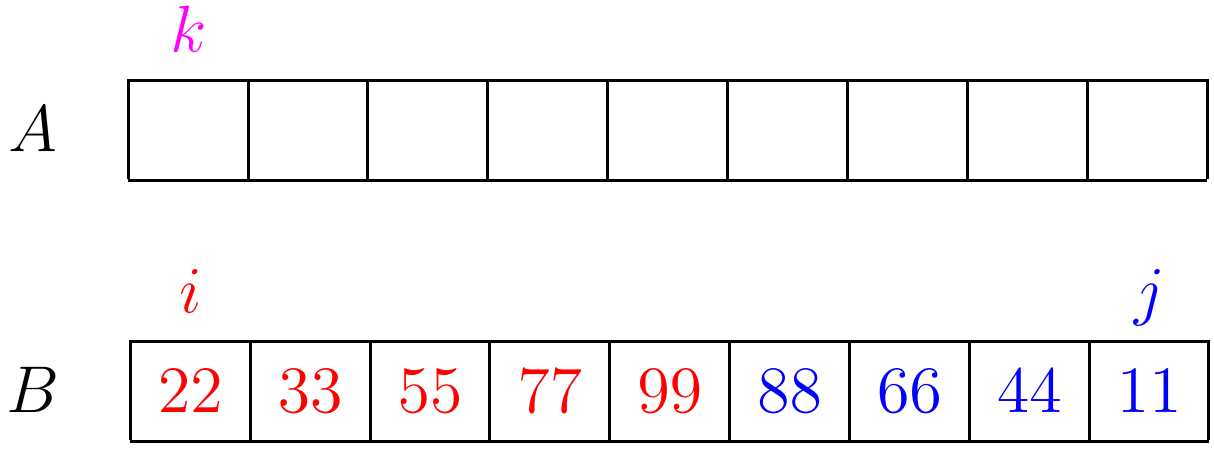
Sai:



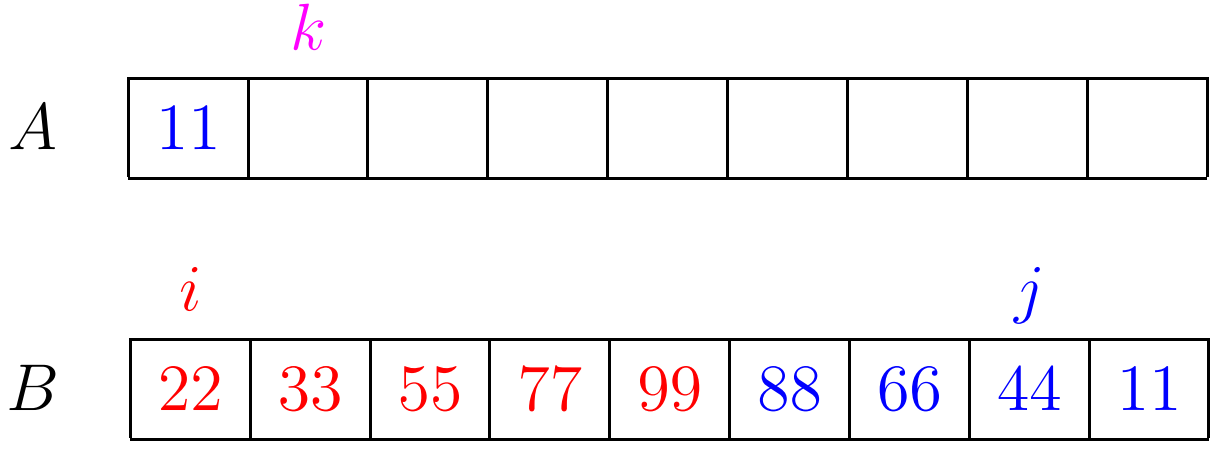
Intercalação



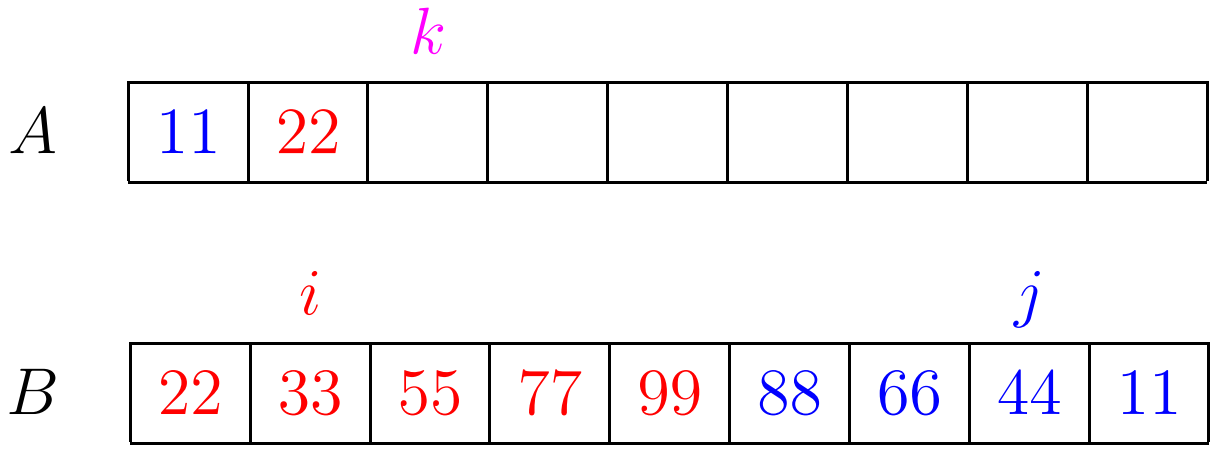
Intercalação



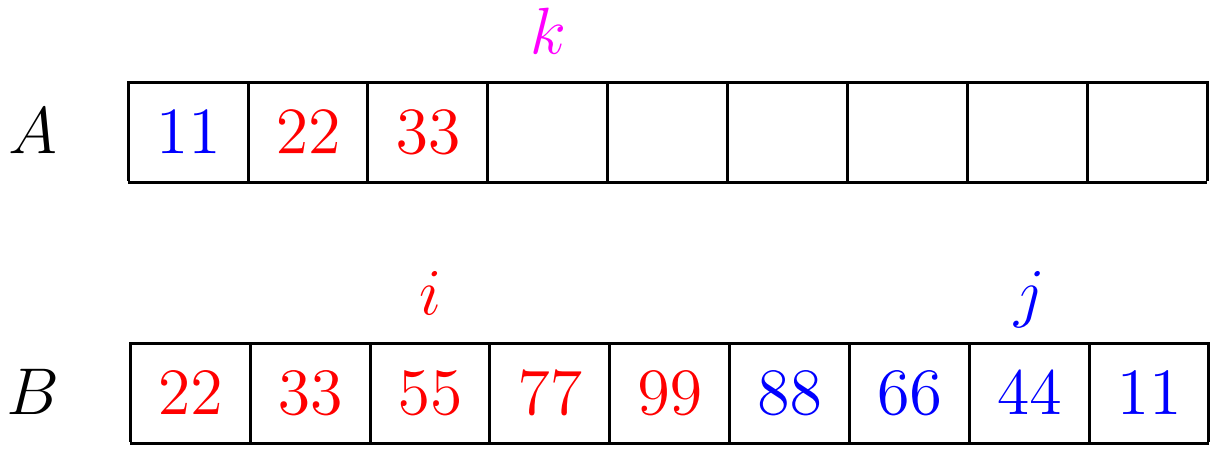
Intercalação



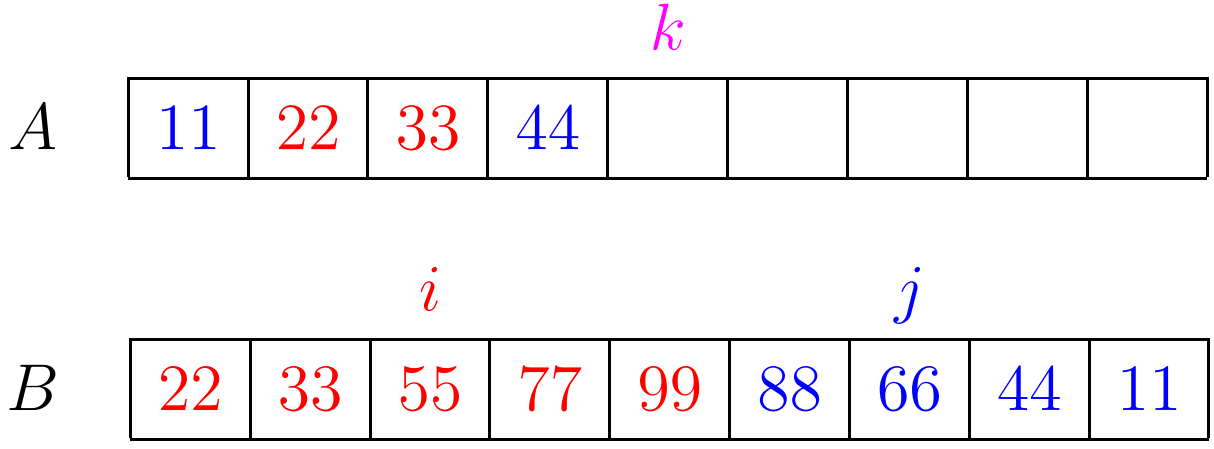
Intercalação



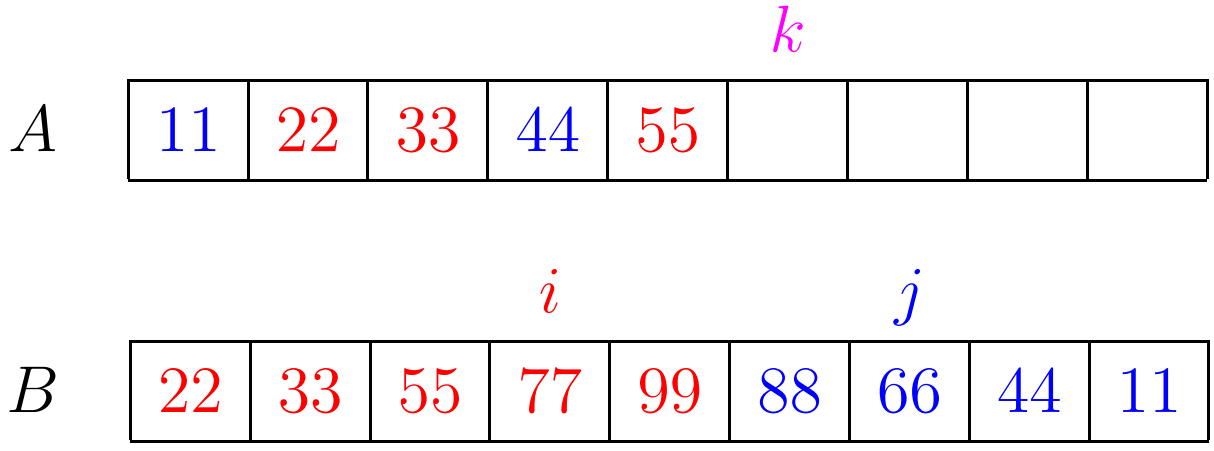
Intercalação



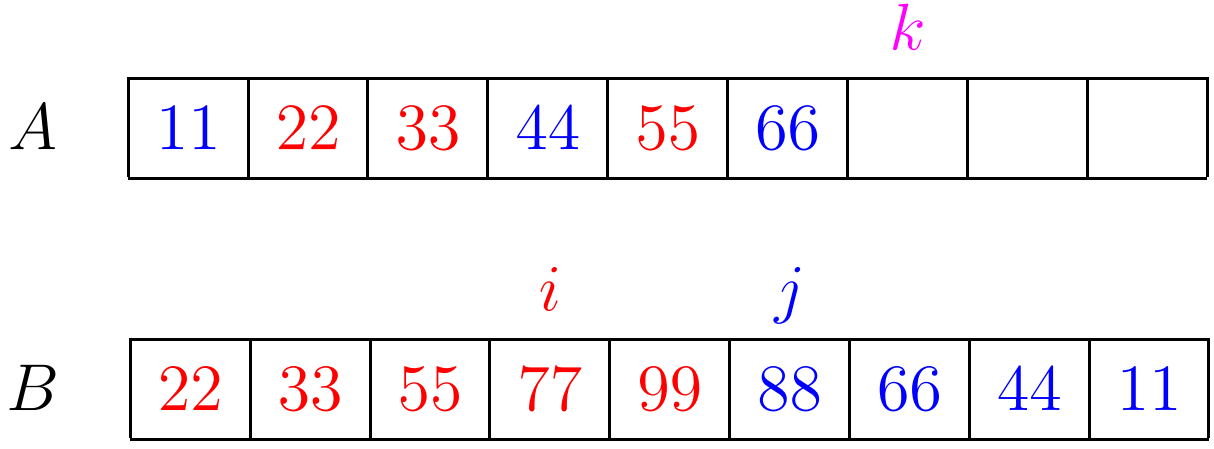
Intercalação



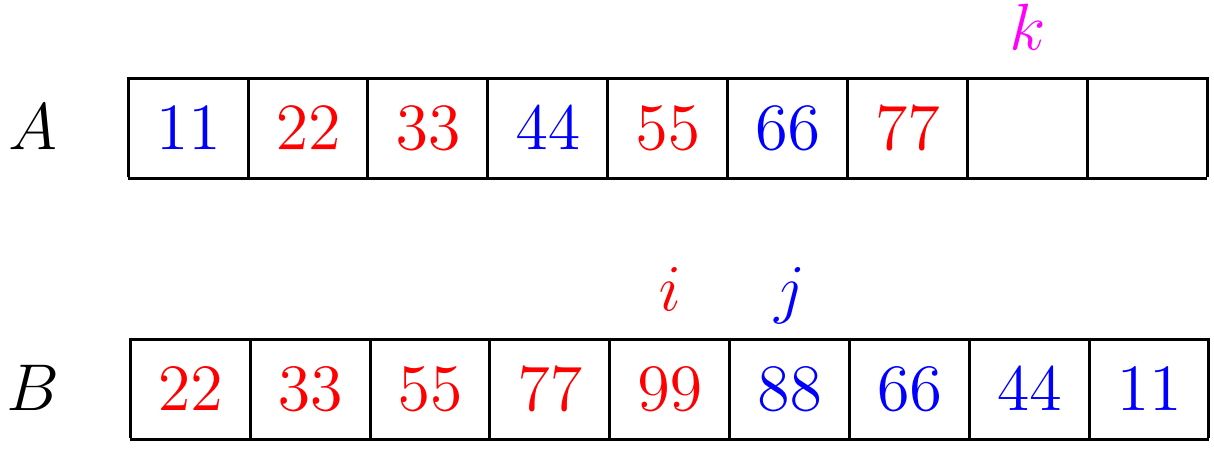
Intercalação



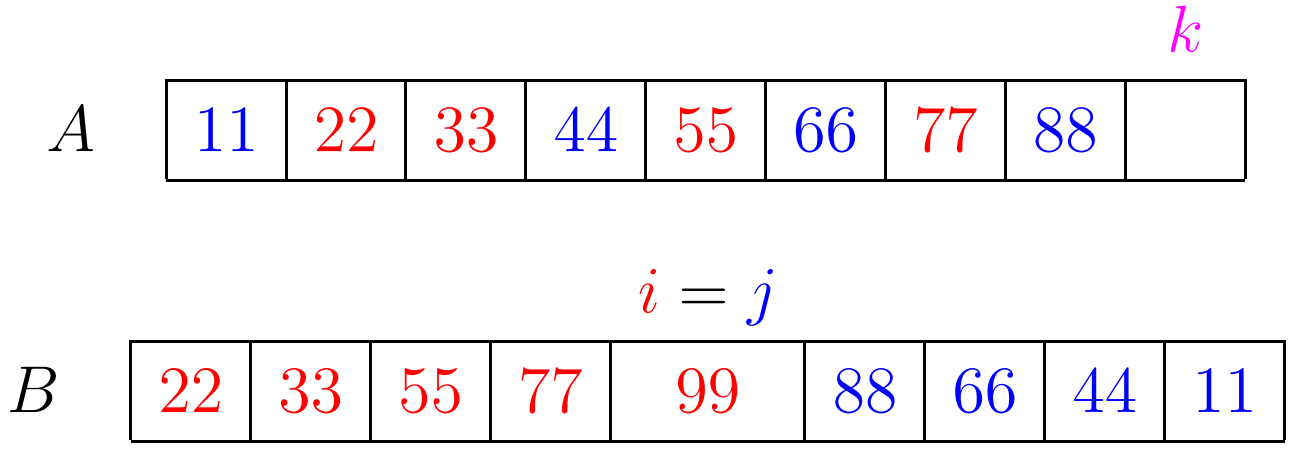
Intercalação



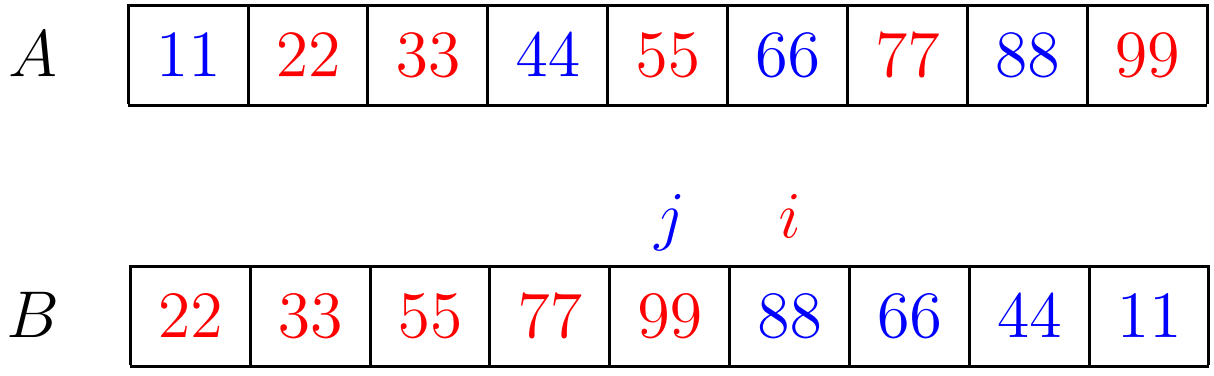
Intercalação



Intercalação



Intercalação



Intercalação

```
INTERCALA (A, p, q, r)
 0   ▷ B[p..r] é um vetor auxiliar
 1   para i ← p até q faça
 2         B[i] ← A[i]
 3   para j ← q + 1 até r faça
 4         B[r + q + 1 − j] ← A[j]
 5   i ← p
 6   j ← r
 7   para k ← p até r faça
 8         se B[i] ≤ B[j]
 9               então A[k] ← B[i]
10                     i ← i + 1
11               senão A[k] ← B[j]
12                     j ← j − 1
```

Consumo de tempo

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

linha	todas as execuções da linha
1	?
2	?
3	?
4	?
5	?
6	?
7	?
8	?
9–12	?
total	?

Consumo de tempo

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

linha		todas as execuções da linha
1	$=$	$q - p + 2 \;=\; n - r + q + 1$
2	$=$	$q - p + 1 \;=\; n - r + q$
3	$=$	$r - (q + 1) + 2 \;=\; n - q + p$
4	$=$	$r - (q + 1) + 1 \;=\; n - q + p - 1$
5	$=$	1
6	$=$	1
7	$=$	$r - p + 2 \;=\; n + 1$
8	$=$	$r - p + 1 \;=\; n$
9–12	$=$	$2\,(r - p + 1) \;=\; 2n$
total	$=$	$8n - 2(r - p + 1) + 5 = \; 6n + 5$

Consumo de tempo em O

Quanto tempo consome em função de $n := r - p + 1$?

linha	consumo de todas as execuções da linha
1–4	?
5–6	?
7	?
8	?
9–12	?
total	?

Consumo de tempo

Quanto tempo consome em função de $n := r - p + 1$?

linha	consumo de todas as execuções da linha
1–4	$\mathrm{O}(n)$
5–6	$\mathrm{O}(1)$
7	$n\mathrm{O}(1) = \mathrm{O}(n)$
8	$n\mathrm{O}(1) = \mathrm{O}(n)$
9–12	$n\mathrm{O}(1) = \mathrm{O}(n)$
total	$\mathrm{O}(4n + 1) = \mathrm{O}(n)$

Consumo de tempo

Quanto tempo consome em função de $n := r - p + 1$?

linha	consumo de todas as execuções da linha
1–4	$\mathrm{O}(n)$
5–6	$\mathrm{O}(1)$
7	$n\mathrm{O}(1) = \mathrm{O}(n)$
8	$n\mathrm{O}(1) = \mathrm{O}(n)$
9–12	$n\mathrm{O}(1) = \mathrm{O}(n)$
total	$\mathrm{O}(4n + 1)$ = $\mathrm{O}(n)$

Conclusão:

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

Exercício

Exercício 5.A

Analise a correção e o consumo de tempo do seguinte algoritmo:

```
EXERC (A, p, r)
1   q ← ⌊(p + r)/2⌋
2   ORDENA-POR-INSERÇÃO (A, p, q)
3   ORDENA-POR-INSERÇÃO (A, q + 1, r)
4   INTERCALA (A, p, q, r)
```

Irmãos de O

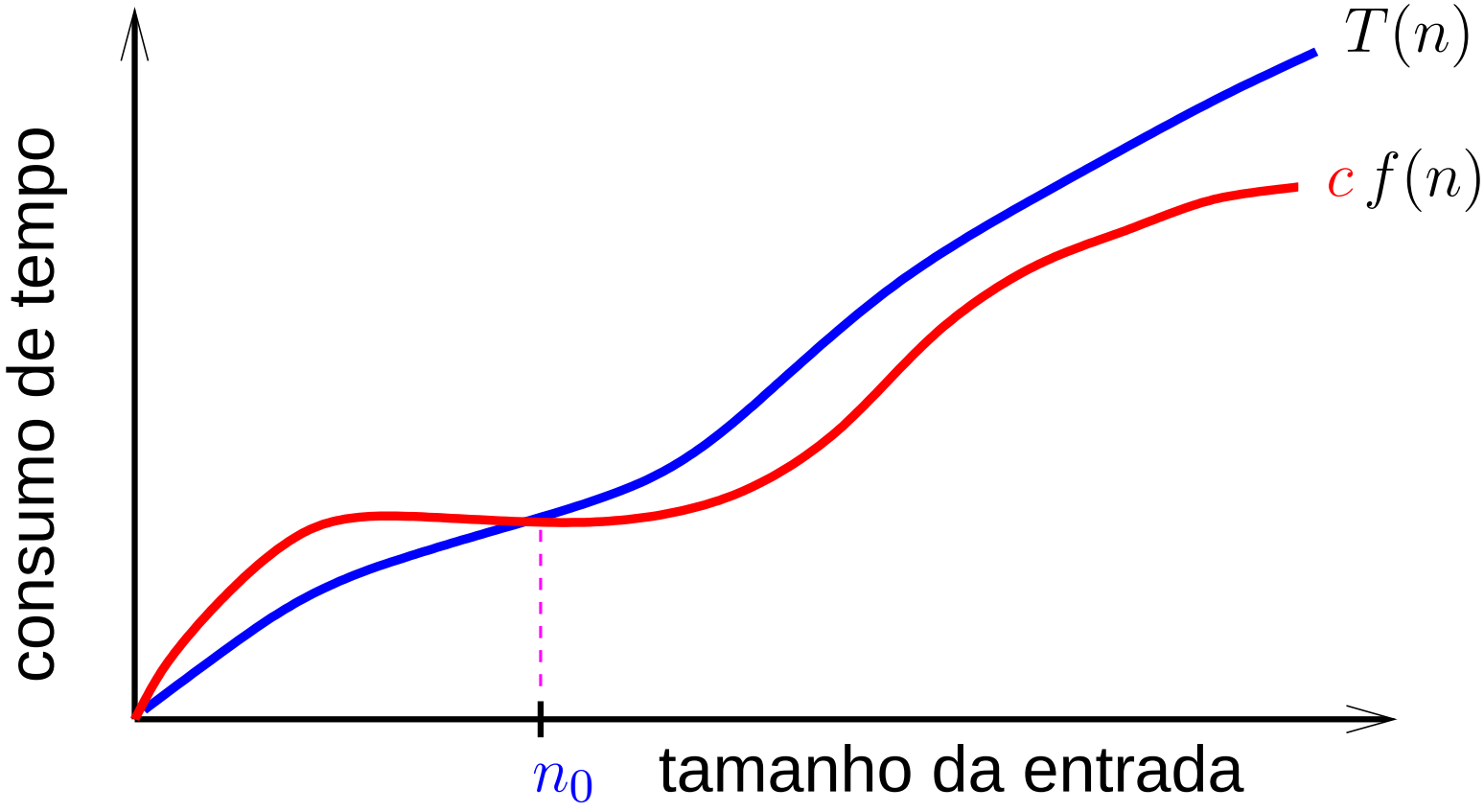
CLRS 3.1

Definição

Dizemos que $T(n)$ é $\Omega(f(n))$ se existem constantes positivas c e n_0 tais que

$$c\,f(n) \leq T(n)$$

para todo $n \geq n_0$.

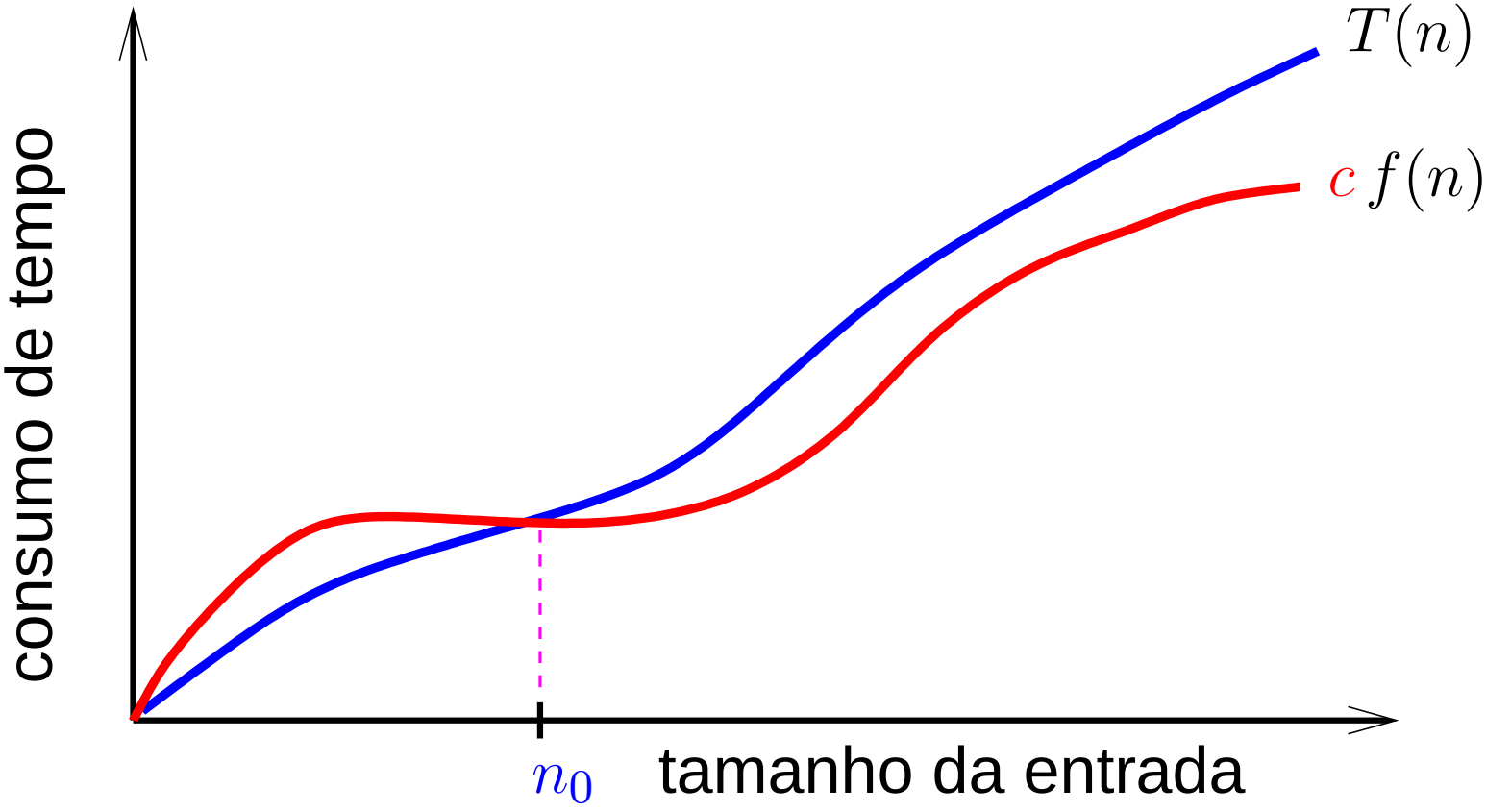


Mais informal

$T(n) = \Omega(f(n))$ se existe $c > 0$ tal que

$$c\, f(n) \leq T(n)$$

para todo n suficientemente GRANDE.



Exemplos

Exemplo 1

Se $T(n) \geq 0.001n^2$ para todo $n \geq 8$, então $T(n)$ é $\Omega(n^2)$.

Exemplos

Exemplo 1

Se $T(n) \geq 0.001n^2$ para todo $n \geq 8$, então $T(n)$ é $\Omega(n^2)$.

Prova: Aplique a definição com $c = 0.001$ e $n_0 = 8$.

Exemplo 2

O consumo de tempo do INTERCALA é $\mathrm{O}(n)$ e também $\Omega(n)$.

Exemplo 2

O consumo de tempo do INTERCALA é $\mathrm{O}(n)$ e também $\Omega(n)$.

linha	todas as execuções da linha
1	$= \quad q - p + 2 \;=\; n - r + q + 1$
2	$= \quad q - p + 1 \;=\; n - r + q$
3	$= \quad r - (q + 1) + 2 \;=\; n - q + p$
4	$= \quad r - (q + 1) + 1 \;=\; n - q + p - 1$
5	$= \quad 1$
6	$= \quad 1$
7	$= \quad r - p + 2 \;=\; n + 1$
8	$= \quad r - p + 1 \;=\; n$
9–12	$= \quad 2\,(r - p + 1) \;=\; 2n$
total	$= \quad 8n - 2(r - p + 1) + 5 = \; 6n + 5$

Exemplo 3

Se $T(n)$ é $\Omega(f(n))$, então $f(n)$ é $\mathrm{O}(T(n))$.

Exemplo 3

Se $T(n)$ é $\Omega(f(n))$, então $f(n)$ é $\mathrm{O}(T(n))$.

Prova: Se $T(n)$ é $\Omega(f(n))$, então existem constantes positivas c e n_0 tais que

$$c\,f(n) \leq T(n)$$

para todo $n \geq n_0$. Logo,

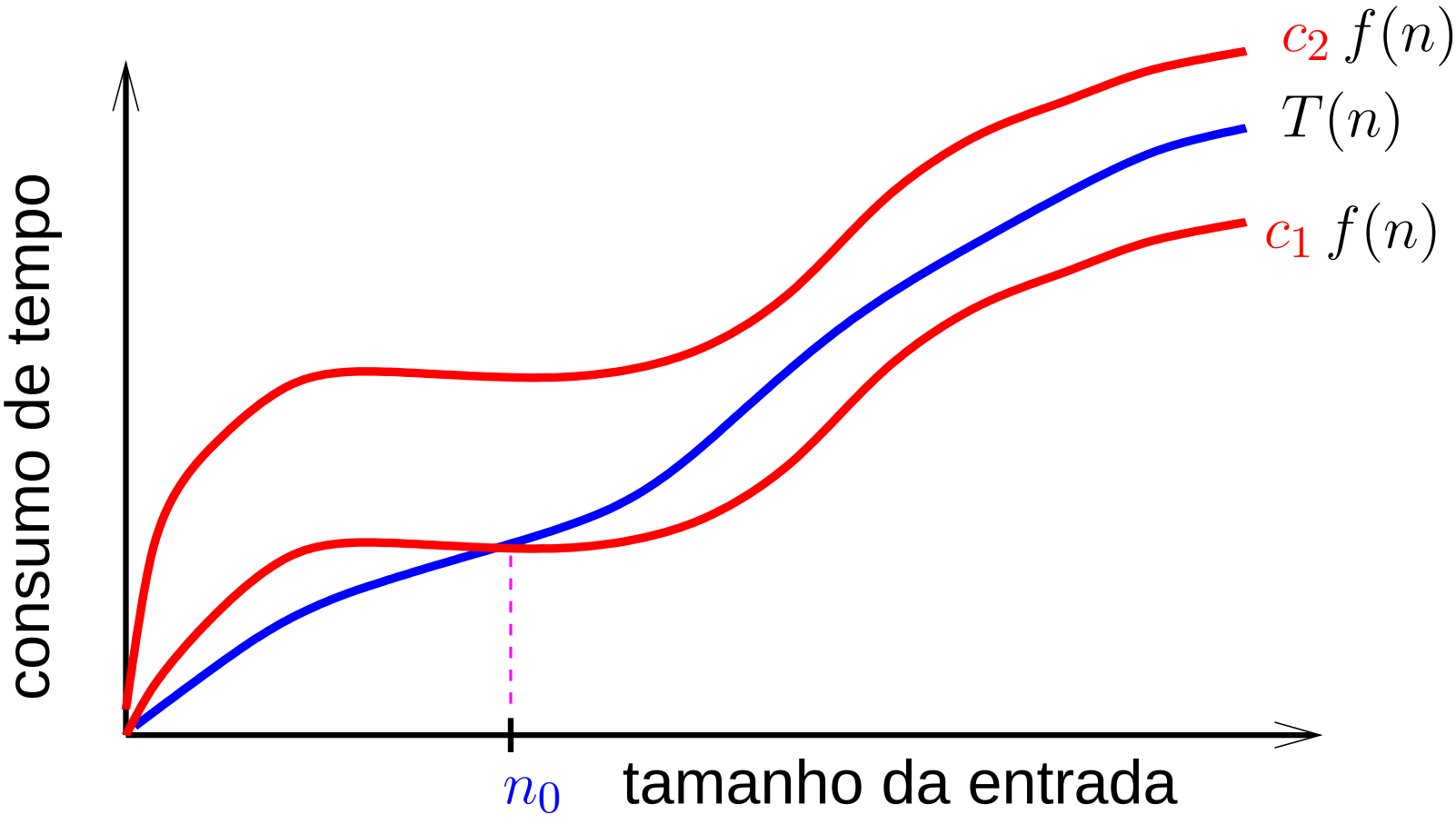
$$f(n) \leq 1/c\,T(n)$$

para todo $n \geq n_0$. Portanto, $f(n)$ é $\mathrm{O}(T(n))$.

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros no reais.
Dizemos que $T(n)$ é $\Theta(f(n))$ se

$T(n)$ é $\mathrm{O}(f(n))$ e $T(n)$ é $\Omega(f(n))$.

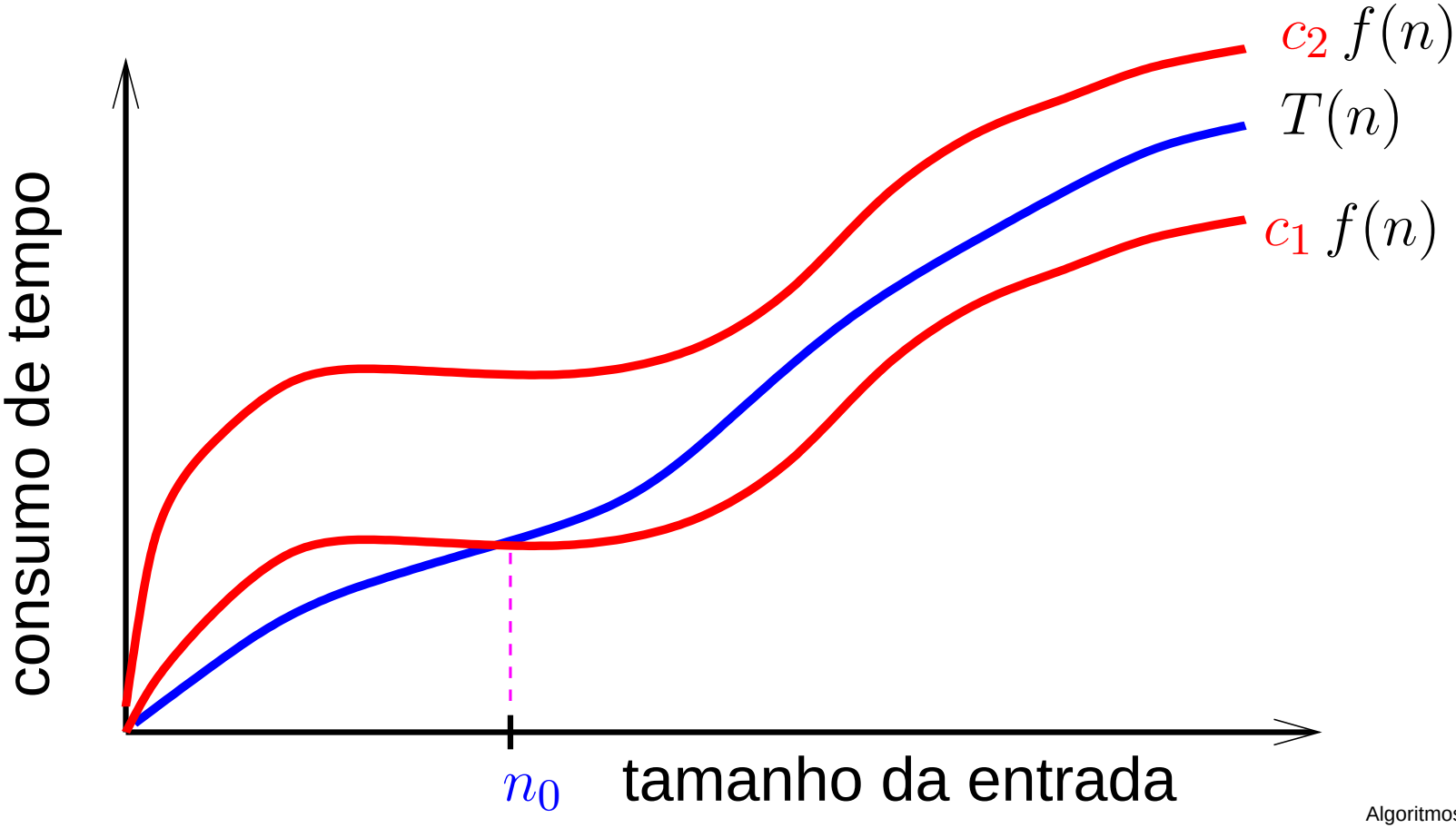


Definição

Dizemos que $T(n)$ é $\Theta(f(n))$ se se existem constantes positivas c_1, c_2 e n_0 tais que

$$c_1\, f(n) \;\leq\; T(n) \;\leq\; c_2\, f(n)$$

para todo $n \geq n_0$.



Tamanho máximo de problemas

Suponha que cada operação consome 1 microsegundo ($1\mu s$).

consumo de tempo(μs)	Tamanho máximo de problemas (n)		
	1 segundo	1 minuto	1 hora
$400n$	2500	150000	9000000
$20\,n\,\lceil \lg n \rceil$	4096	166666	7826087
$2n^2$	707	5477	42426
n^4	31	88	244
2^n	19	25	31

Michael T. Goodrich e Roberto Tamassia, *Projeto de Algoritmos*, Bookman.

Nomes de classes Θ

classe	nome
$\Theta(1)$	constante
$\Theta(\log n)$	logarítmica
$\Theta(n)$	linear
$\Theta(n \log n)$	$n \log n$
$\Theta(n^2)$	quadrática
$\Theta(n^3)$	cúbica
$\Theta(n^k)$ com $k >= 1$	polinomial
$\Theta(2^n)$	exponencial
$\Theta(a^n)$ com $a > 1$	exponencial

Palavras de Cautela

Suponha que A e B são algoritmos para um mesmo problema. Suponha que o consumo de tempo de A é "essencialmente" $100\,n$ e que o consumo de tempo de B é "essencialmente" $n \log_{10} n$.

Palavras de Cautela

Suponha que A e B são algoritmos para um mesmo problema. Suponha que o consumo de tempo de A é "essencialmente" $100\,n$ e que o consumo de tempo de B é "essencialmente" $n \log_{10} n$.

$100\,n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.
Logo, A é assintoticamente mais eficiente que B.

Palavras de Cautela

Suponha que A e B são algoritmos para um mesmo problema. Suponha que o consumo de tempo de A é "essencialmente" $100\,n$ e que o consumo de tempo de B é "essencialmente" $n \log_{10} n$.

$100\,n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.
Logo, A é assintoticamente mais eficiente que B.

A é mais eficiente que B para $n \geq 10^{100}$.
$10^{100} =$ um *googol*
$\approx$ número de átomos no universo observável
$=$ número ENORME

Palavras de Cautela

Conclusão:

> Lembre das constantes e termos de baixa ordem que estão "escondidos" na notação assintótica.

Em geral um algoritmo que consome tempo $\Theta(n \lg n)$, e com fatores constantes razoáveis, é bem eficiente.

Um algoritmo que consome tempo $\Theta(n^2)$ pode, algumas vezes ser satisfatório.

Um algoritmo que consome tempo $\Theta(2^n)$ é dificilmente aceitável.

Do ponto de vista de AA, eficiente $=$ polinomial.

Exercícios

Exercício 6.A
Já sabemos que ORDENA-POR-INSERÇÃO é $\mathrm{O}(n^2)$.
Mostre que o algoritmo é $\Omega(n)$.

Exercício 6.B
Mostre que ORDENA-POR-INSERÇÃO é $\Omega(n^2)$
no pior caso.

Exercício 6.C
Mostre que ORDENA-POR-INSERÇÃO é $\mathrm{O}(n)$
no melhor caso.

Mais exercícios

Exercício 6.D

Prove que $n^2 + 10n + 20 = \Omega(n^2)$. Prove que $n^2 - 10n - 20 = \Theta(n^2)$.

Exercício 6.E

Prove que $n = \Omega(\lg n)$.

Exercício 6.F

Prove que $\lg n = \Theta(\log_{10} n)$.

Exercício 6.G

É verdade que $2^n = \Omega(3^n)$?

Exercício 6.H

É verdade que $2n^3 + 5\sqrt{n} = \Theta(n^3)$?

Mais exercícios ainda

Exercício 6.I

Suponha que os algoritmos $\mathcal{A}$ e $\mathcal{B}$ só dependem de um parâmetro n. Suponha ainda que $\mathcal{A}$ consome $S(n)$ unidades de tempo enquanto $\mathcal{B}$ consome $T(n)$ unidades de tempo. Quero provar que algoritmo $\mathcal{A}$ é pelo menos tão eficiente quanto o algoritmo $\mathcal{B}$ (no sentido assintótico). Devo mostrar que existe $f(n)$ tal que

$S(n) = \mathrm{O}(f(n))$ e $T(n) = \mathrm{O}(f(n))$?

$S(n) = \mathrm{O}(f(n))$ e $T(n) = \Omega(f(n))$?

$S(n) = \Omega(f(n))$ e $T(n) = \mathrm{O}(f(n))$?

$S(n) = \Omega(f(n))$ e $T(n)\Omega(f(n))$?

Que devo fazer para mostrar que $\mathcal{A}$ é mais eficiente que $\mathcal{B}$?

Exercício 6.J

Mostre que o consumo de tempo do algoritmo INTERCALA é $\Theta(n)$, sendo n o número de elementos do vetor que o algoritmo recebe.

Recursão

CLRS 2.3
AU 2.6, 2.7, 2.9

"To understand recursion, we must first understand recursion."

Recursão

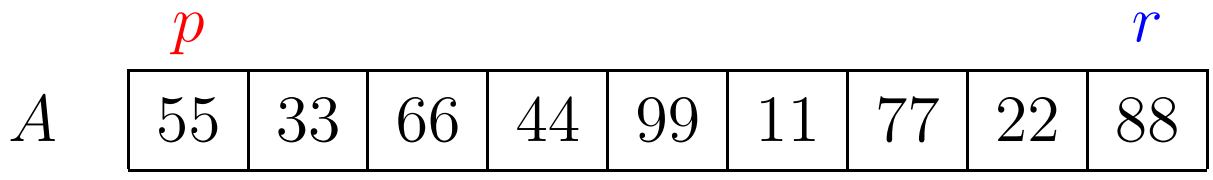
Recursão: resolve um problema a partir das soluções de seus subproblemas

Recursão

Recursão: resolve um problema a partir das soluções de seus subproblemas

Problema: Rearranjar $A[p\,..\,r]$ de modo que ele fique em ordem crescente.

Entra:



Recursão

Recursão: resolve um problema a partir das soluções de seus subproblemas

Problema: Rearranjar $A[p\,..\,r]$ de modo que ele fique em ordem crescente.

Entra:

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

Sai:

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

Divisão-e-conquista

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

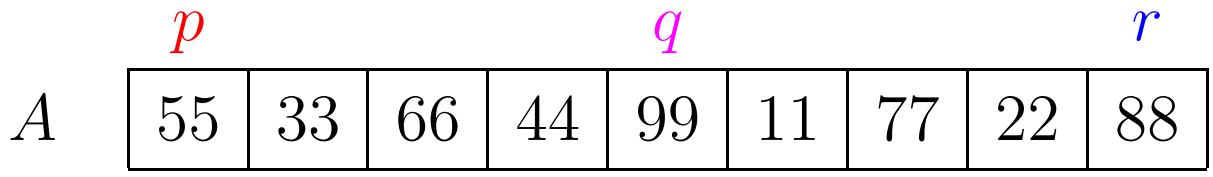
Dividir: o problema é dividido em subproblemas de tamanho menor;

Conquistar: os subproblemas são resolvidos recursivamente e subproblemas "pequenos" são resolvidos diretamente;

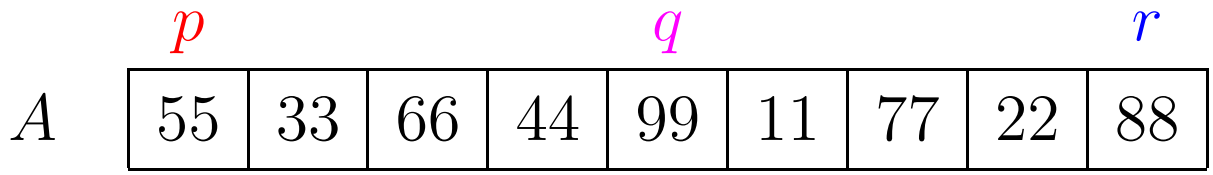
Combinar: as solução dos subproblemas são combinadas para obter uma solução do problema original.

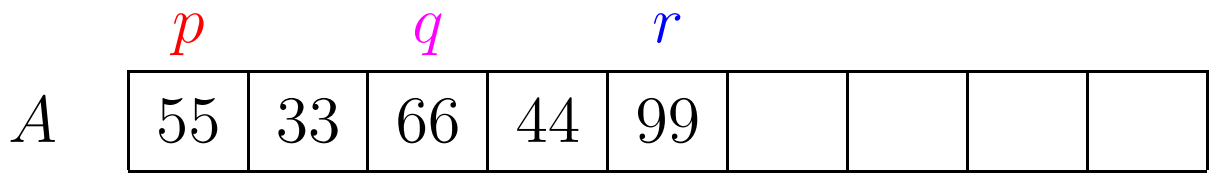
Exemplo: ordenação por intercalação (Merge-sort).

Merge-Sort

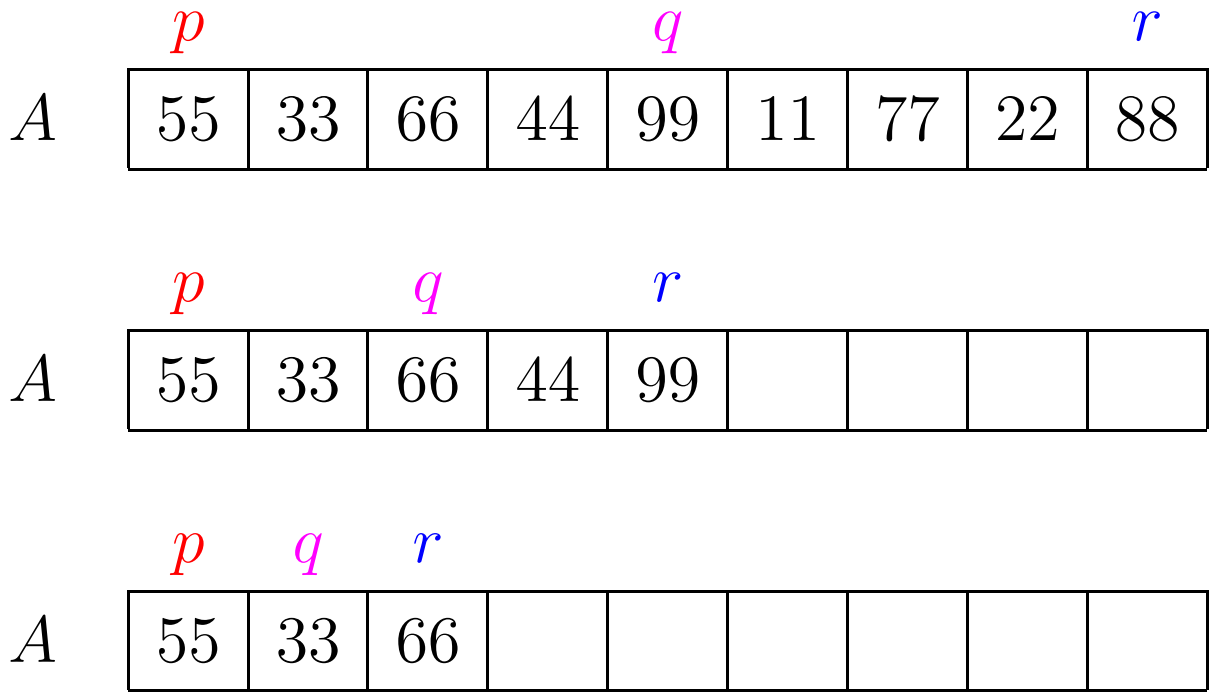


Merge-Sort

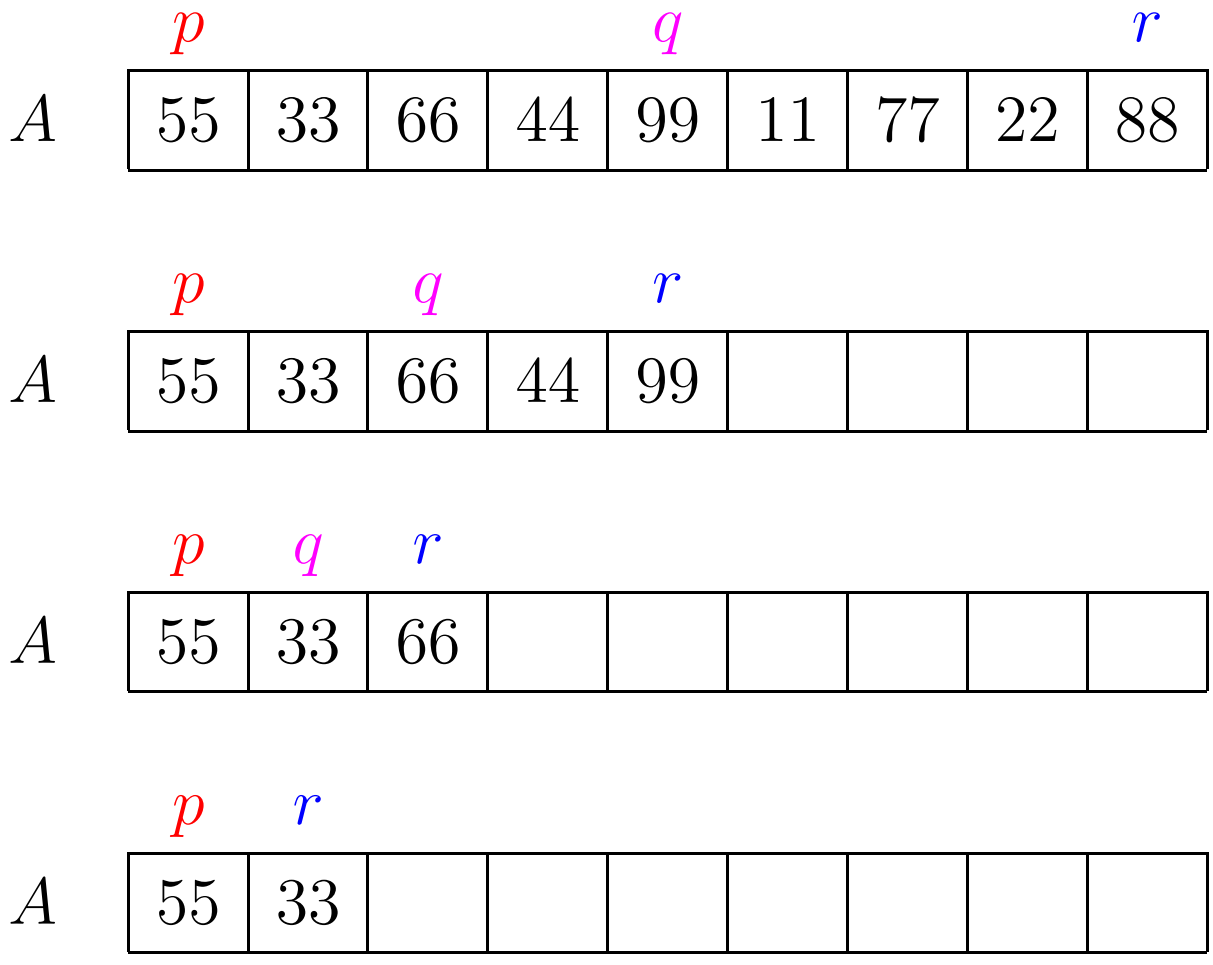




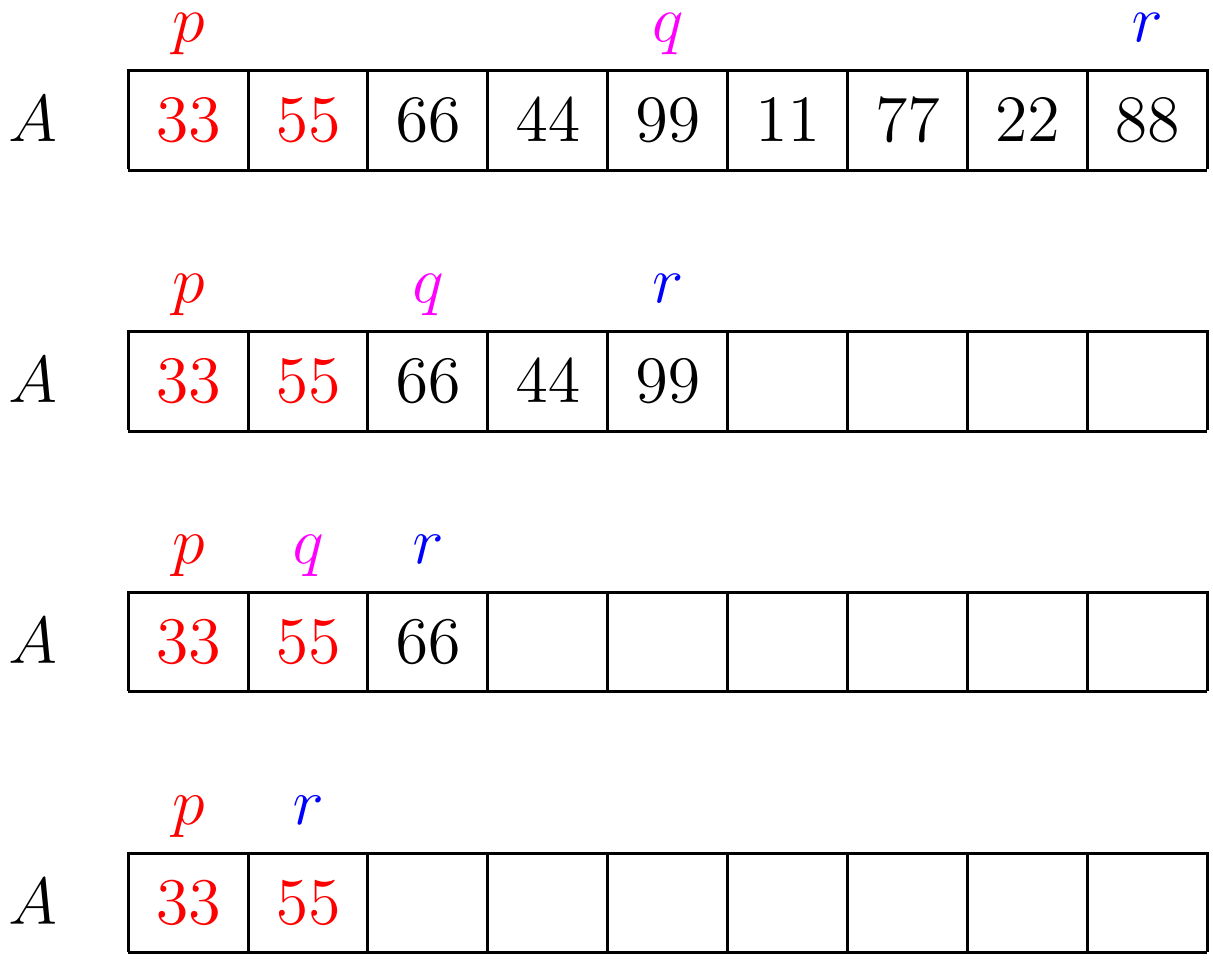
Merge-Sort



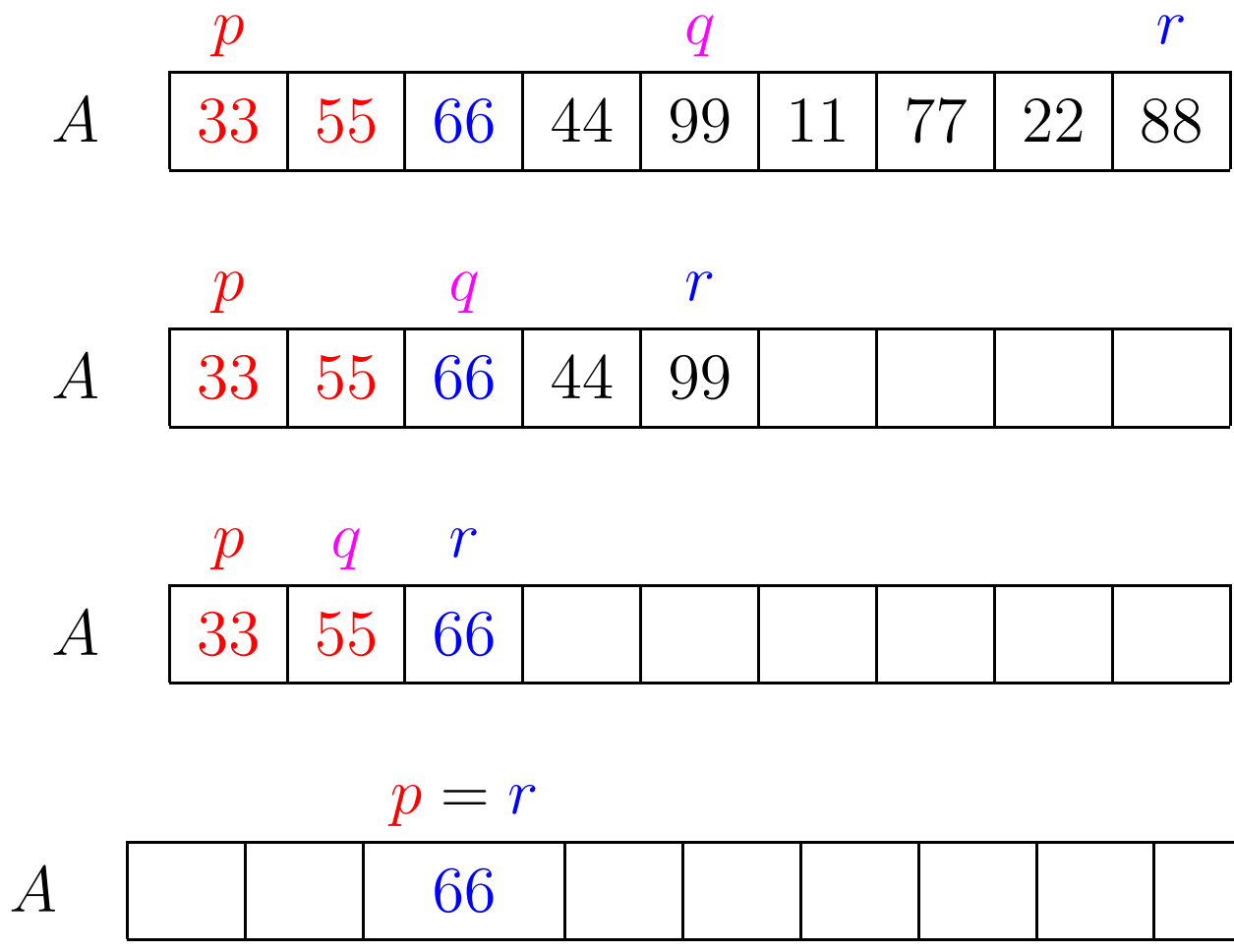
Merge-Sort


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

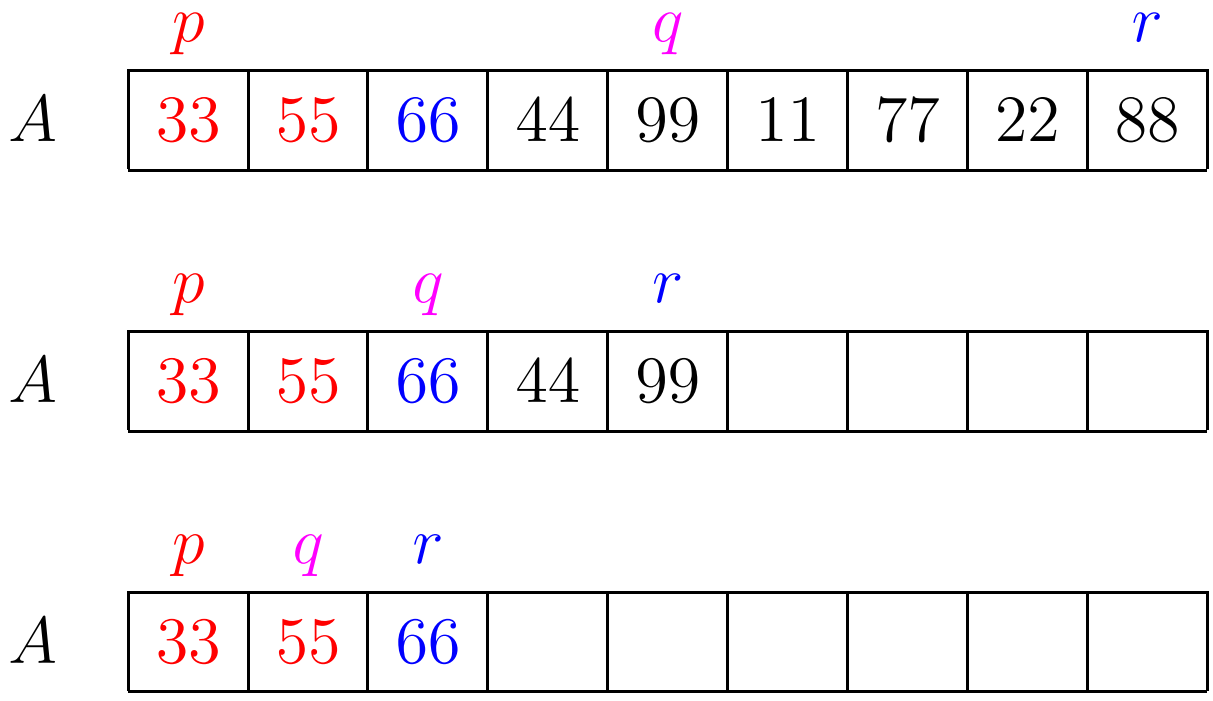
Merge-Sort


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

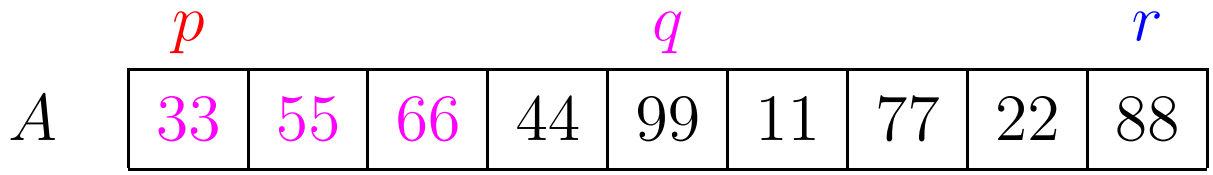
Merge-Sort

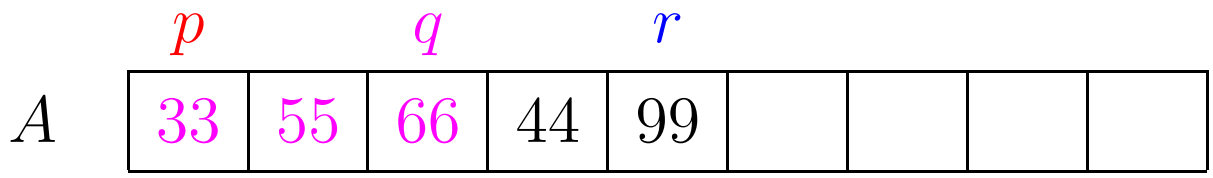

p
q
r
A
33 55 66 44 99 11 77 22 88
p
q
r
A
33 55 66 44 99
p
q
r
A
33 55 66
p = r
A
66

Merge-Sort

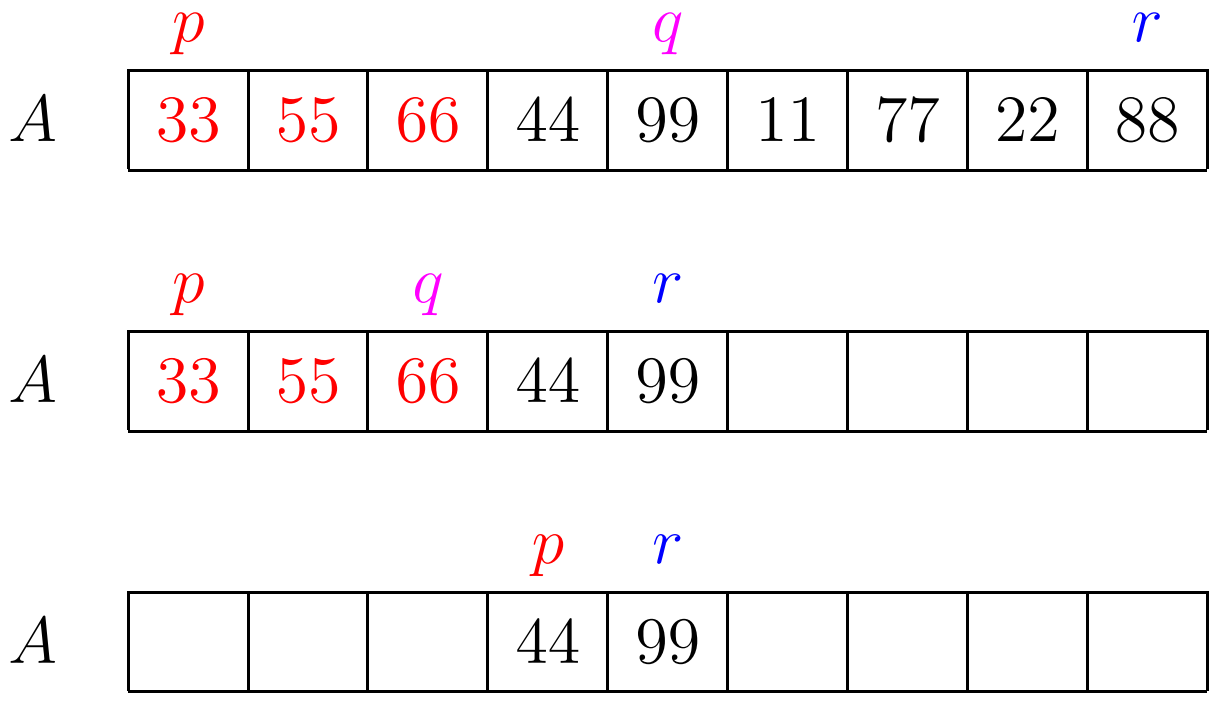


Merge-Sort

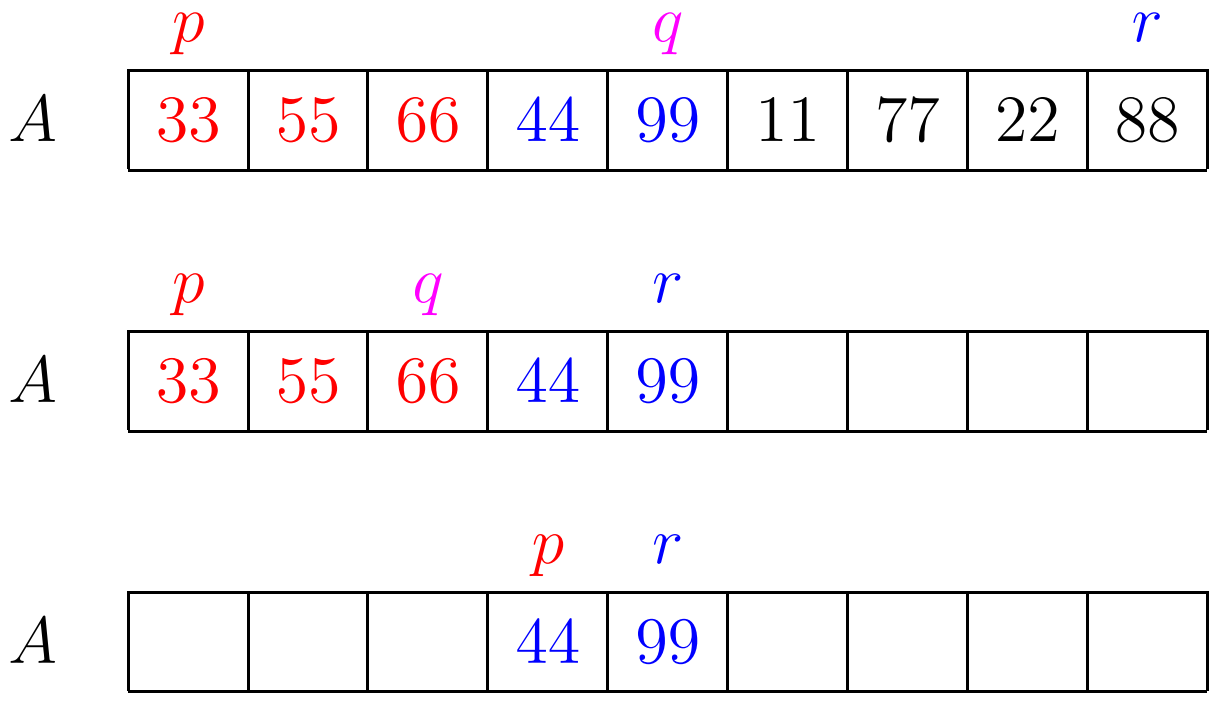




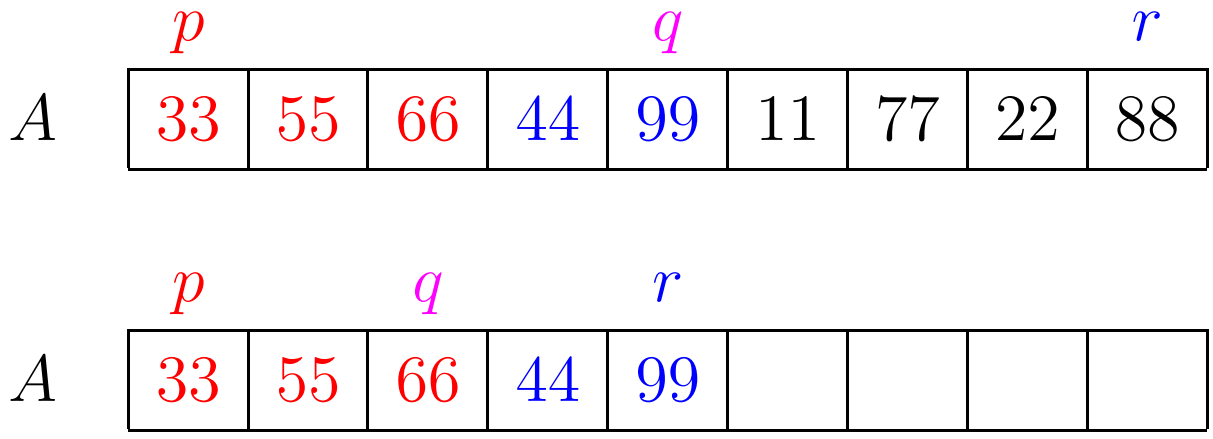
Merge-Sort



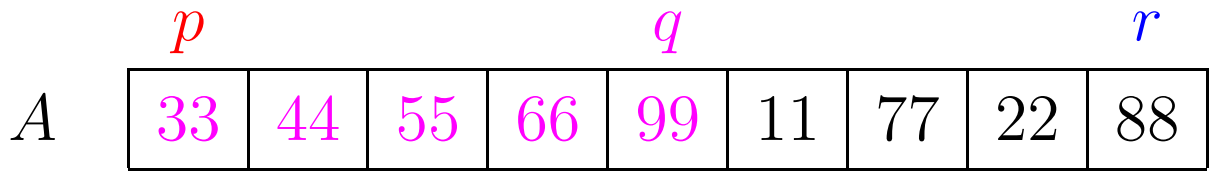
Merge-Sort

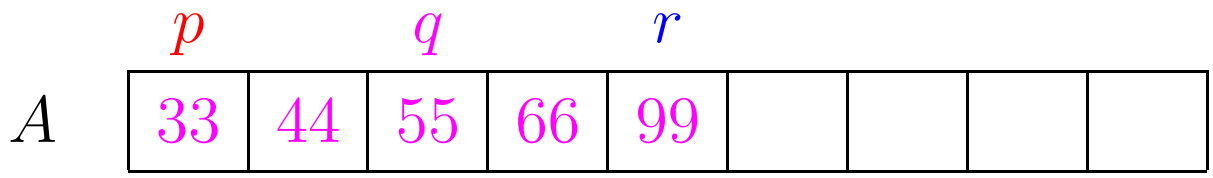


Merge-Sort

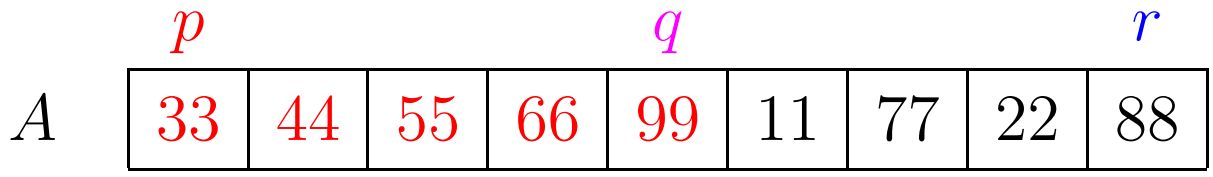


Merge-Sort

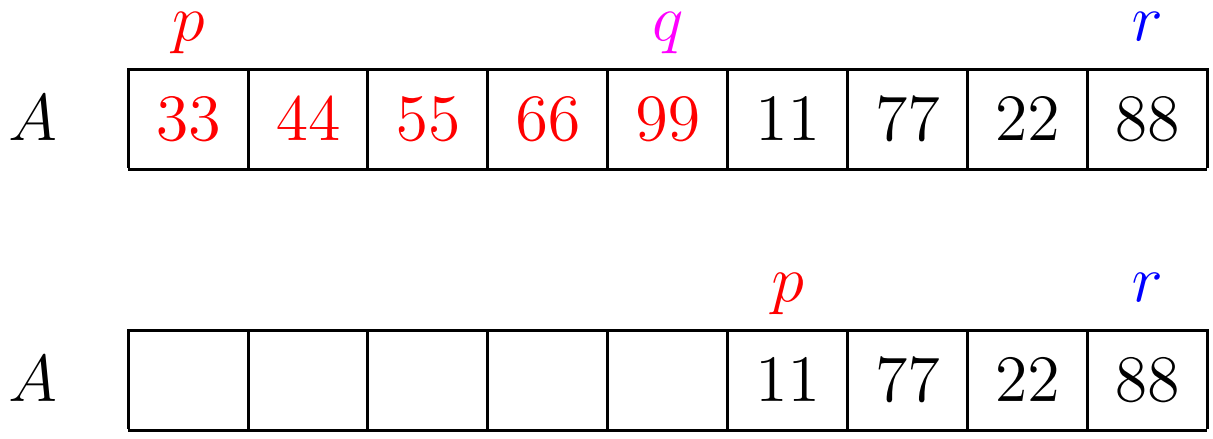




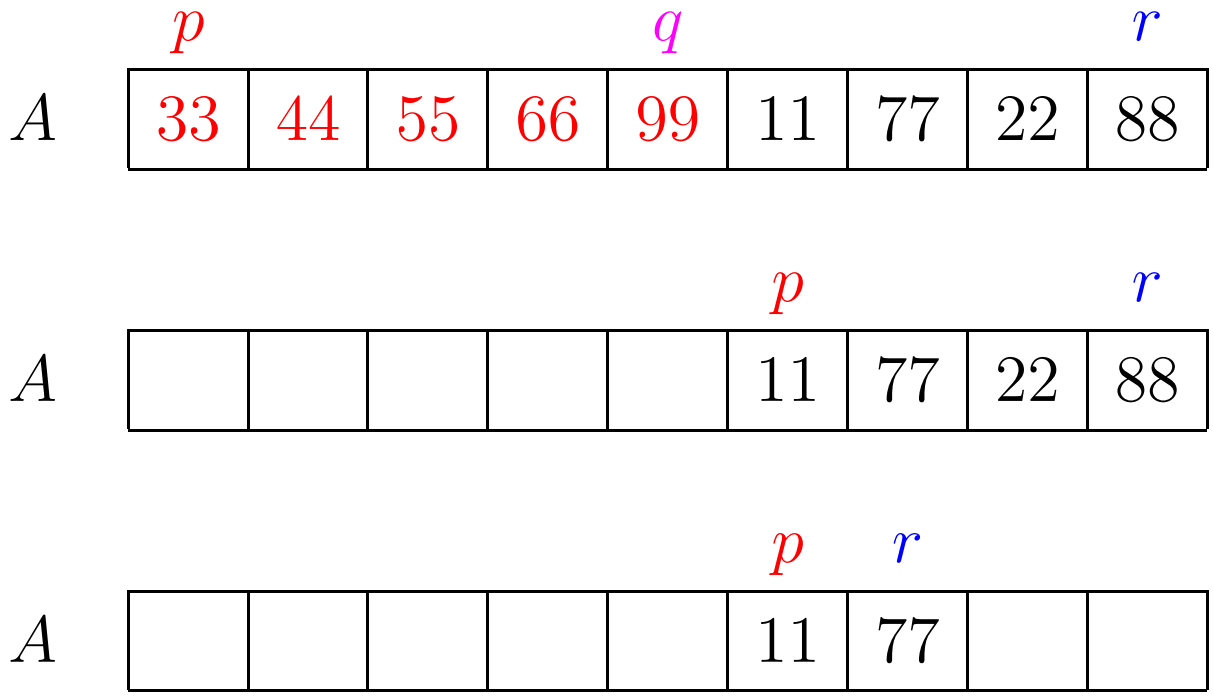
Merge-Sort



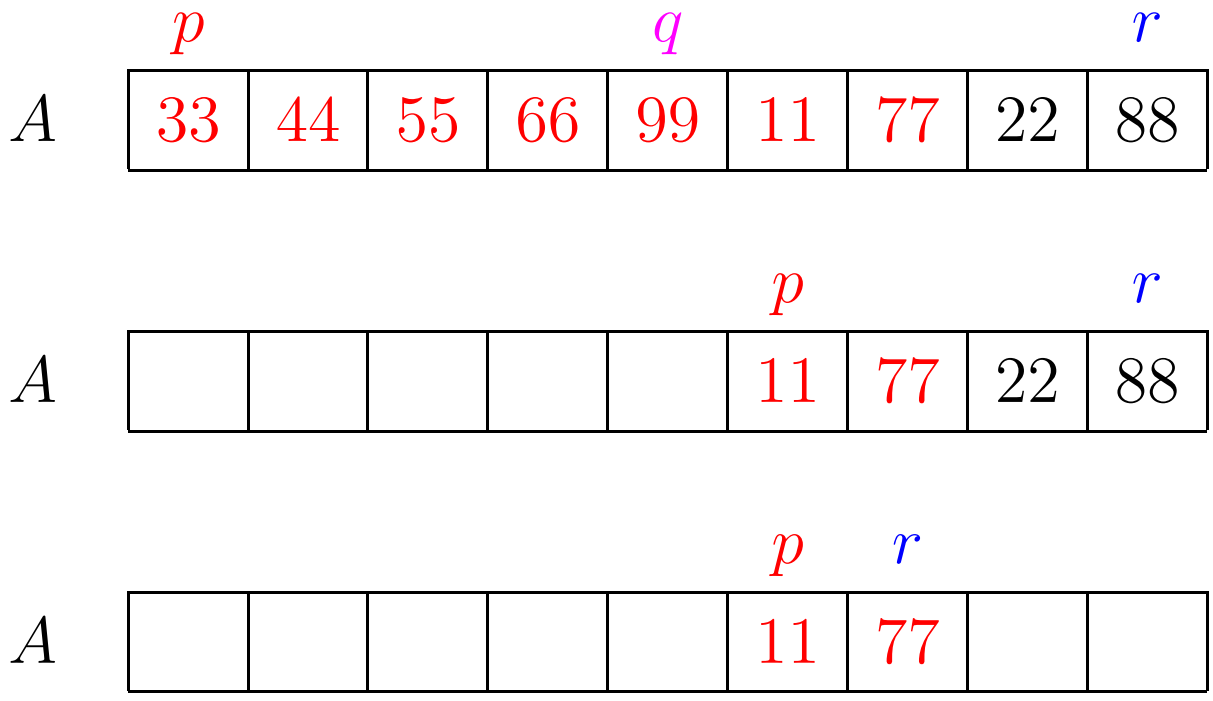
Merge-Sort



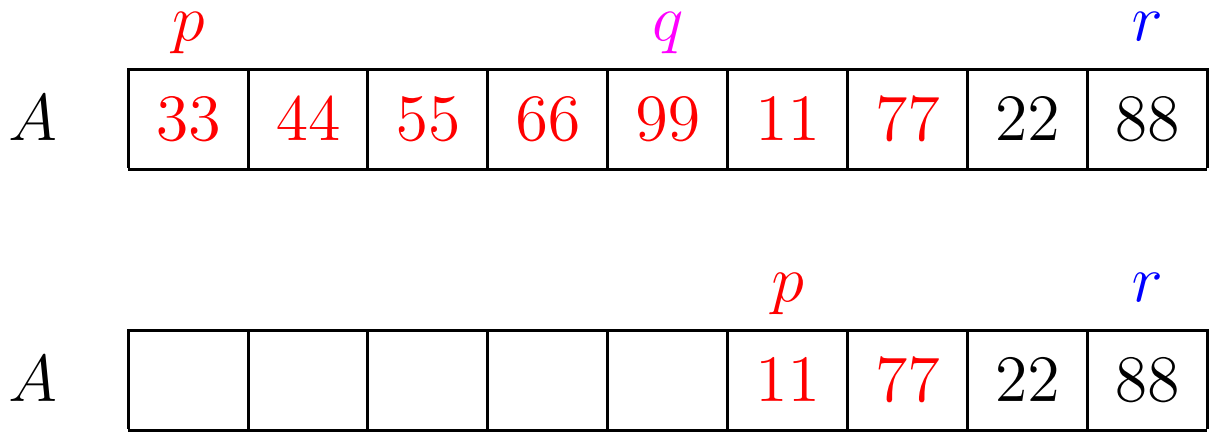
Merge-Sort



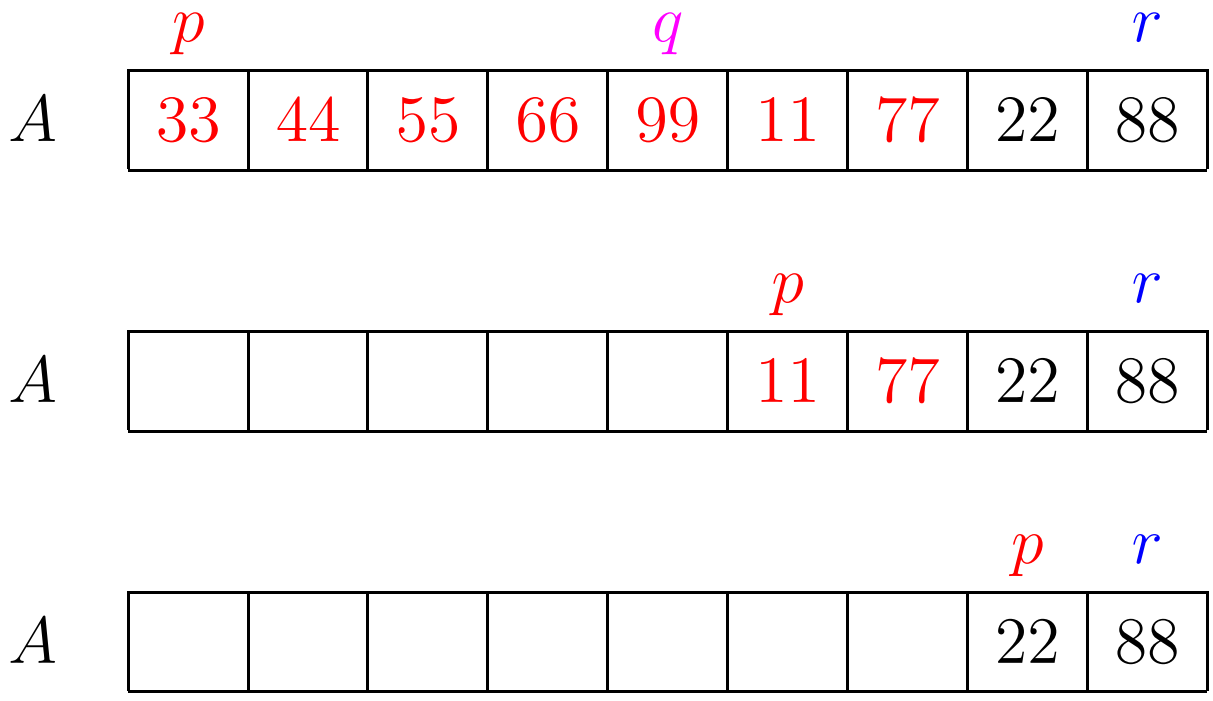
Merge-Sort



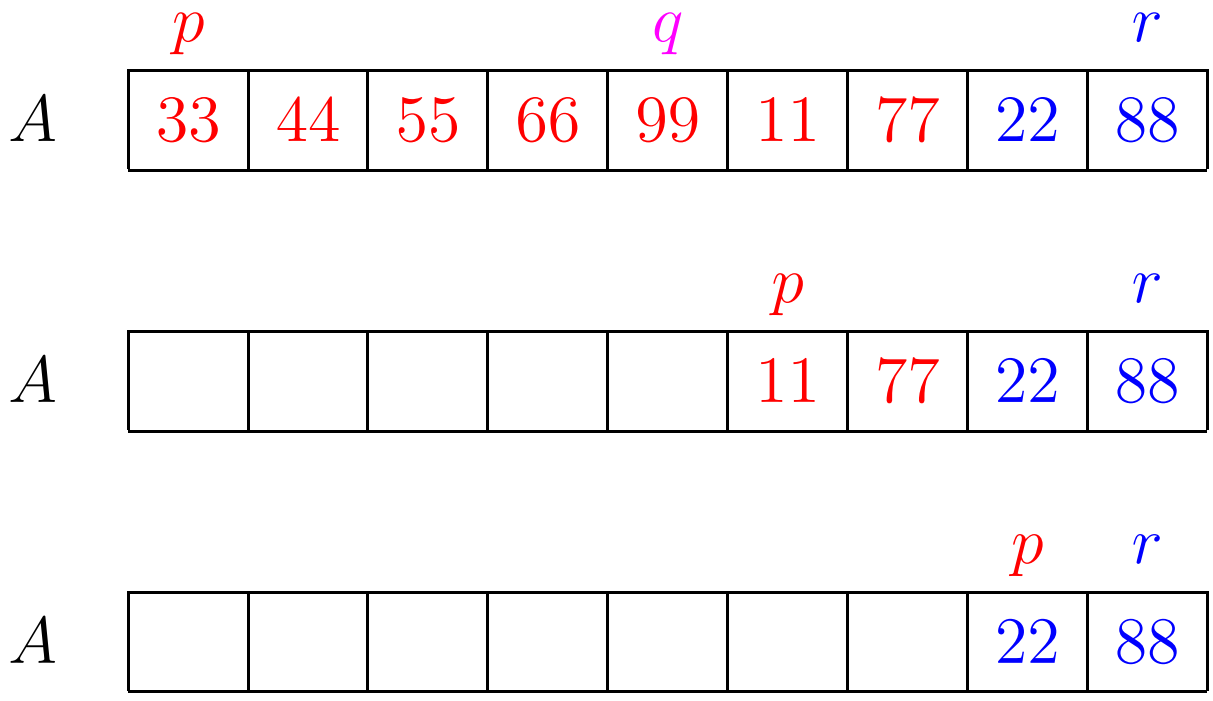
Merge-Sort



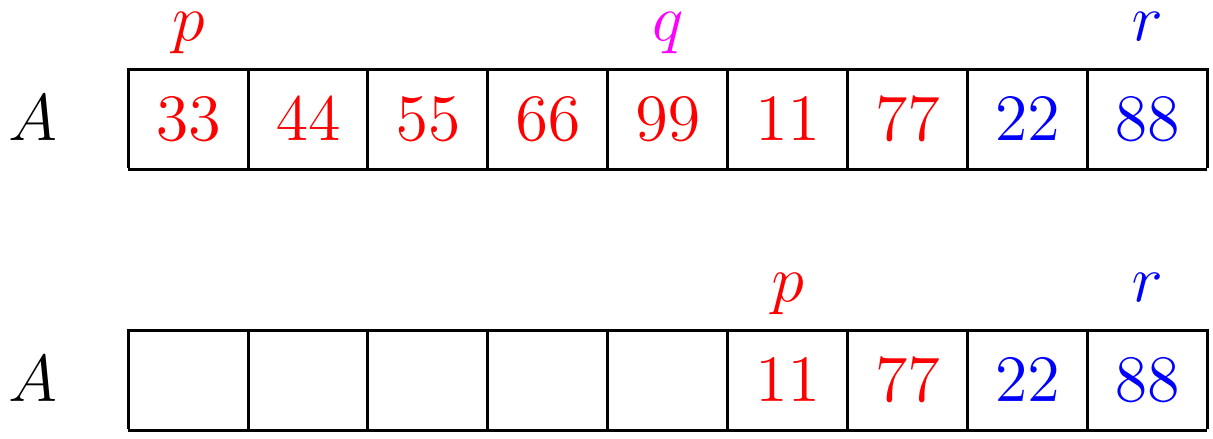
Merge-Sort



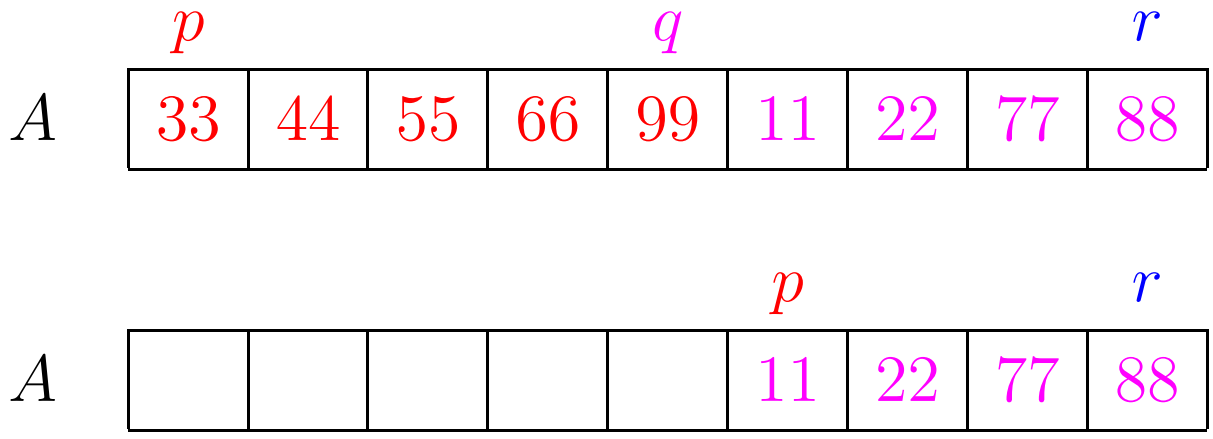
Merge-Sort



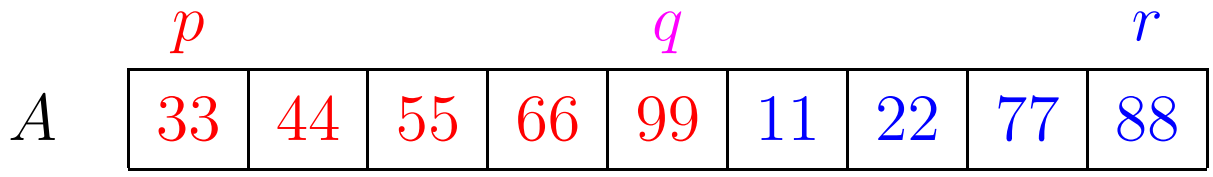
Merge-Sort



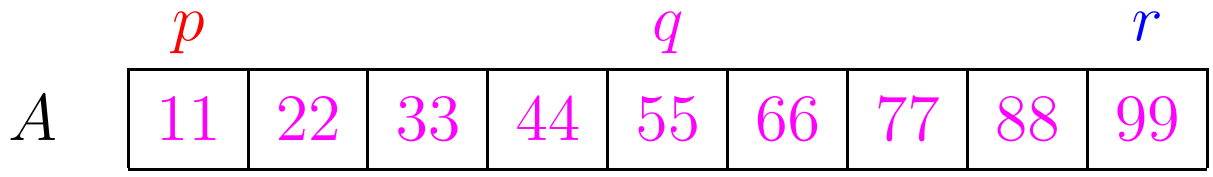
Merge-Sort



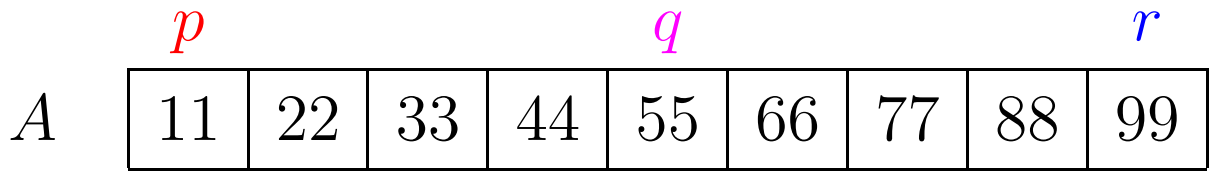
Merge-Sort



Merge-Sort



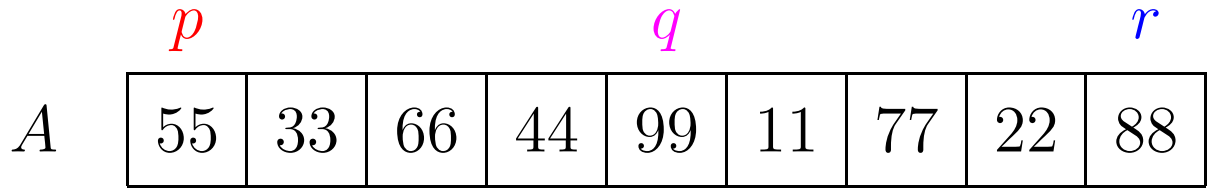
Merge-Sort



Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

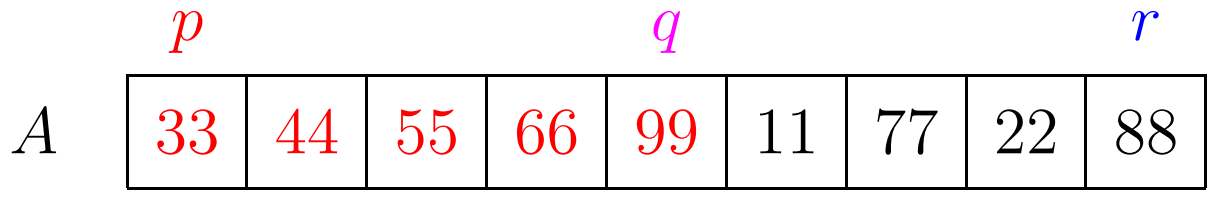
```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```



Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

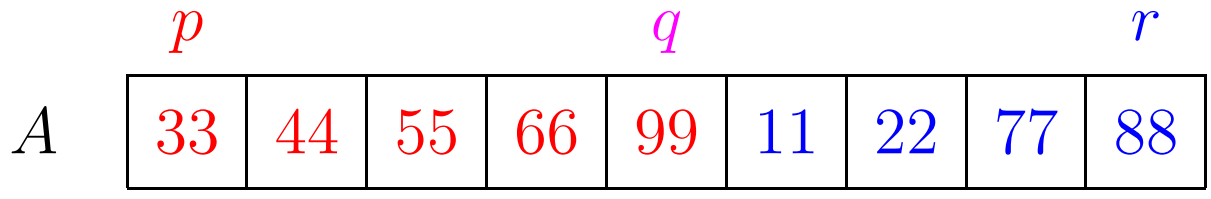
```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```



Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

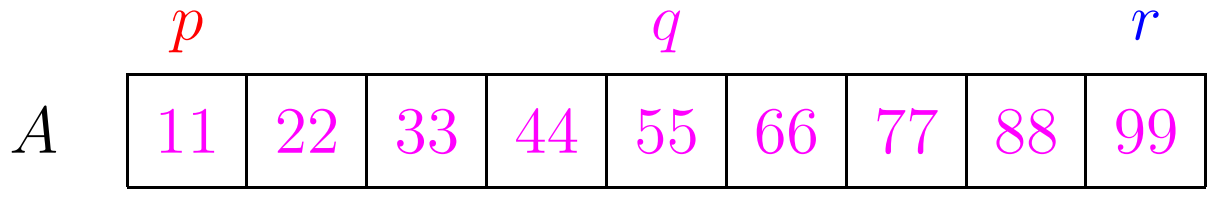
```
MERGE-SORT (A, p, r)
1    se p < r
2         então q ← ⌊(p + r)/2⌋
3               MERGE-SORT (A, p, q)
4               MERGE-SORT (A, q + 1, r)
5               INTERCALA (A, p, q, r)
```



Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
1    se p < r
2         então q ← ⌊(p + r)/2⌋
3               MERGE-SORT (A, p, q)
4               MERGE-SORT (A, q + 1, r)
5               INTERCALA (A, p, q, r)
```



Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```

O algoritmo está correto?

Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```

O algoritmo está correto?
A correção do algoritmo, que se apóia na correção do INTERCALA, pode ser demonstrada por indução em $n := r - p + 1$.

Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```

Consumo de tempo?

Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```

Consumo de tempo?

$T(n) :=$ consumo de tempo máximo quando $n = r - p + 1$

Merge-Sort

```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, r)
5             INTERCALA (A, p, q, r)
```

linha	consumo na linha
1	?
2	?
3	?
4	?
5	?

$T(n) = ?$