Análise de Algoritmos

Slides de Paulo Feofiloff

[com erros do coelho e agora também da cris]

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

crescem todas com a mesma velocidade

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

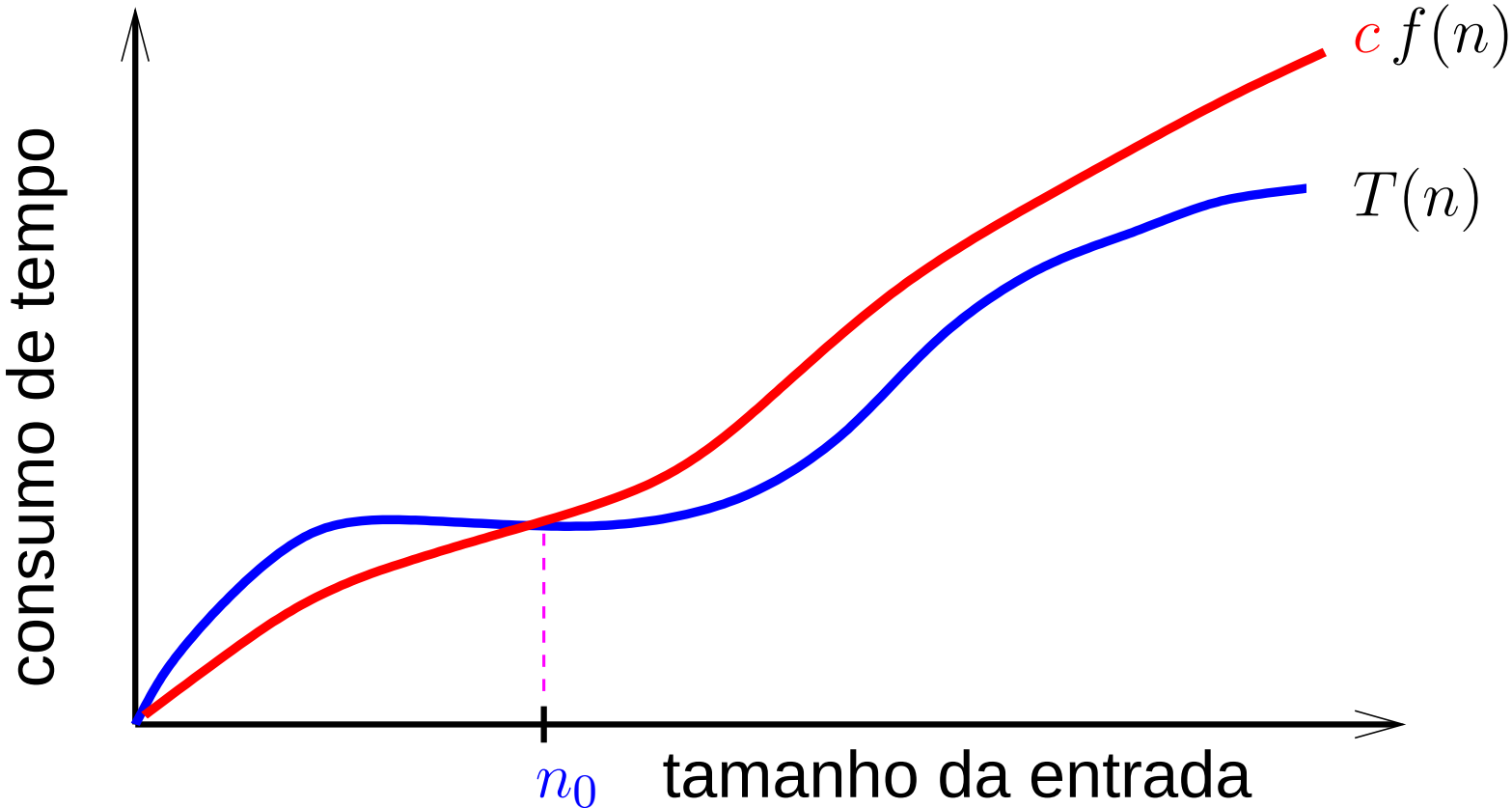
- $n^2 + 99n$ é $\mathrm{O}(n^2)$
- $33n^2$ é $\mathrm{O}(n^2)$
- $9n + 2$ é $\mathrm{O}(n^2)$
- $0{,}00001n^3 - 200n^2$ não é $\mathrm{O}(n^2)$

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros nos reais. Dizemos que $T(n)$ é $\mathrm{O}(f(n))$ se existem constantes positivas c e n_0 tais que

$$0 \leq T(n) \leq c\,f(n)$$

para todo $n \geq n_0$.

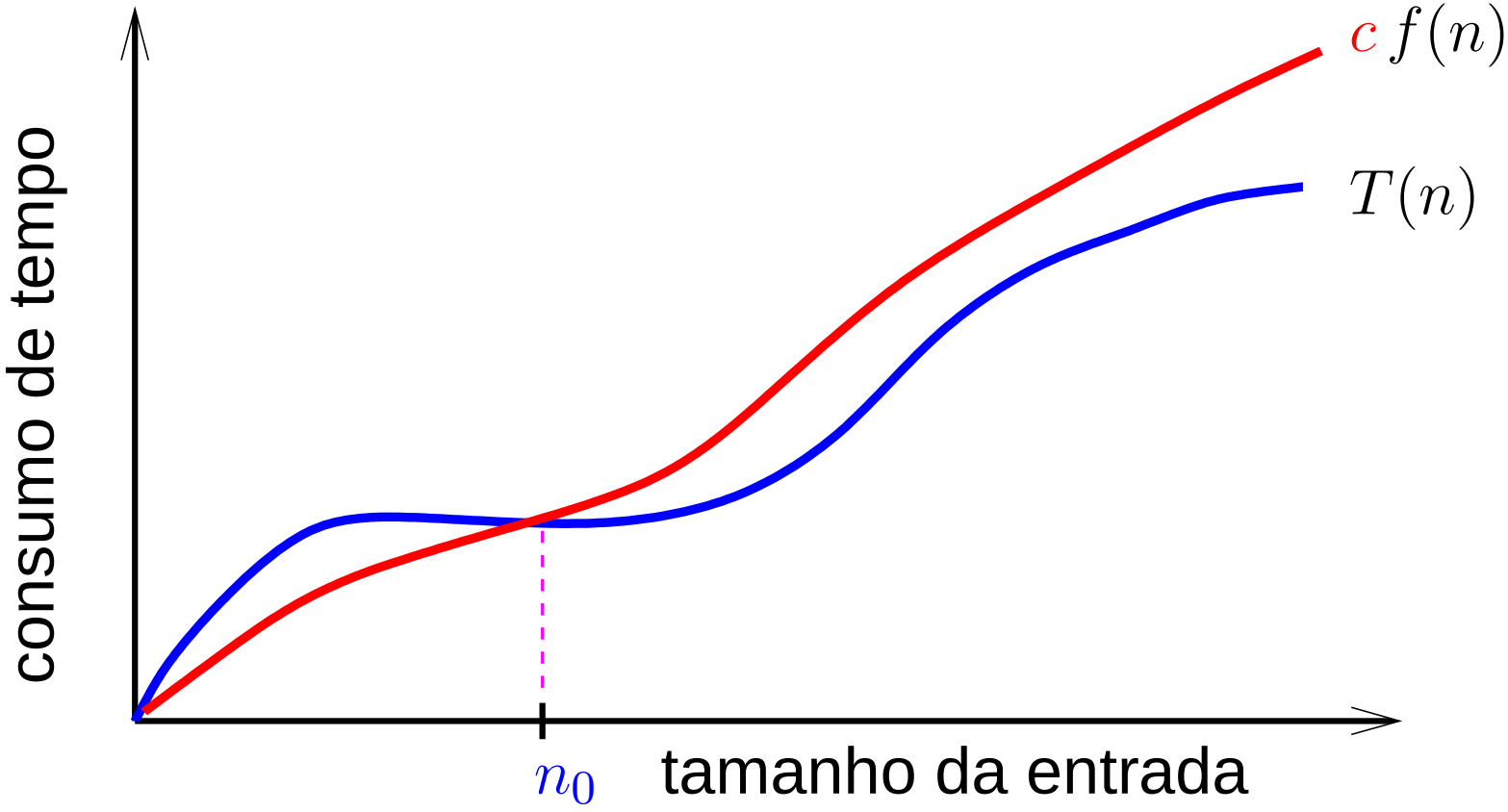


Mais informal

$T(n)$ é $\mathrm{O}(f(n))$ se existe $c > 0$ tal que

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

para todo n suficientemente GRANDE.



Uso da notação O

$\mathrm{O}(f(n)) = \{T(n)$: existem c *e* n_0 tq $0 \leq T(n) \leq cf(n), n \geq n_0\}$

"$T(n)$ 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

$$0 \ \leq \ T(n) \ \leq \ g(n) + c\, f(n)$$

para todo $n \geq n_0$.

Nomes de classes O

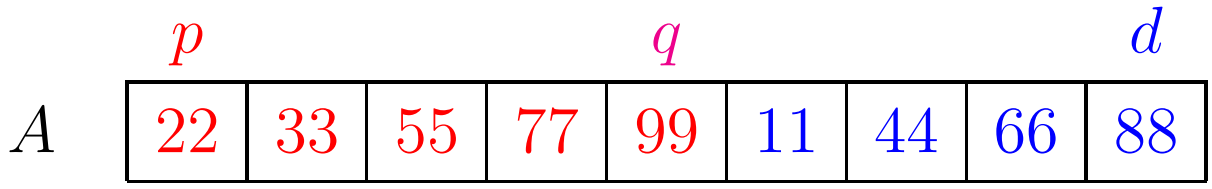
classe	nome
$\mathrm{O}(1)$	constante
$\mathrm{O}(\lg n)$	logarítmica
$\mathrm{O}(n)$	linear
$\mathrm{O}(n \lg n)$	$n \log n$
$\mathrm{O}(n^2)$	quadrática
$\mathrm{O}(n^3)$	cúbica
$\mathrm{O}(n^k)$ com $k \geq 1$	polinomial
$\mathrm{O}(2^n)$	exponencial
$\mathrm{O}(a^n)$ com $a > 1$	exponencial

Análise da intercalação

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

Para que valores de q o problema faz sentido?

Entra:

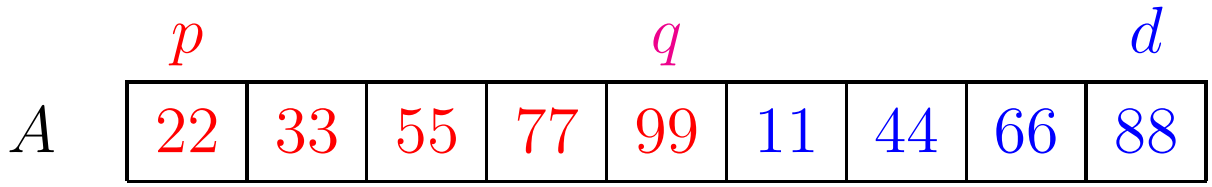


Análise da intercalação

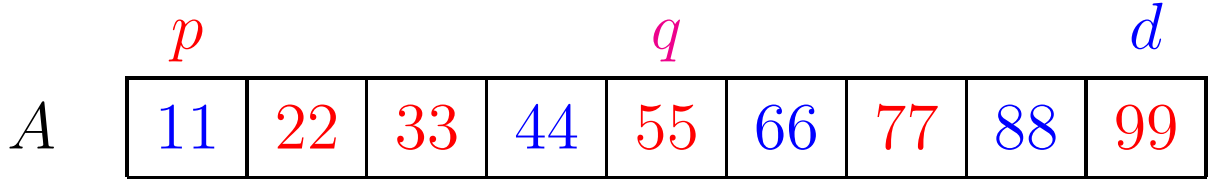
Problema: Dados $A[p\,..\,q]$ e $A[q{+}1\,..\,d]$ crescentes, rearranjar $A[p\,..\,d]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

Entra:



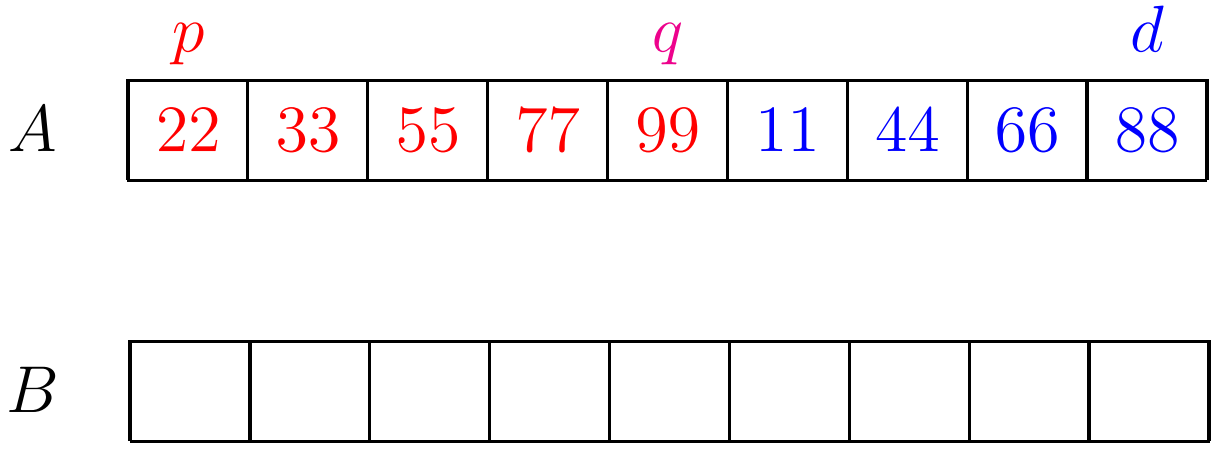
Sai:



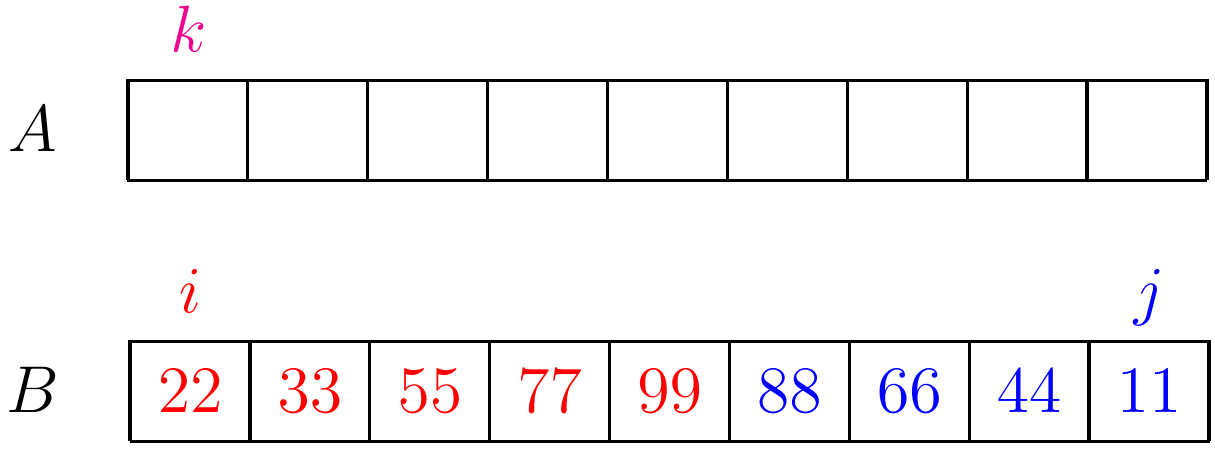
Intercalação

```
INTERCALA (A, p, q, d)
 0   ▷ B[p..d] é um vetor auxiliar
 1   para i ← p até q faça
 2         B[i] ← A[i]
 3   para j ← q + 1 até d faça
 4         B[d + q + 1 − j] ← A[j]
 5   i ← p
 6   j ← d
 7   para k ← p até d 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
```

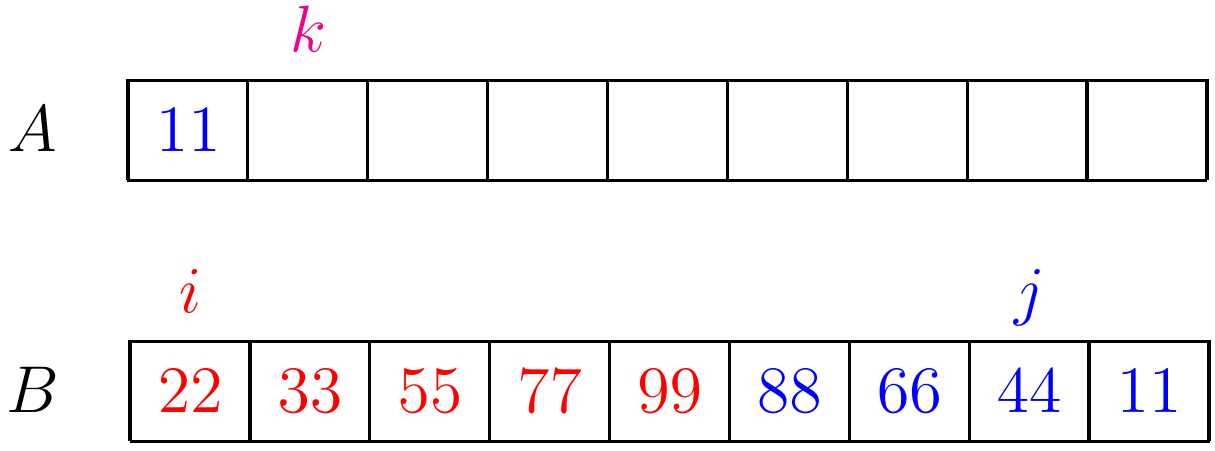
Intercalação



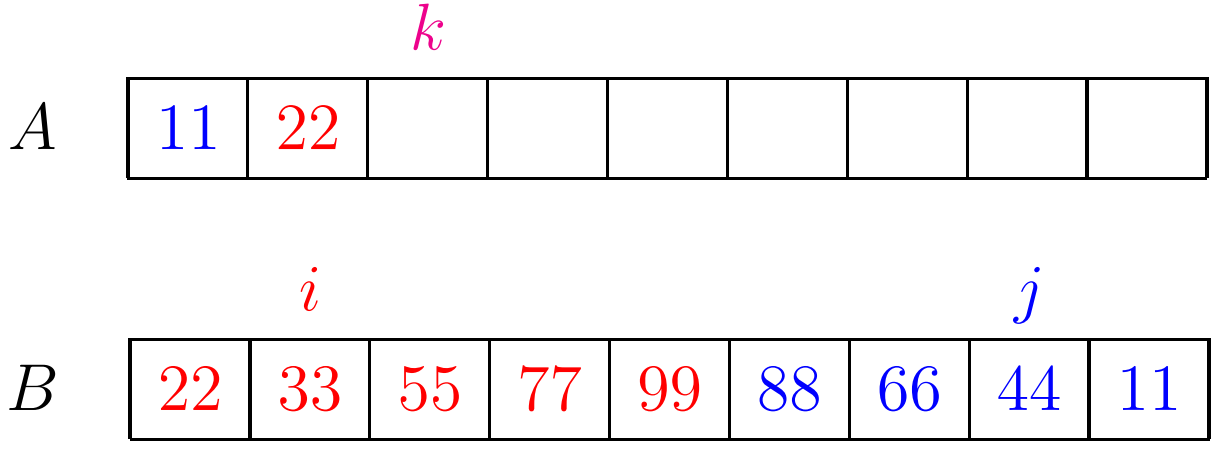
Intercalação



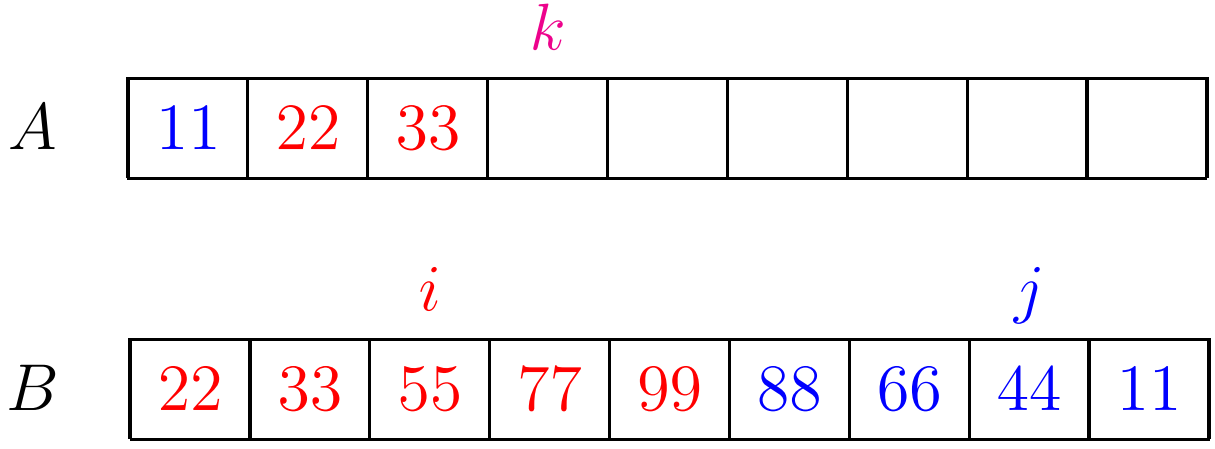
Intercalação



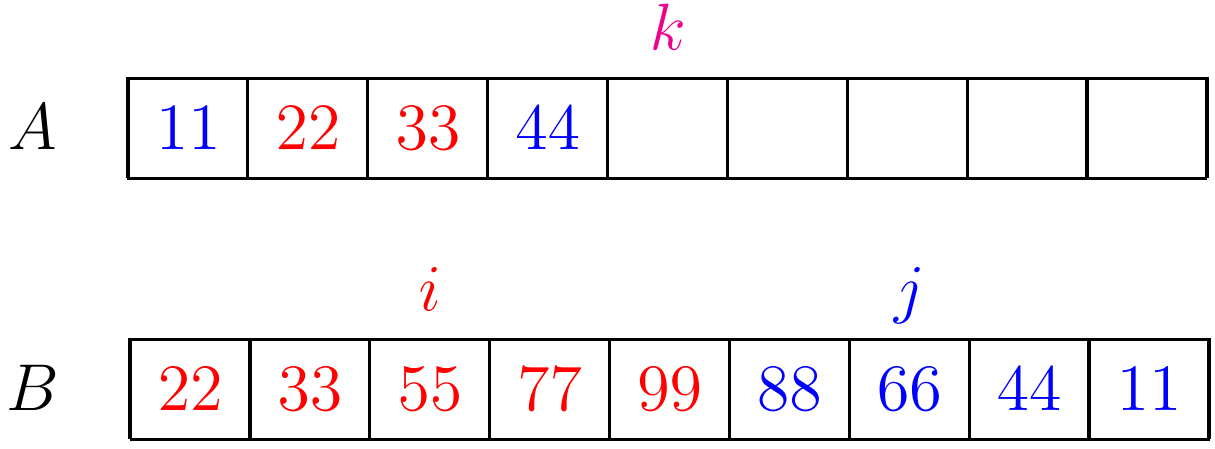
Intercalação



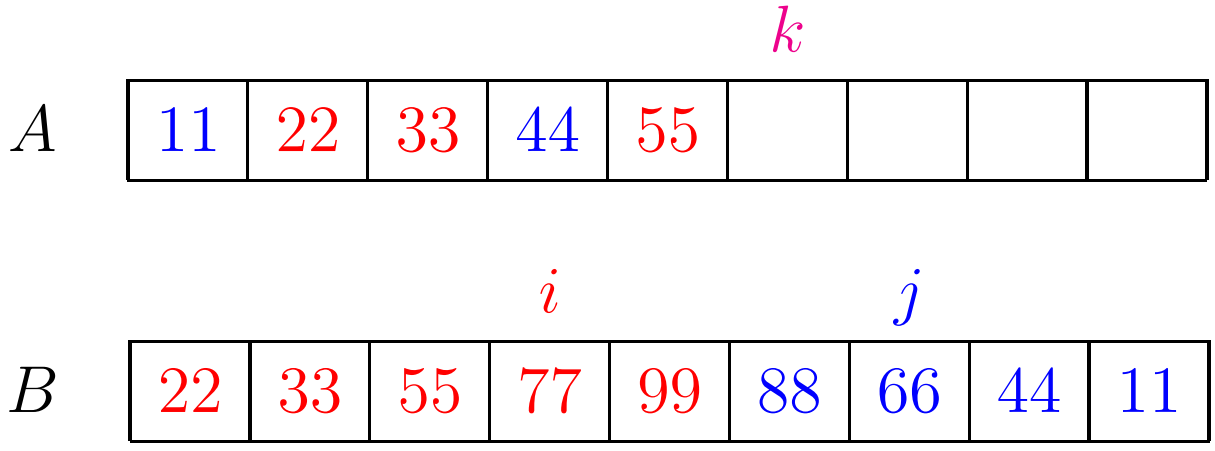
Intercalação



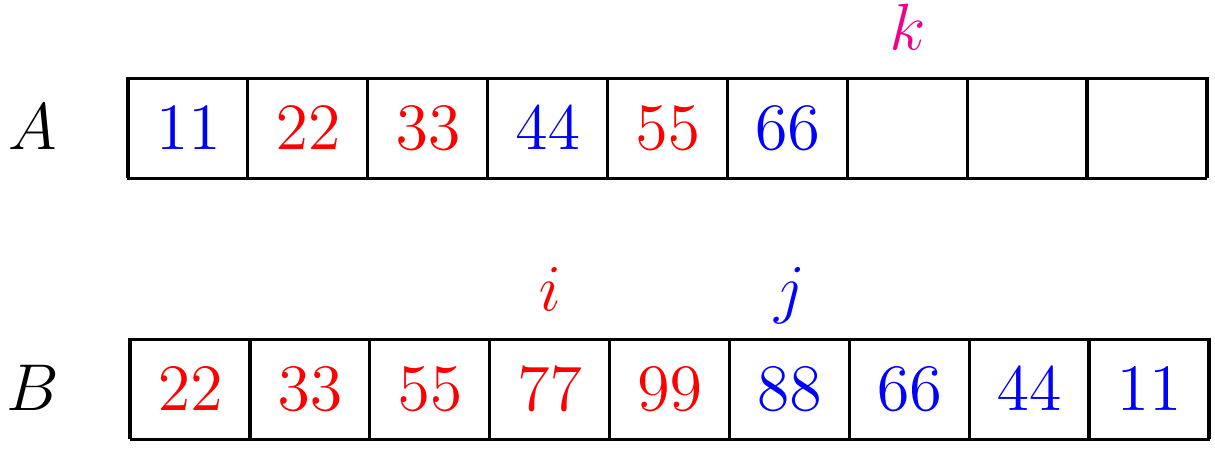
Intercalação



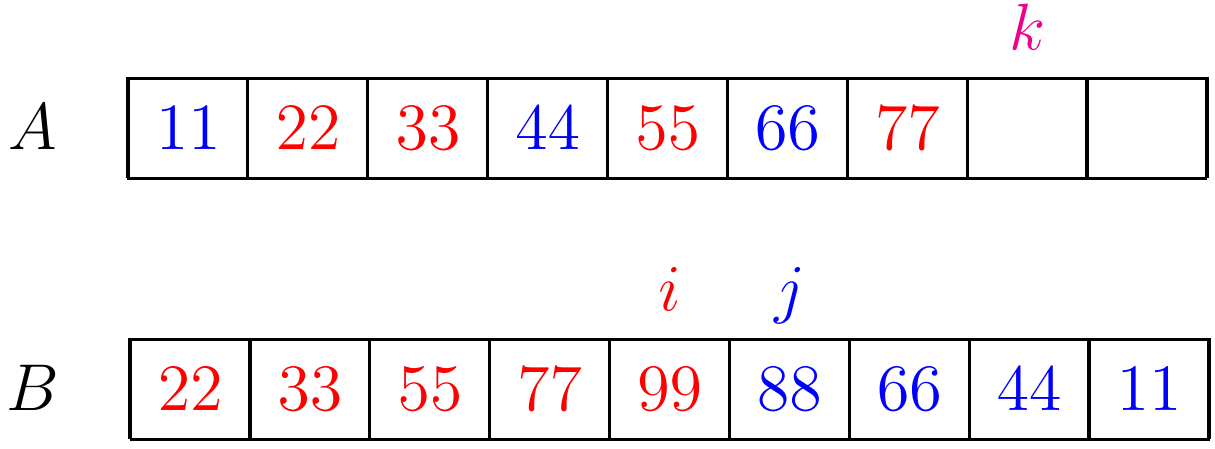
Intercalação



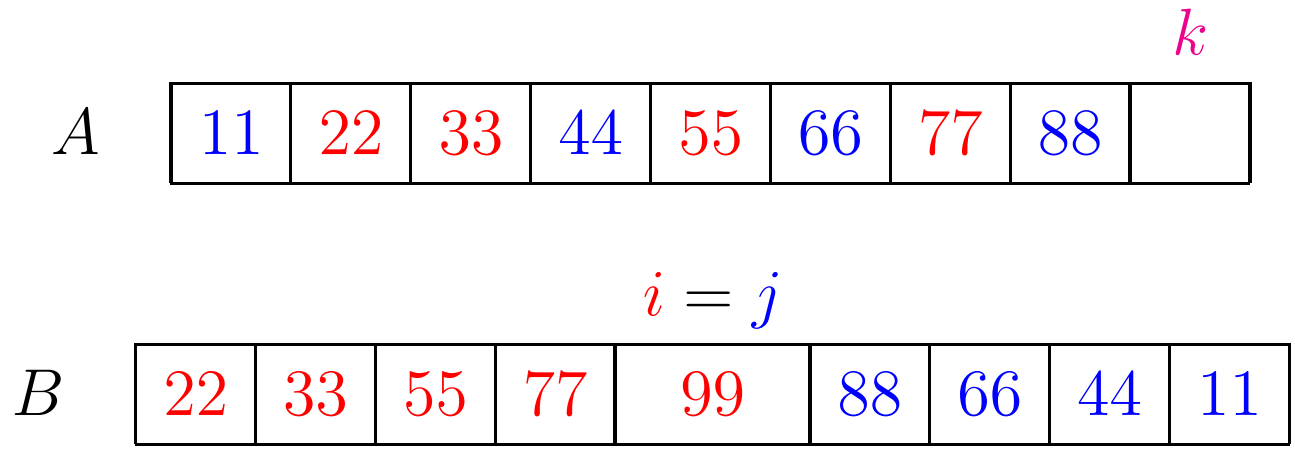
Intercalação



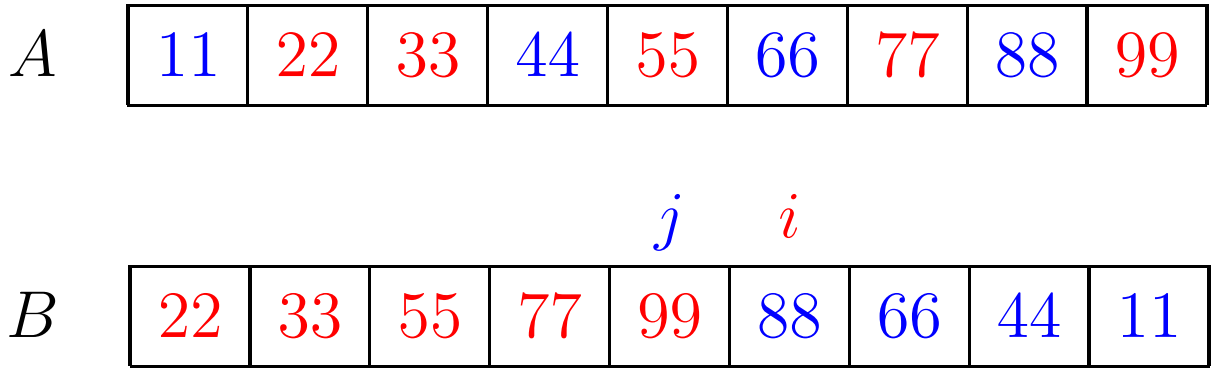
Intercalação



Intercalação



Intercalação



Intercalação

```
INTERCALA (A, p, q, d)
 0   ▷ B[p..d] é um vetor auxiliar
 1   para i ← p até q faça
 2         B[i] ← A[i]
 3   para j ← q + 1 até d faça
 4         B[d + q + 1 − j] ← A[j]
 5   i ← p
 6   j ← d
 7   para k ← p até d 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	$= \quad q - p + 2 \;=\; n - d + q + 1$
2	$= \quad q - p + 1 \;=\; n - d + q$
3	$= \quad d - (q + 1) + 2 \;=\; n - q + p$
4	$= \quad d - (q + 1) + 1 \;=\; n - q + p - 1$
5	$= \quad 1$
6	$= \quad 1$
7	$= \quad d - p + 2 \;=\; n + 1$
8	$= \quad d - p + 1 \;=\; n$
9–12	$= \quad 2\,(d - p + 1) \;=\; 2n$
total	$= \quad 8n - 2(d - p + 1) + 5 = \; 6n + 5$

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:

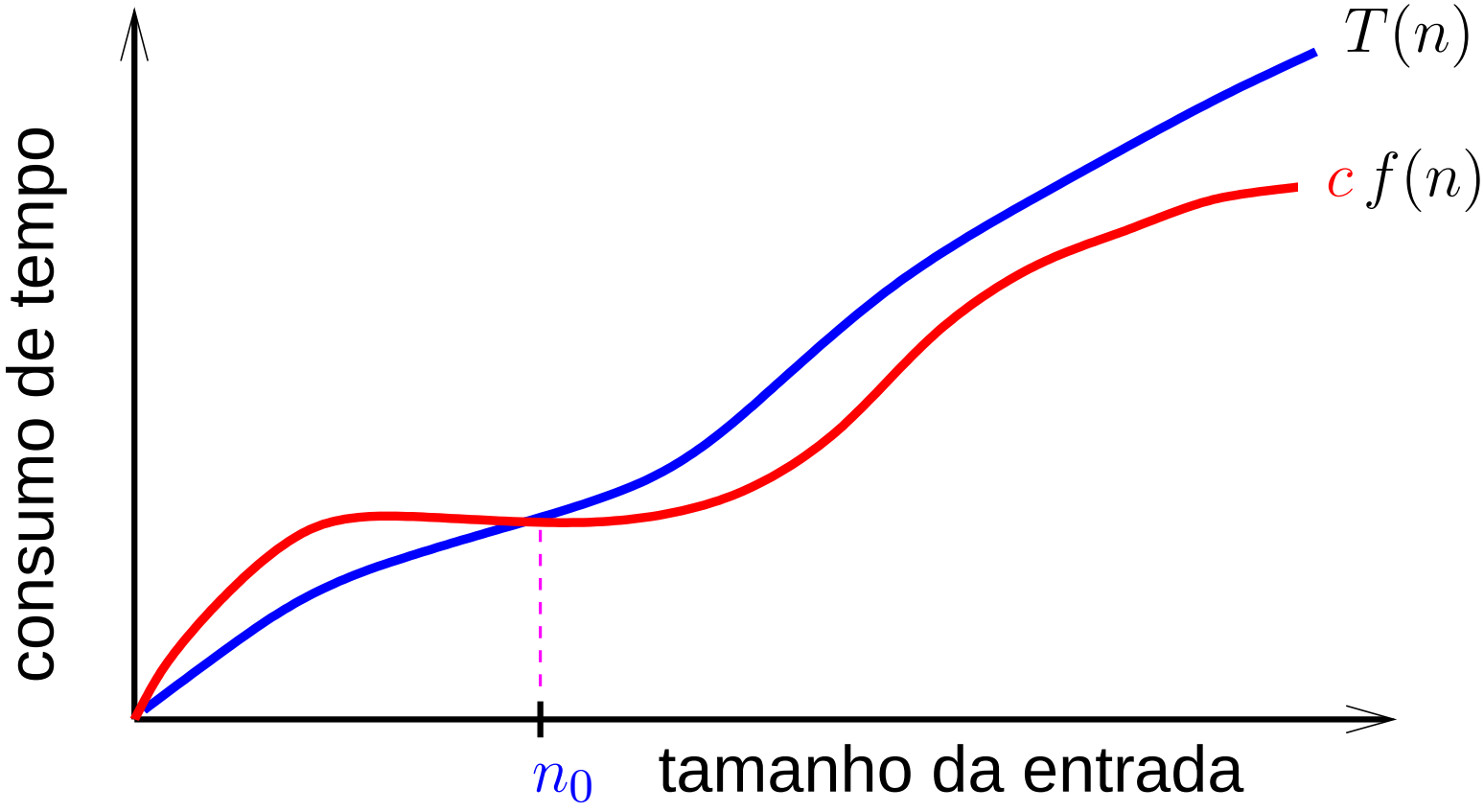
O algoritmo consome $\mathrm{O}(n)$ unidades de tempo.

Definição

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

$$0 \leq c\,f(n) \leq T(n)$$

para todo $n \geq n_0$.

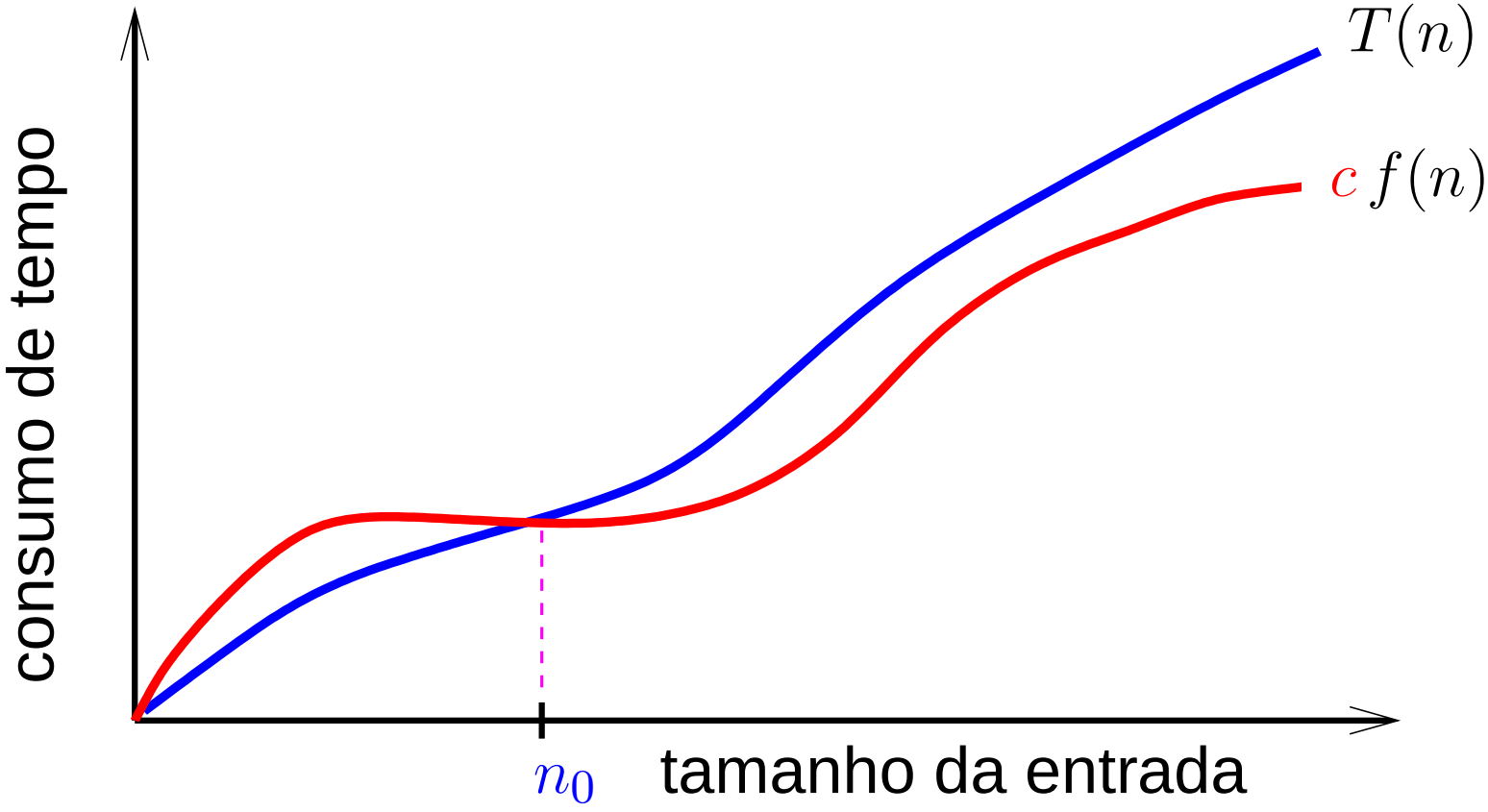


Mais informal

$T(n) = \Omega(f(n))$ 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	$= \quad q - p + 2 \;=\; n - d + q + 1$
2	$= \quad q - p + 1 \;=\; n - d + q$
3	$= \quad d - (q + 1) + 2 \;=\; n - q + p$
4	$= \quad d - (q + 1) + 1 \;=\; n - q + p - 1$
5	$= \quad 1$
6	$= \quad 1$
7	$= \quad d - p + 2 \;=\; n + 1$
8	$= \quad d - p + 1 \;=\; n$
9–12	$= \quad 2\,(d - p + 1) \;=\; 2n$
total	$= \quad 8n - 2(d - p + 1) + 5 = \; 6n + 5$

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

$$0 \leq c\,f(n) \leq T(n)$$

para todo $n \geq n_0$. Logo,

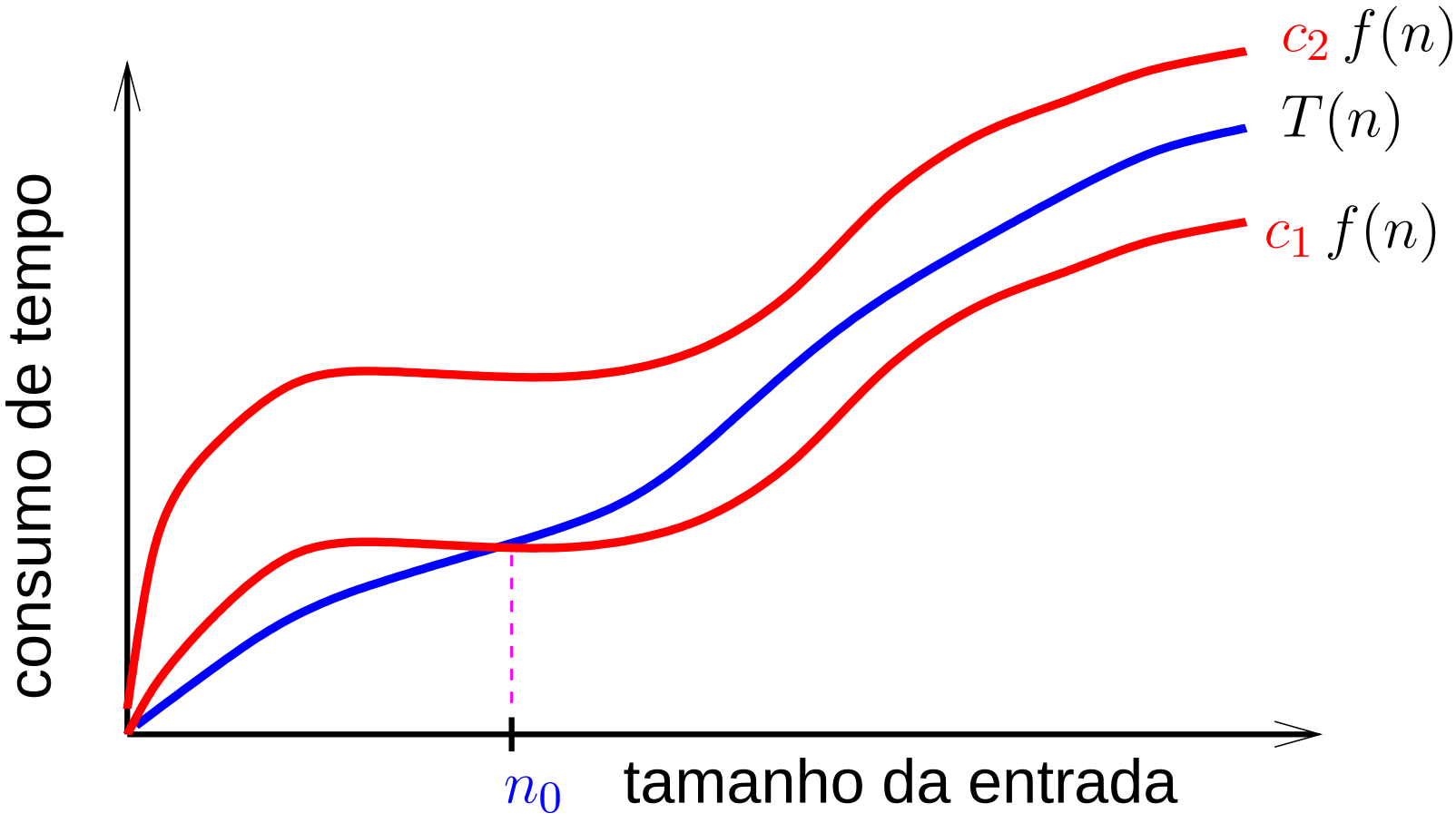
$$f(n) \leq 1/c\,T(n)$$

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

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros nos reais.
Dizemos que $T(n)$ é $\Theta(f(n))$ se

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

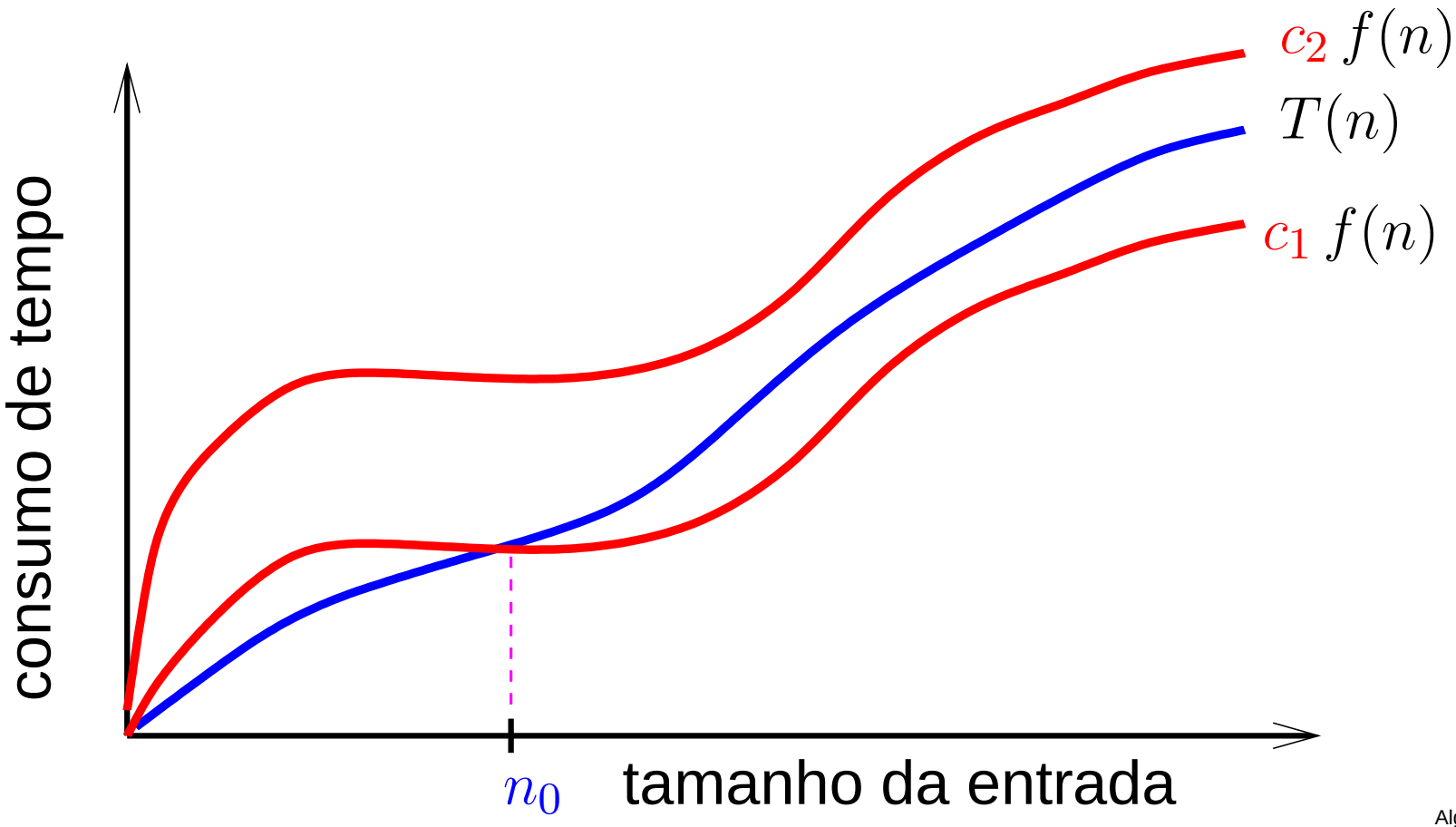


Definição

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 $n \geq n_0$.



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.

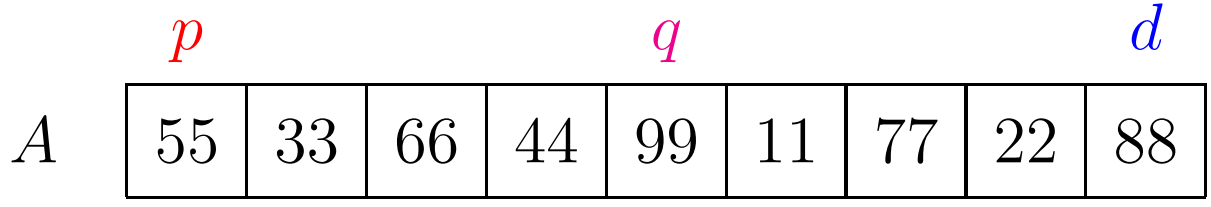
Um algoritmo que consome tempo $\Theta(2^n)$ é dificilmente aceitável.

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

Mergesort

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

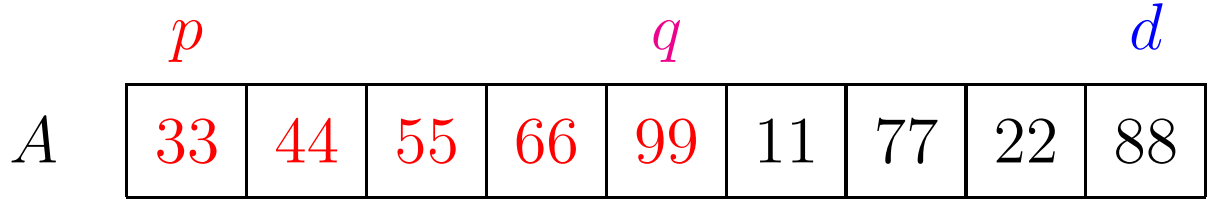
```
MERGE-SORT (A, p, d)
1    se p < d
2         então q ← ⌊(p + d)/2⌋
3               MERGE-SORT (A, p, q)
4               MERGE-SORT (A, q + 1, d)
5               INTERCALA (A, p, q, d)
```



Merge-Sort

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

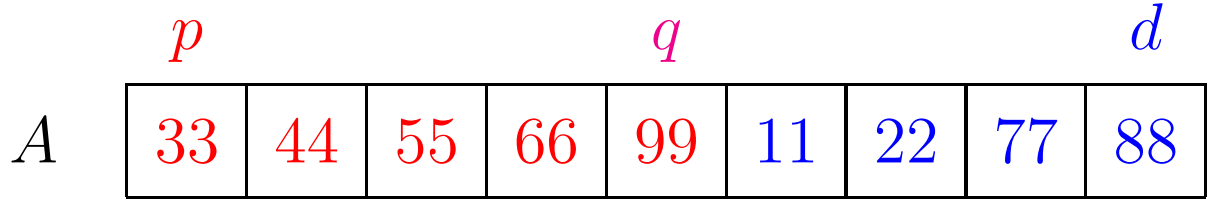
```
MERGE-SORT (A, p, d)
1   se p < d
2       então q ← ⌊(p + d)/2⌋
3             MERGE-SORT (A, p, q)
4             MERGE-SORT (A, q + 1, d)
5             INTERCALA (A, p, q, d)
```



Mergesort

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

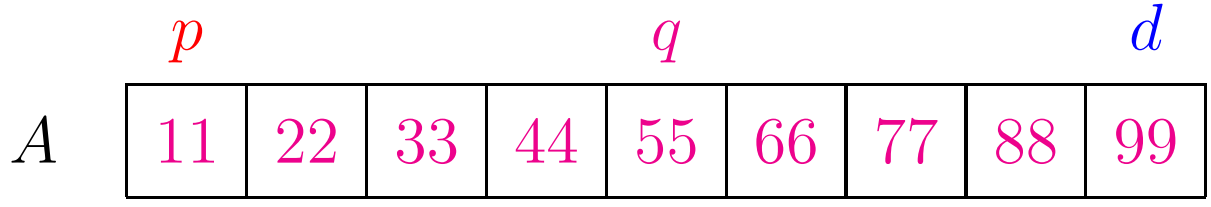
```
MERGE-SORT (A, p, d)
1    se p < d
2         então q ← ⌊(p + d)/2⌋
3               MERGE-SORT (A, p, q)
4               MERGE-SORT (A, q + 1, d)
5               INTERCALA (A, p, q, d)
```



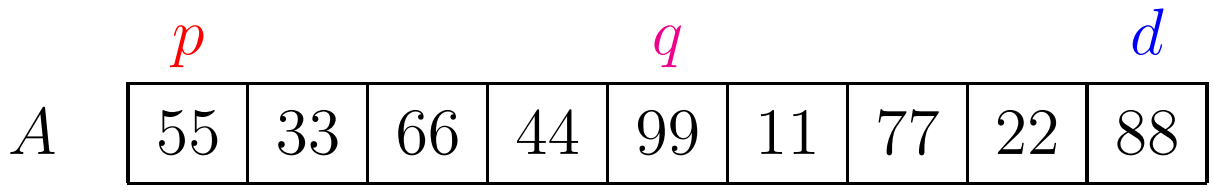
Mergesort

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

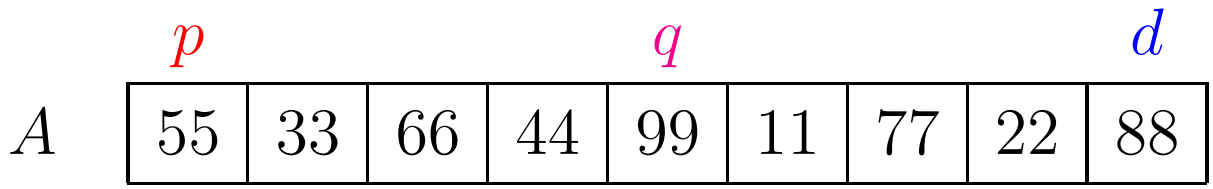
```
MERGE-SORT (A, p, d)
1    se p < d
2         então q ← ⌊(p + d)/2⌋
3               MERGE-SORT (A, p, q)
4               MERGE-SORT (A, q + 1, d)
5               INTERCALA (A, p, q, d)
```

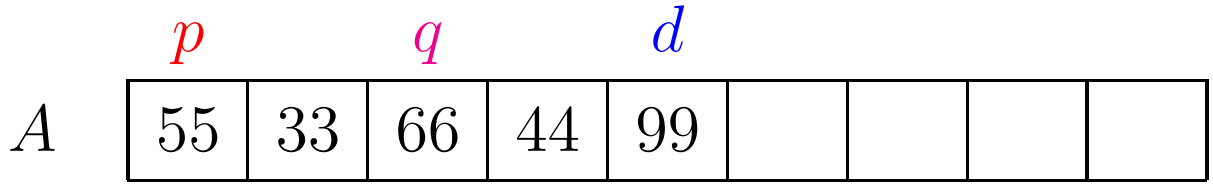


Mergesort

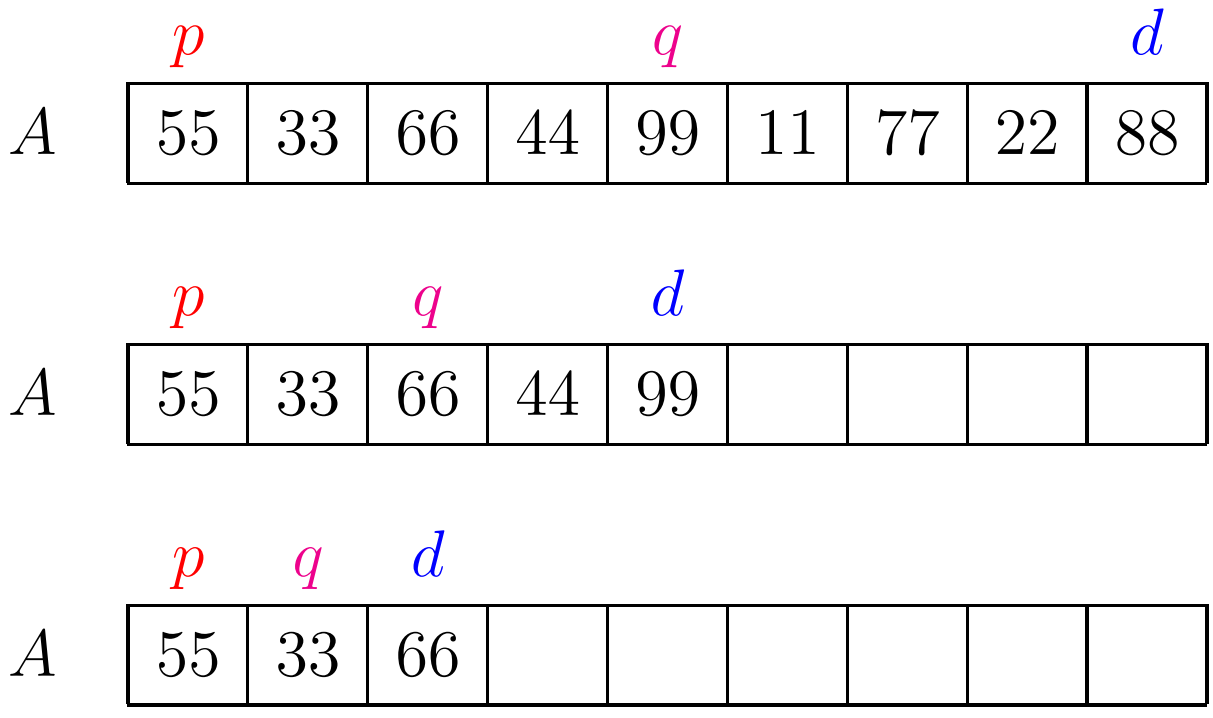


Mergesort

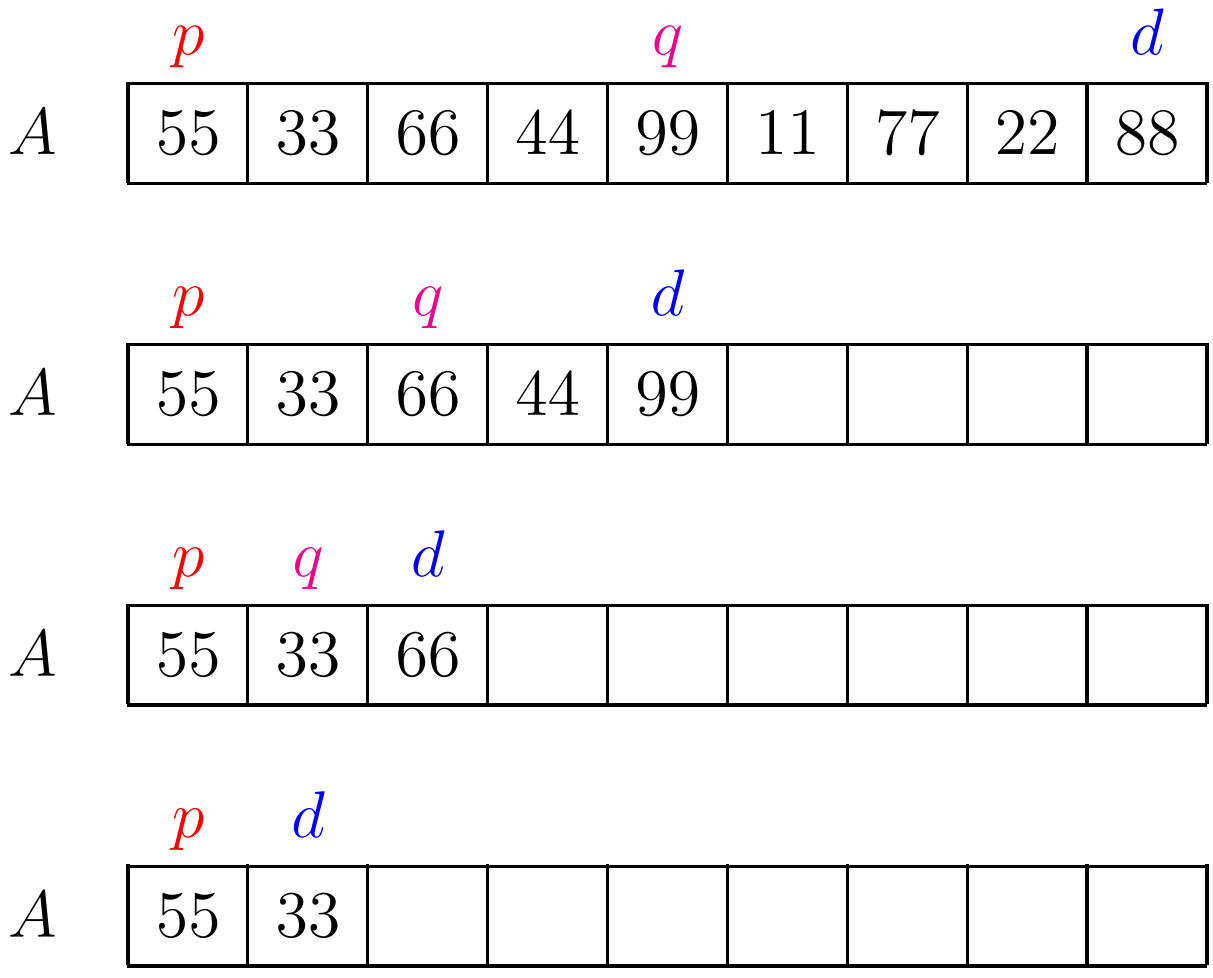




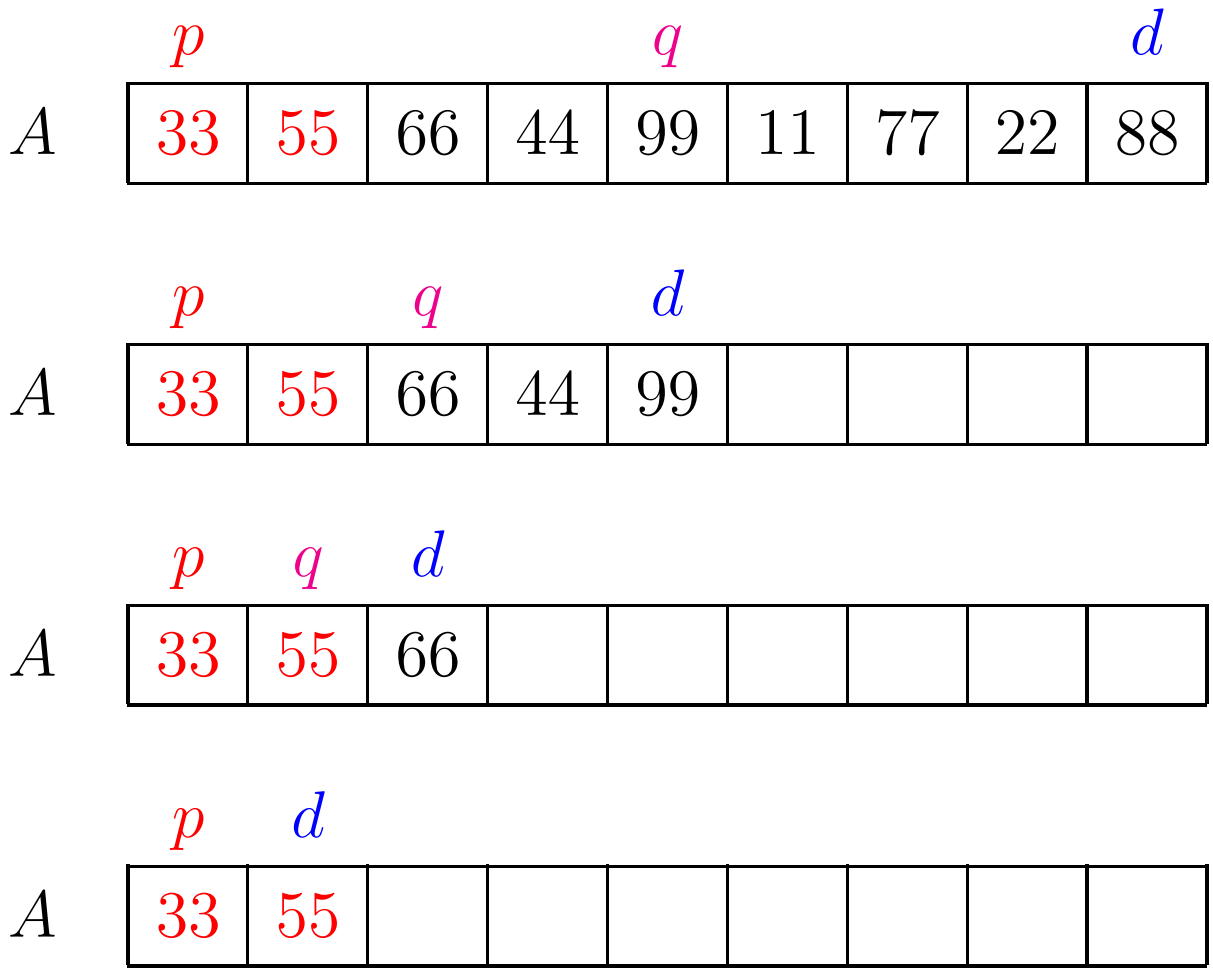
Mergesort



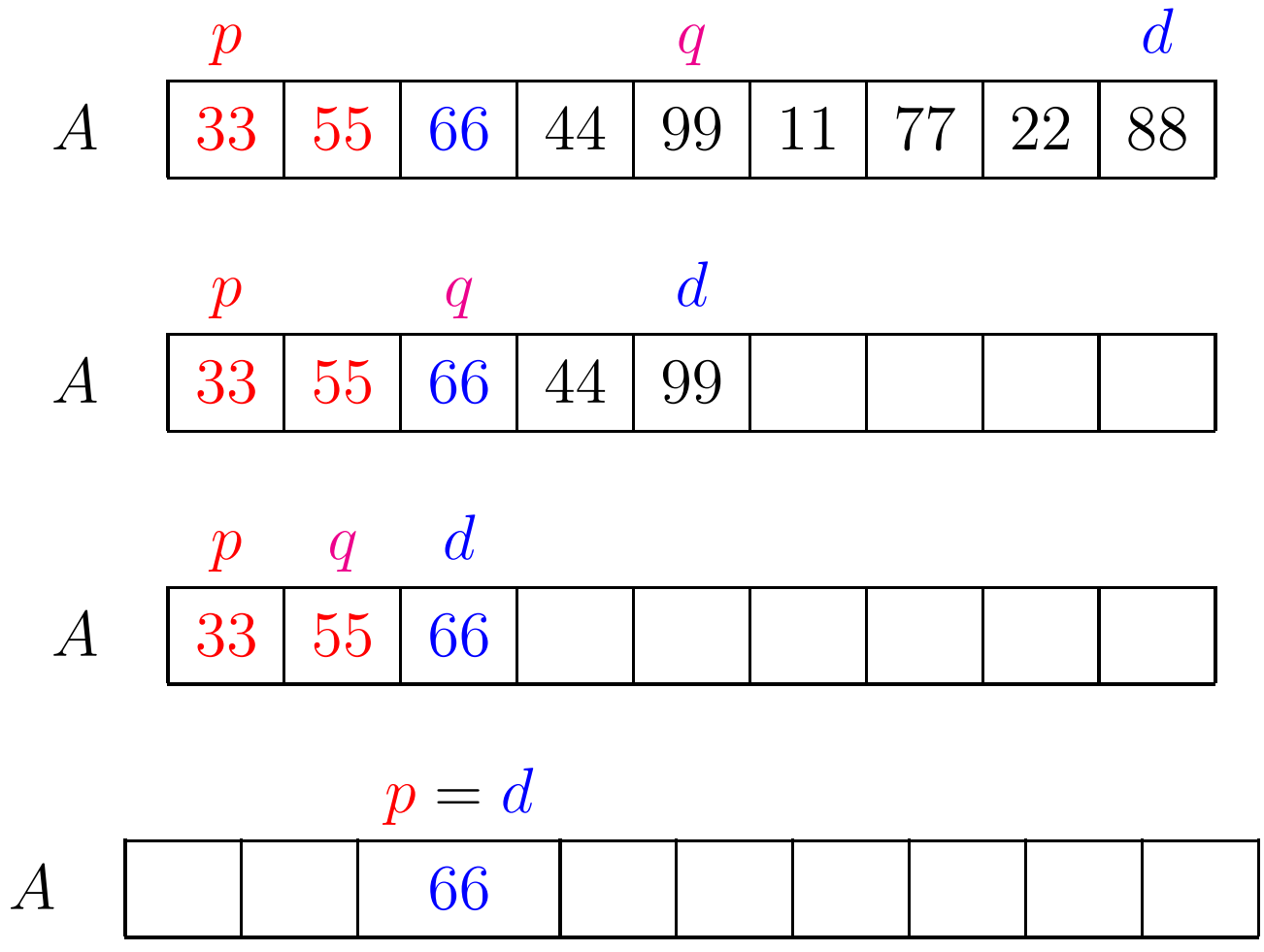
Mergesort


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

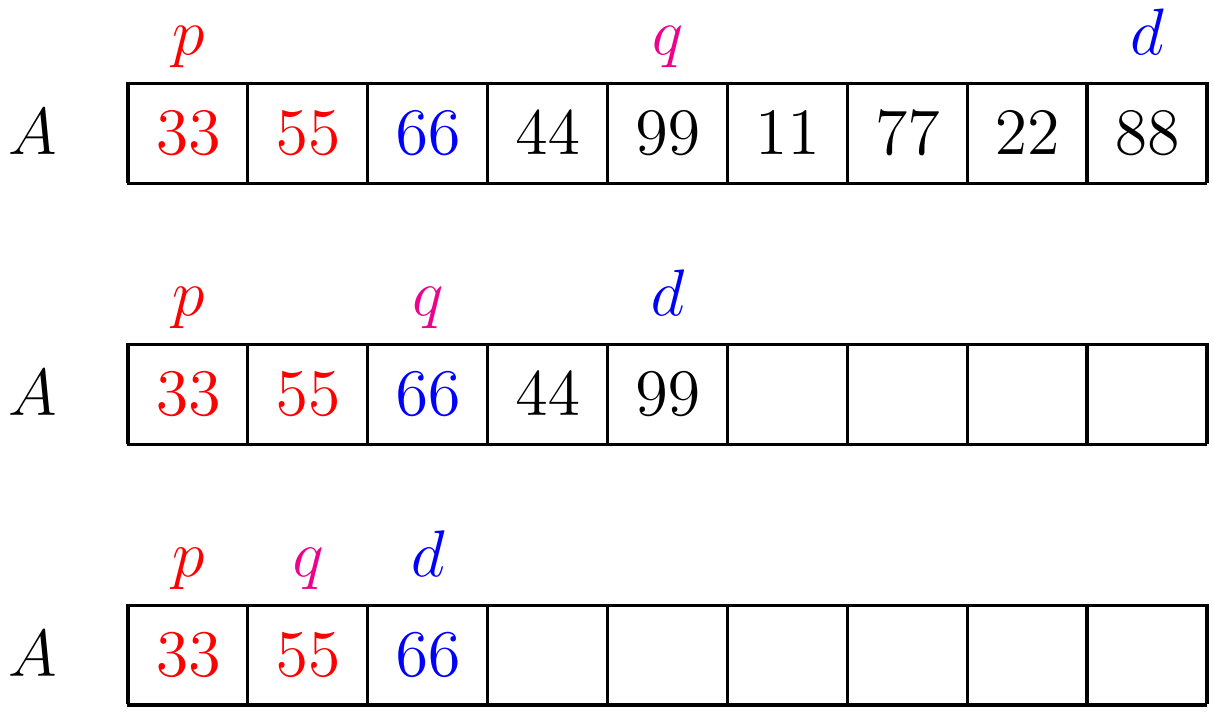
Mergesort


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

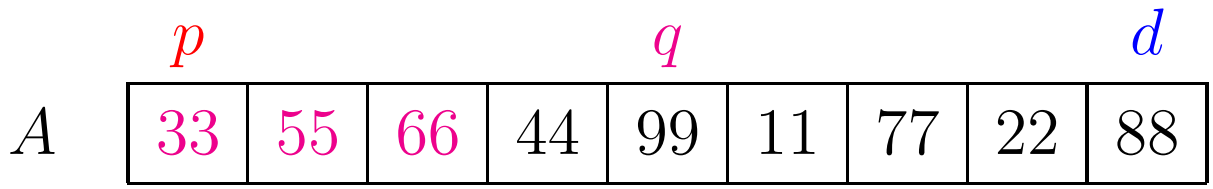
Mergesort

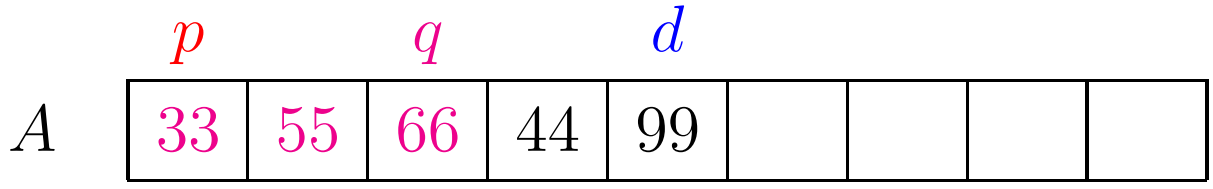

p
q
d
A
33 55 66 44 99 11 77 22 88
p
q
d
A
33 55 66 44 99
p
q
d
A
33 55 66
p = d
A
66

Mergesort

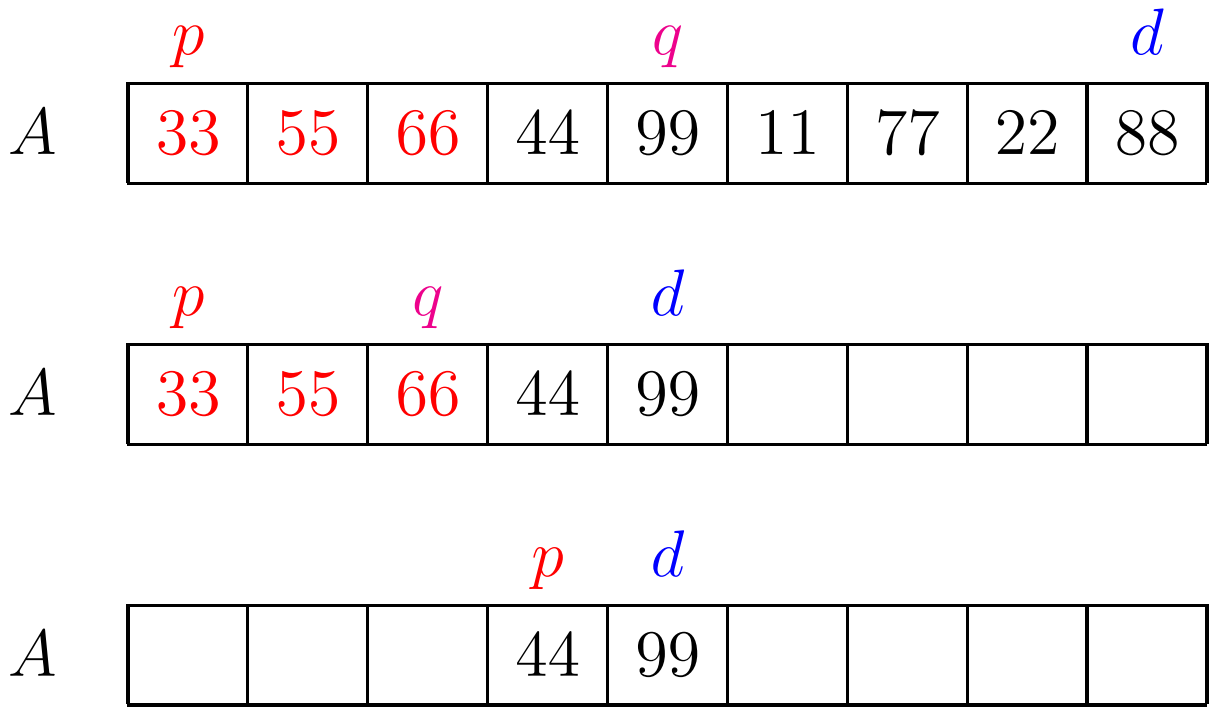


Mergesort

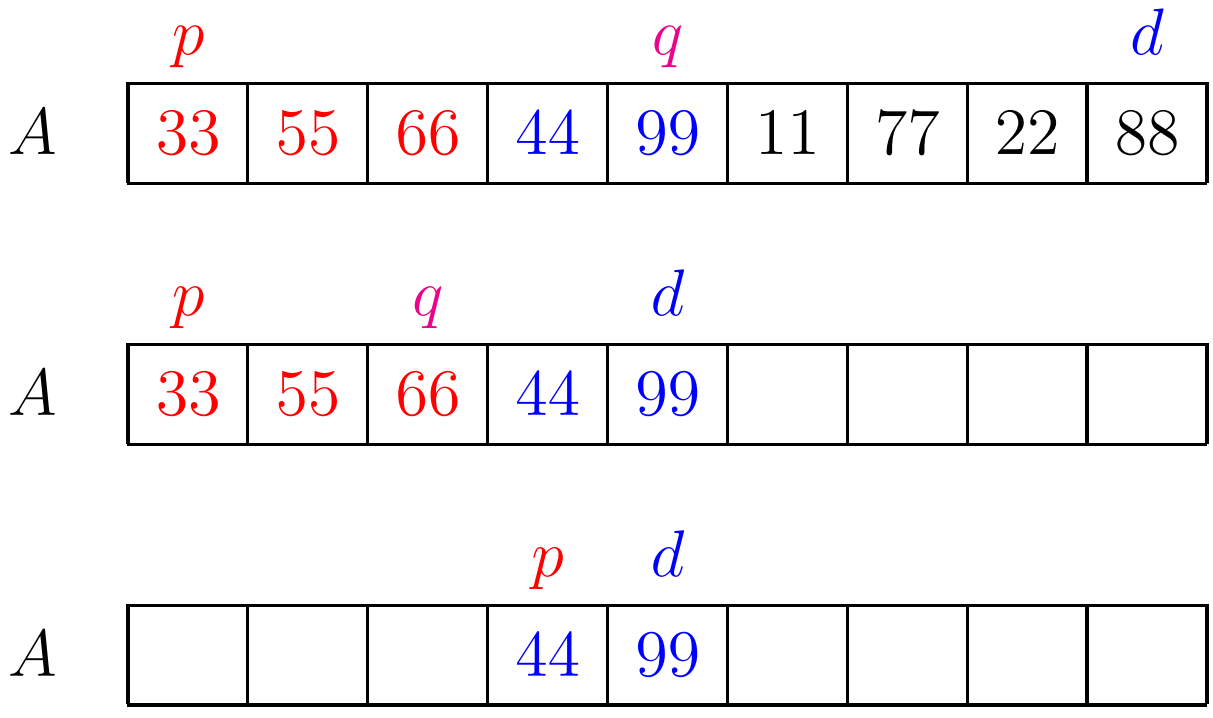




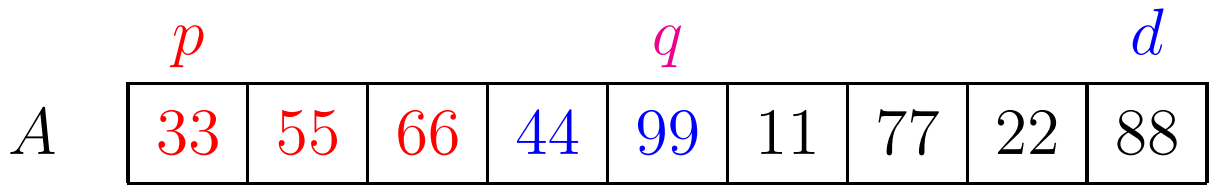
Mergesort

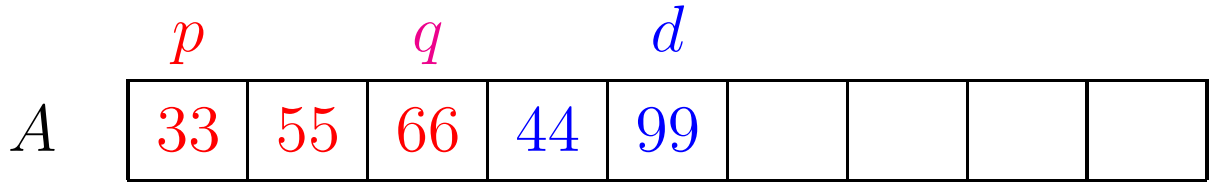


Mergesort

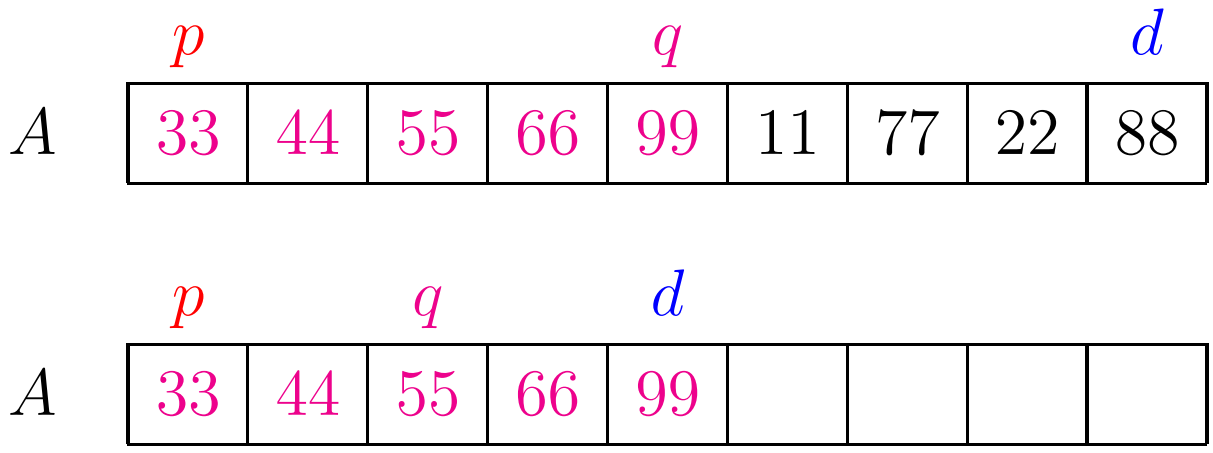


Mergesort

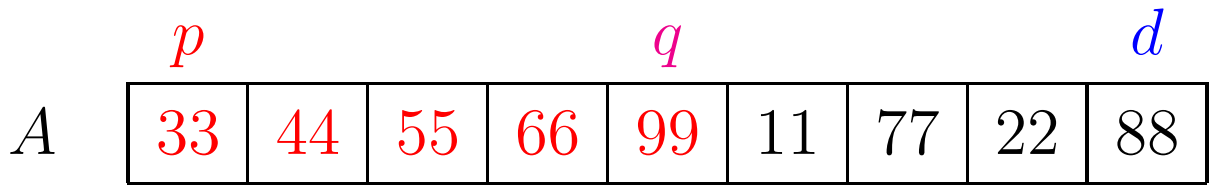




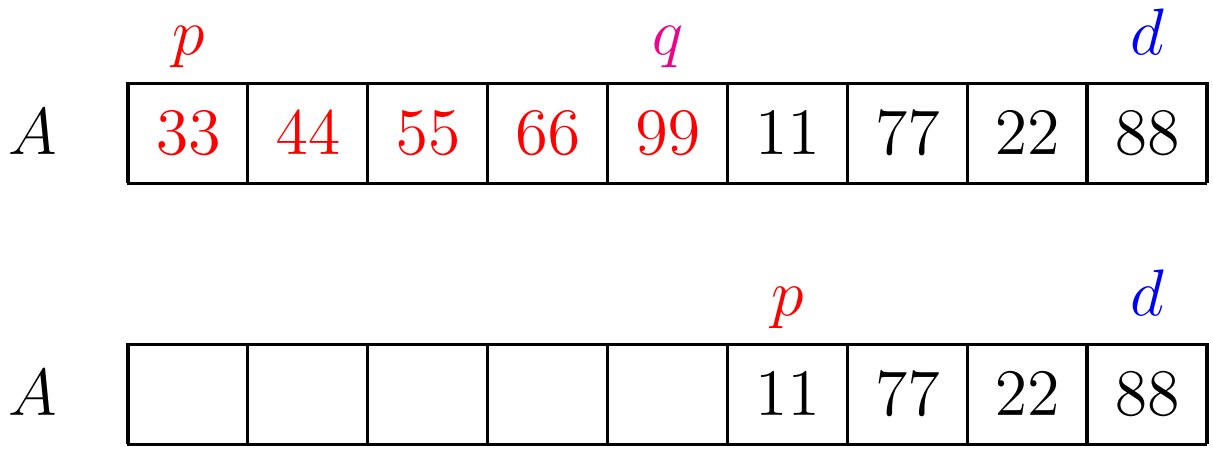
Mergesort



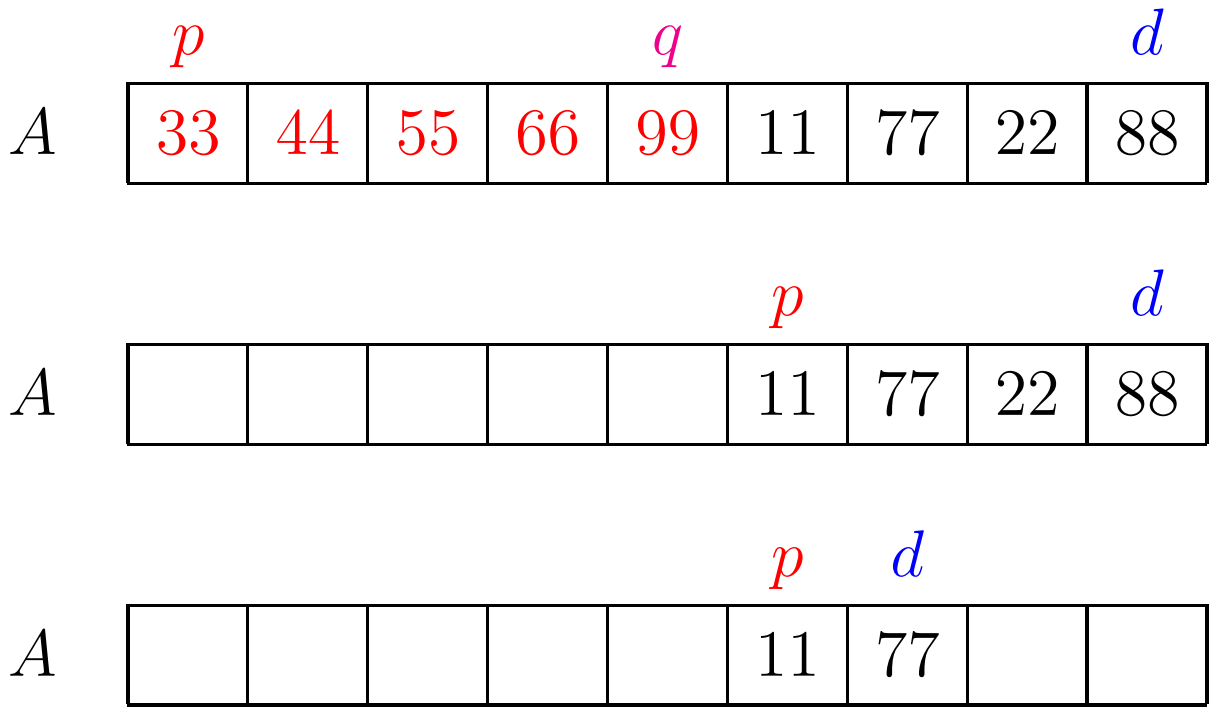
Mergesort



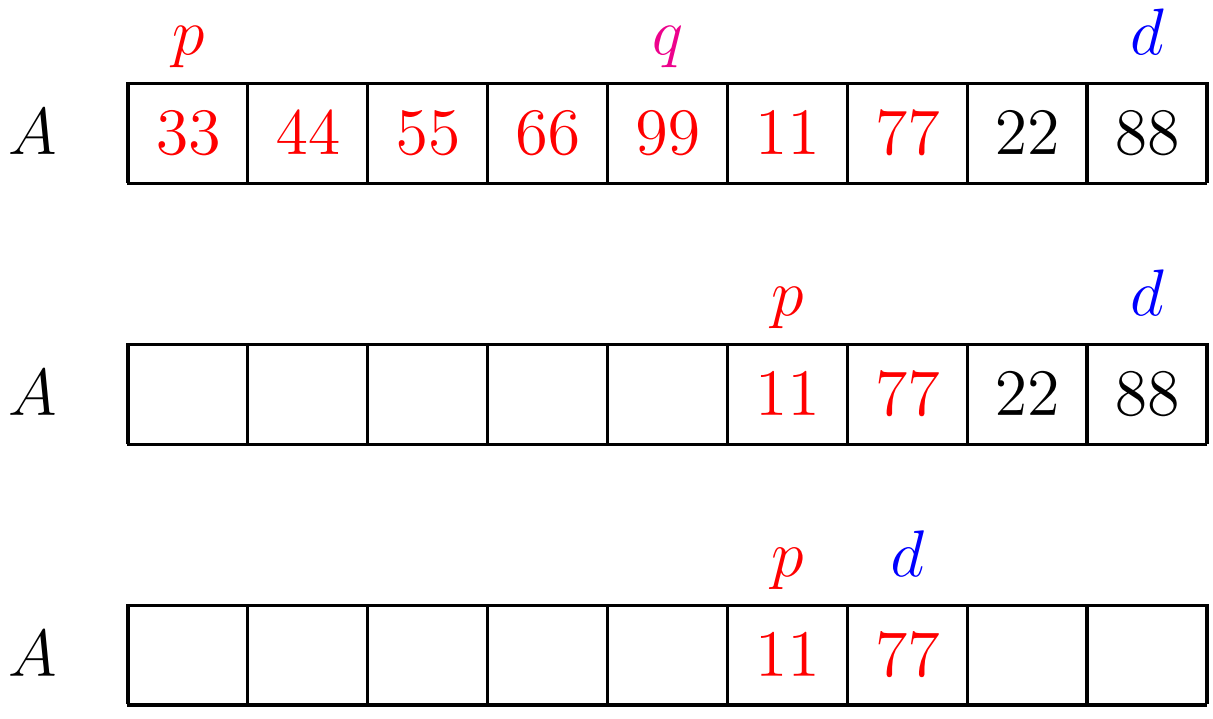
Mergesort



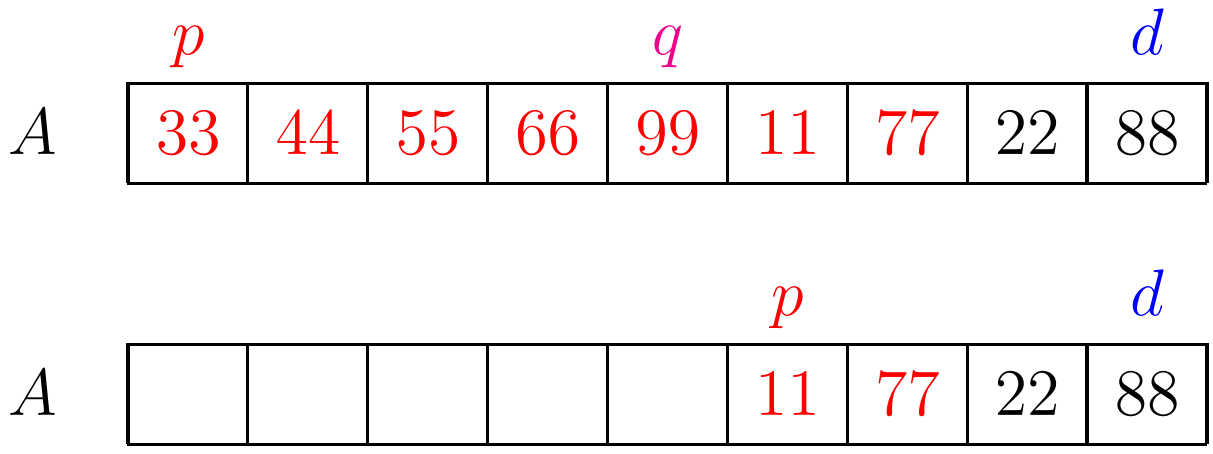
Mergesort



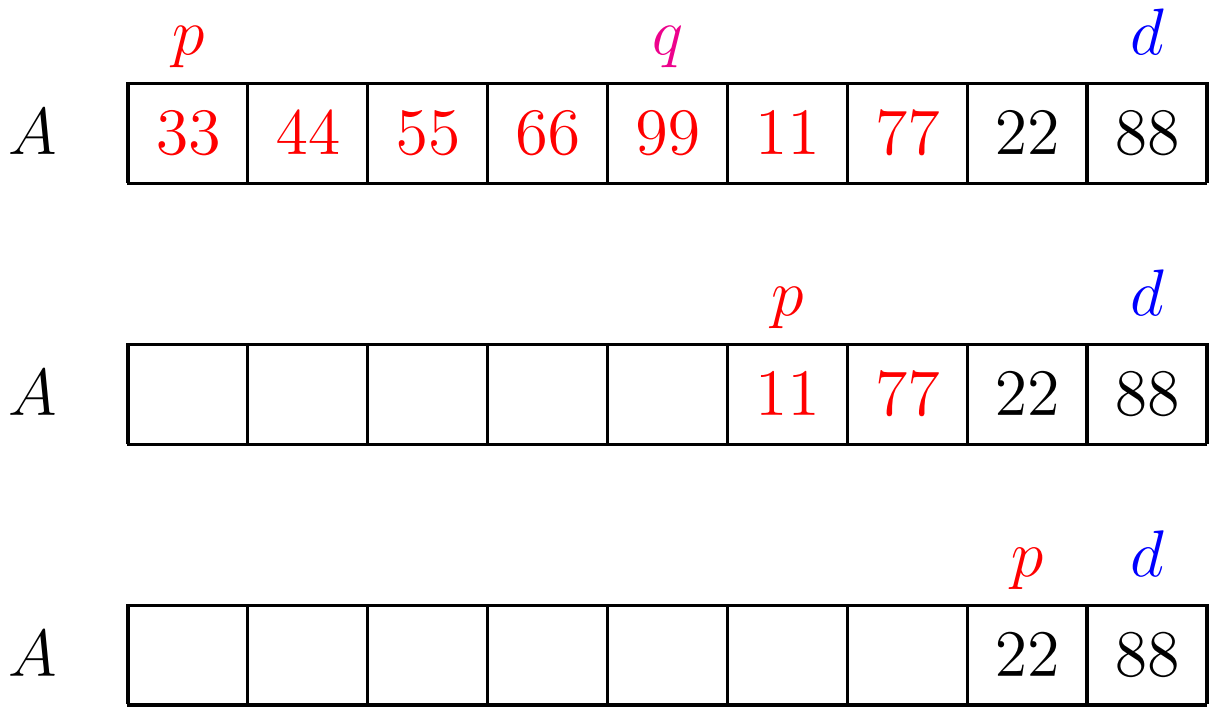
Mergesort



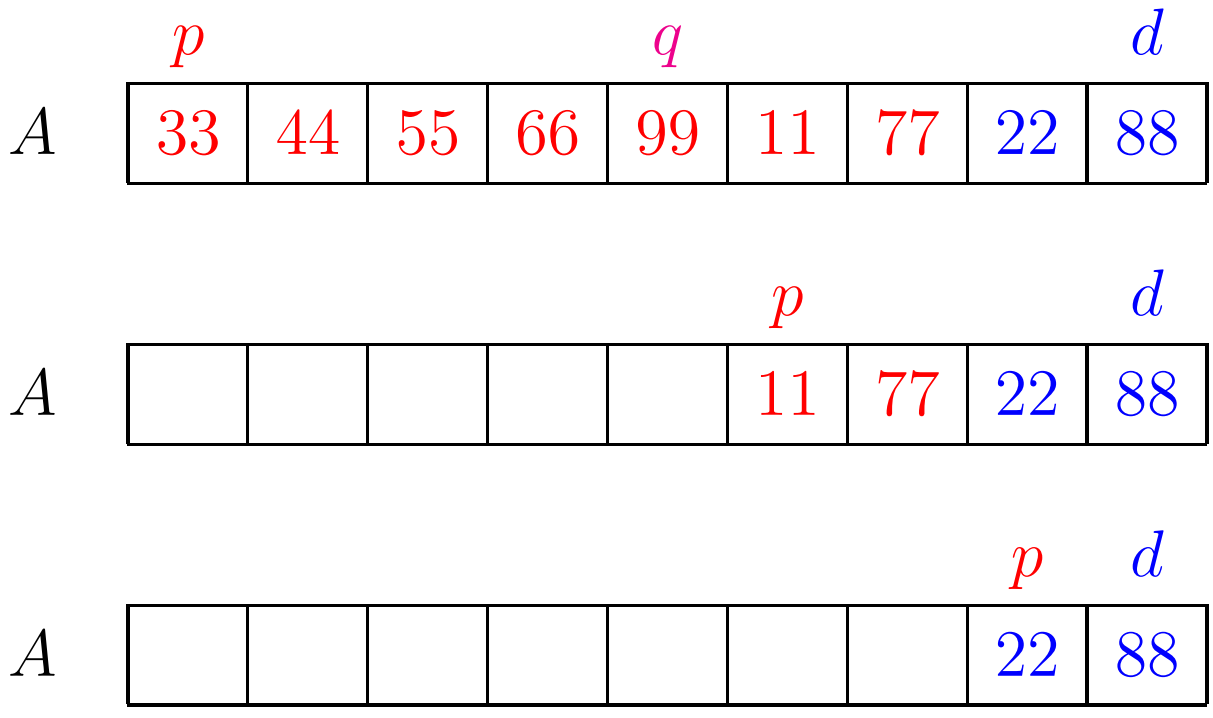
Mergesort



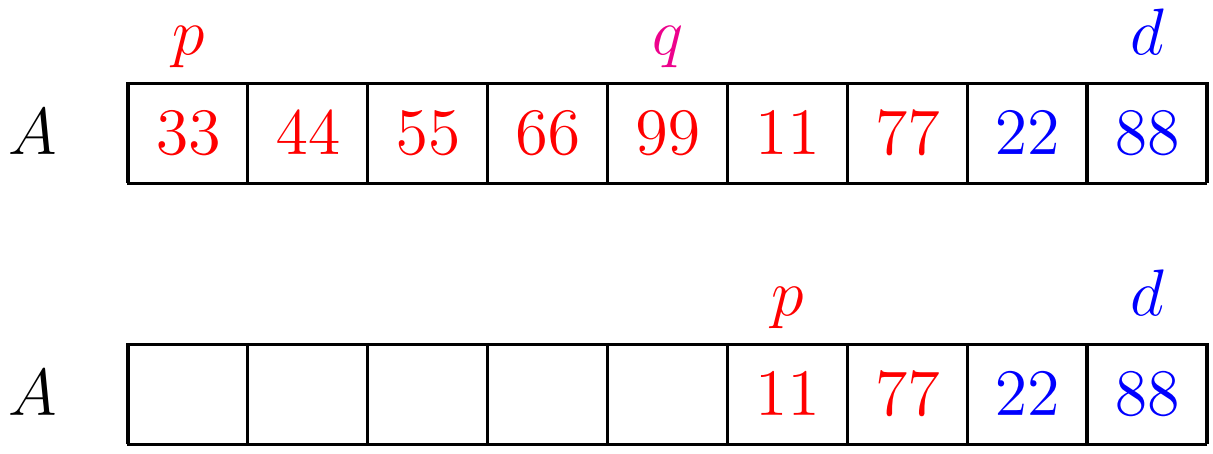
Mergesort



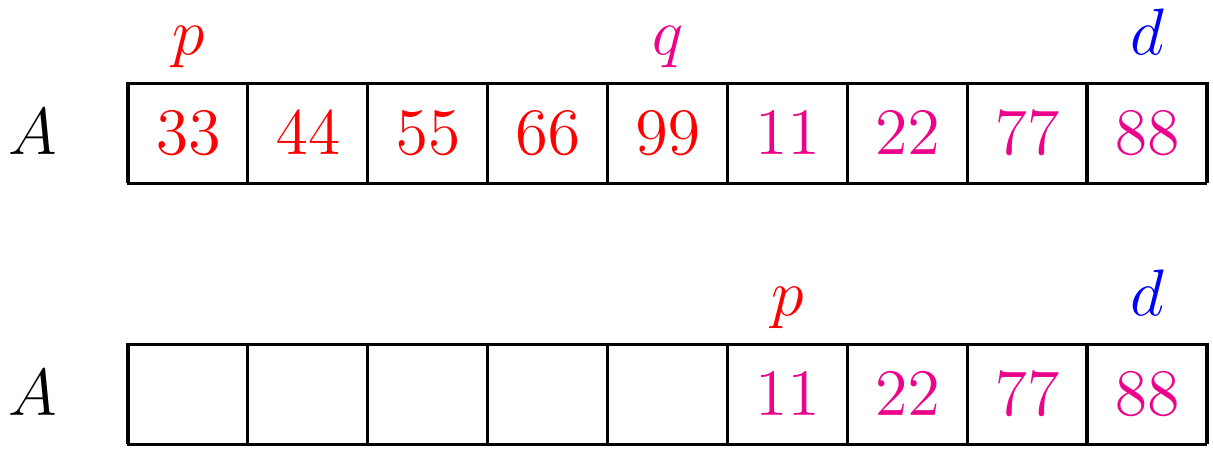
Mergesort



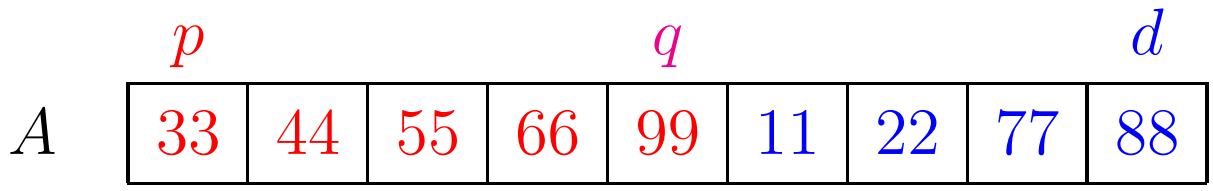
Mergesort



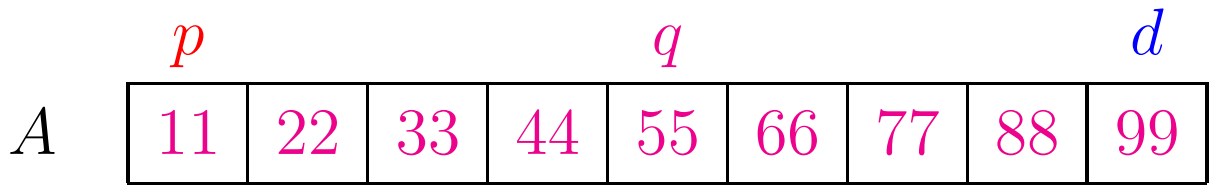
Mergesort



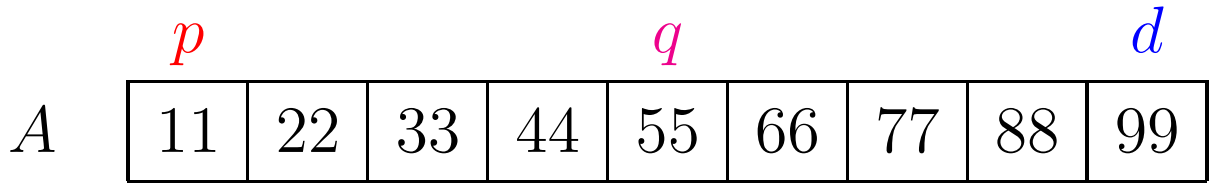
Mergesort



Mergesort



Mergesort



Mergesort

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

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

O algoritmo está correto?

Mergesort

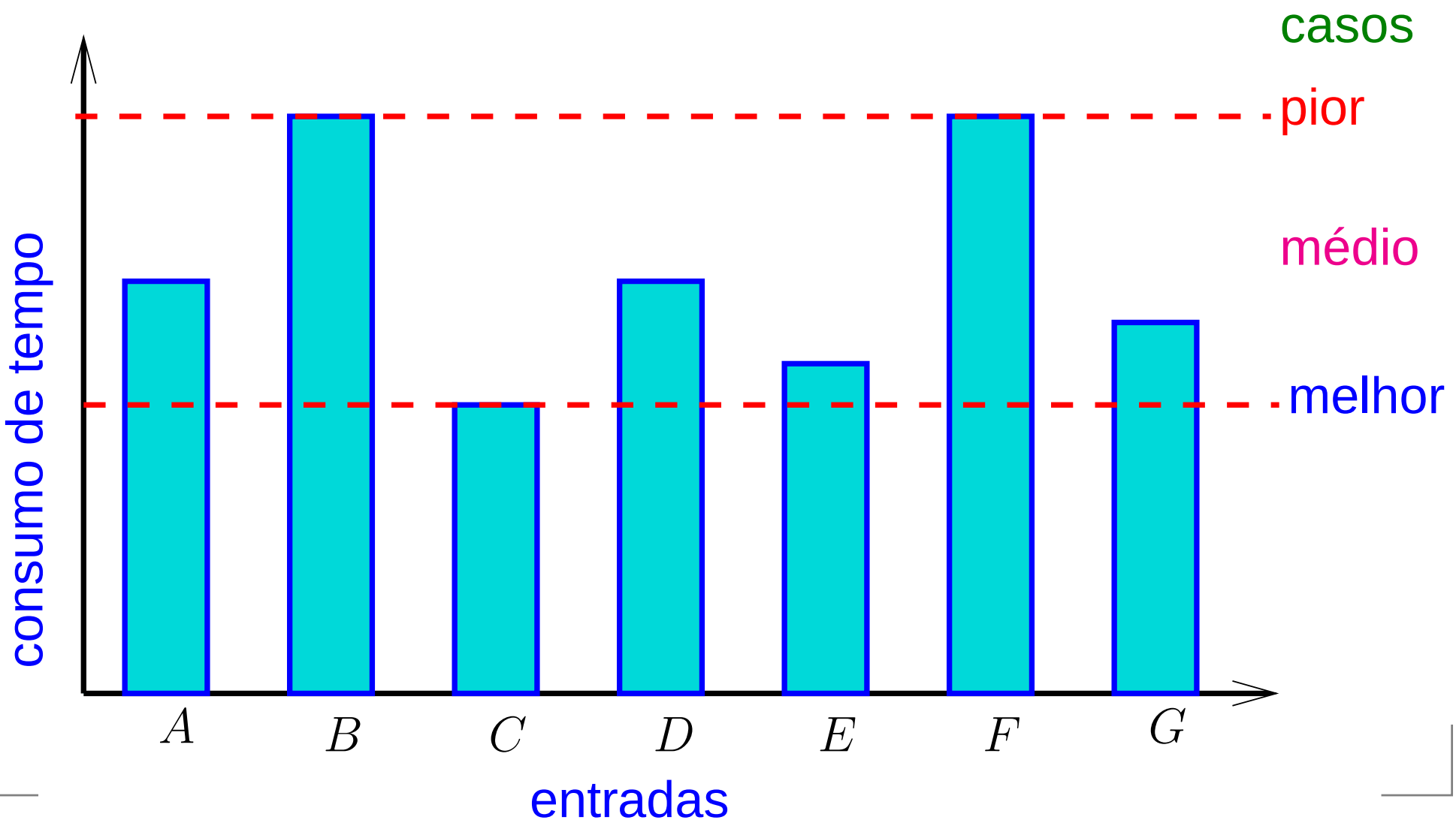
Rearranja $A[p\,..\,d]$, com $p \leq d$, em ordem crescente.

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

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

Consumo de tempo

Mínimo, médio ou máximo?
Melhor caso, caso médio, pior caso?



Mergesort

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

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

Consumo de tempo?

Mergesort

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

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

Consumo de tempo?

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

Mergesort

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

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

$T(n) = ?$

Mergesort

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

linha	consumo na linha
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

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

Mergesort

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

$$T(1) = \Theta(1)$$

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

Mergesort

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

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

Solução: $T(n)$ é $\Theta(???)$.

Demonstração: ...

Mergesort

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

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

Solução: $T(n)$ é $\Theta(???)$.

Demonstração: ...

Veremos, mas antes estudaremos recorrências.