

Análise de Algoritmos

CLRS 2.3, 3.2, 4.1 e 4.2

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Número de inversões

Problema: Dada uma permutação $p[1 \dots n]$, determinar o número de inversões em p .

Queremos um algoritmo $O(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Número de inversões

Problema: Dada uma permutação $p[1 \dots n]$, determinar o número de inversões em p .

Queremos um algoritmo $O(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos **ordenar e contar** ao mesmo tempo!

A ordenação ajuda a contar várias inversões de uma só vez.

Número de inversões

Problema: Dada uma permutação $p[1 \dots n]$, determinar o número de inversões em p .

Queremos um algoritmo $O(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos **ordenar e contar** ao mesmo tempo!

A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Número de inversões

Problema: Dada uma permutação $p[1..n]$, determinar o número de inversões em p .

Queremos um algoritmo $O(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos **ordenar e contar** ao mesmo tempo!

A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Duas opções: o **MERGESORT** e o **HEAPSORT**.
Qual deles parece mais adequado?

Número de inversões

Problema: Dada uma permutação $p[1..n]$, determinar o número de inversões em p .

Queremos um algoritmo $O(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos **ordenar e contar** ao mesmo tempo!

A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Duas opções: o **MERGESORT** e o **HEAPSORT**.
Qual deles parece mais adequado?

Resposta: o **MERGESORT**.

Merge-Sort

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

MERGESORT (A, p, r)

1 **se** $p < r$

2 **então** $q \leftarrow \lfloor (p + r) / 2 \rfloor$

3 **MERGESORT** (A, p, q)

4 **MERGESORT** ($A, q + 1, r$)

5 **INTERCALA** (A, p, q, r)

Método: Divisão e conquista.

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:

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

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:

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

Sai:

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

Intercalação

INTERCALA (A, p, q, r)

0 $\triangleright B[p..r]$ é um vetor auxiliar

1 **para** $i \leftarrow p$ **até** q **faça**

2 $B[i] \leftarrow A[i]$

3 **para** $j \leftarrow q + 1$ **até** r **faça**

4 $B[r + q + 1 - j] \leftarrow A[j]$

5 $i \leftarrow p$

6 $j \leftarrow r$

7 **para** $k \leftarrow p$ **até** r **faça**

8 **se** $B[i] \leq B[j]$

9 **então** $A[k] \leftarrow B[i]$

10 $i \leftarrow i + 1$

11 **senão** $A[k] \leftarrow B[j]$

12 $j \leftarrow j - 1$

Adaptação do Merge-Sort

Conta o número de inversões de $A[p..r]$, com $p \leq r$, e rearranja $A[p..r]$ em ordem crescente.

CONTA-E-ORDENA (A, p, r)

```
1  se  $p \geq r$ 
2      então devolva 0
3      senão  $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
4           $c \leftarrow$  CONTA-E-ORDENA ( $A, p, q$ ) +
5                  CONTA-E-ORDENA ( $A, q + 1, r$ ) +
6                  CONTA-E-INTERCALA ( $A, p, q, r$ )
7      devolva  $c$ 
```

Método: Divisão e conquista.

Contagem na intercalação

CONTA-E-INTERCALA (A, p, q, r)

1 para $i \leftarrow p$ até q faça

2 $B[i] \leftarrow A[i]$

3 para $j \leftarrow q + 1$ até r faça

4 $B[r + q + 1 - j] \leftarrow A[j]$

5 $i \leftarrow p$

6 $j \leftarrow r$

7 $c \leftarrow 0$

▷ inicializa o contador

8 para $k \leftarrow p$ até r faça

9 se $B[i] \leq B[j]$

10 então $A[k] \leftarrow B[i]$

11 $i \leftarrow i + 1$

12 senão $A[k] \leftarrow B[j]$

13 $j \leftarrow j - 1$

14 $c \leftarrow c + (q - i + 1)$

▷ conta inversões

15 devolva c

Simulação

A

<i>p</i>				<i>q</i>				<i>r</i>
22	33	55	77	99	11	44	66	88

B

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

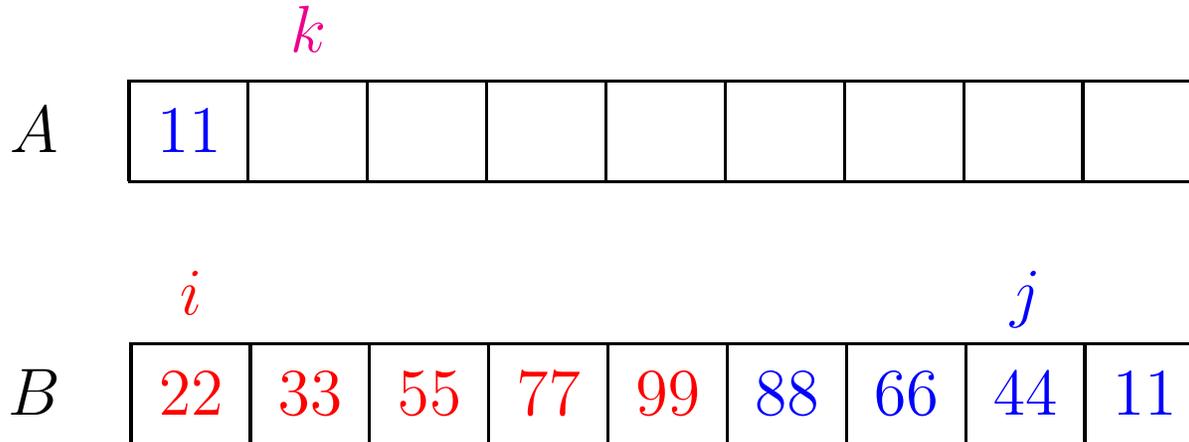
$$c = 0$$

Simulação



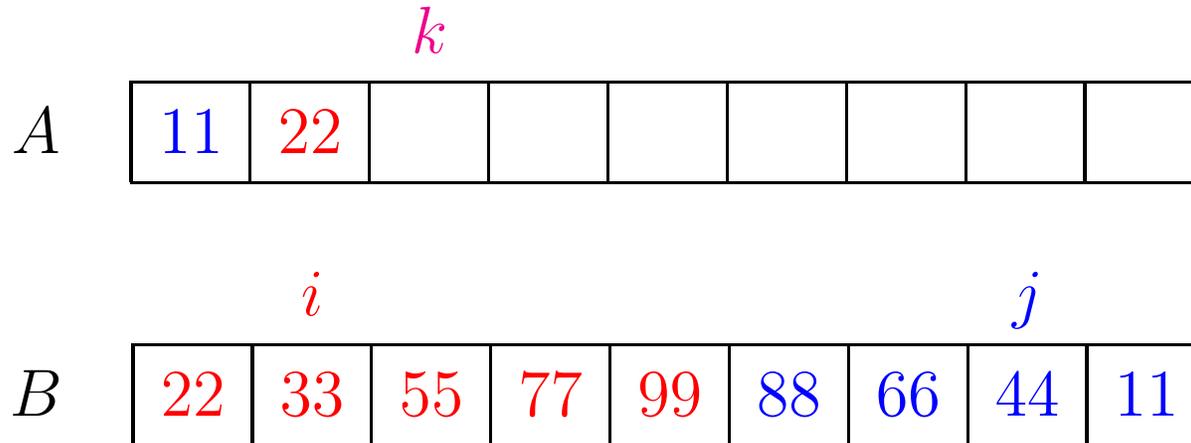
$c = 0$

Simulação



$$c = 0 + 5 = 5$$

Simulação



$$c = 5$$

Simulação

k

A

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

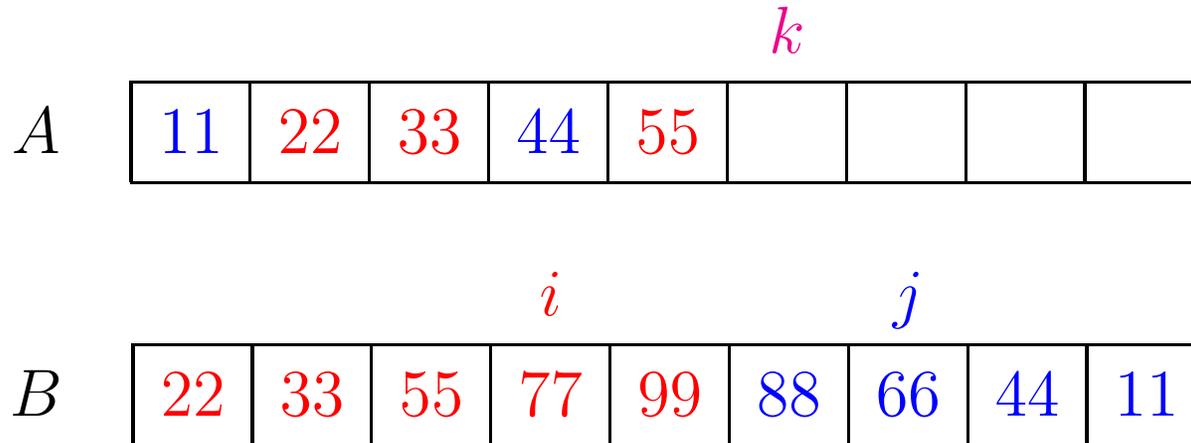
i *j*

B

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

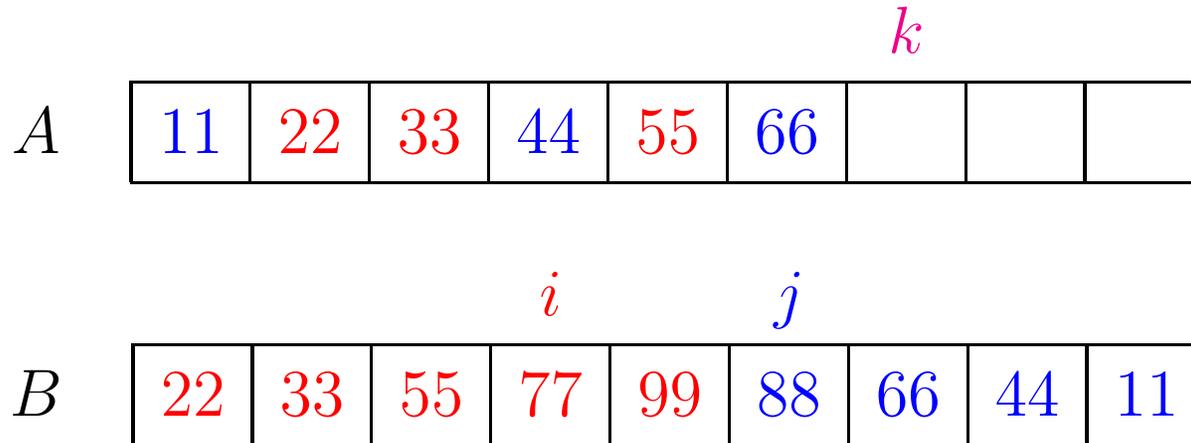
$$c = 5$$

Simulação



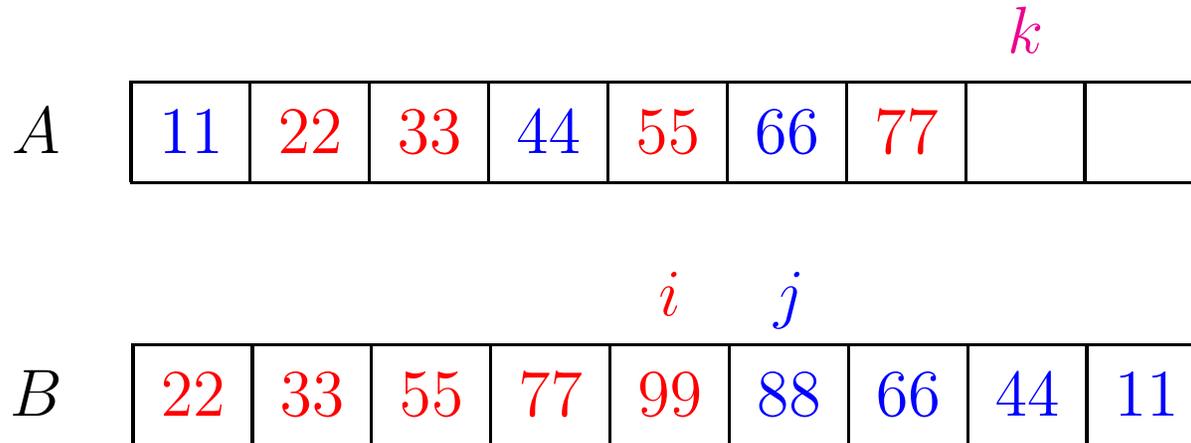
$$c = 8$$

Simulação



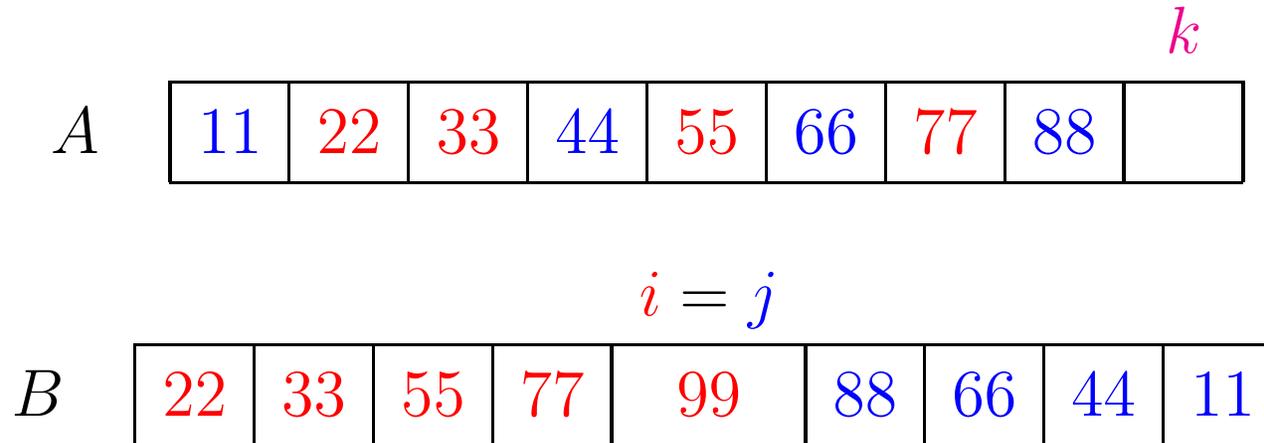
$$c = 8 + 2 = 10$$

Simulação



$$c = 10$$

Simulação



$$c = 10 + 1 = 11$$

Simulação

A

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

B

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

j *i*

$$c = 11$$

Contagem na intercalação

CONTA-E-INTERCALA (A, p, q, r)

1 para $i \leftarrow p$ até q faça

2 $B[i] \leftarrow A[i]$

3 para $j \leftarrow q + 1$ até r faça

4 $B[r + q + 1 - j] \leftarrow A[j]$

5 $i \leftarrow p$

6 $j \leftarrow r$

7 $c \leftarrow 0$

▷ inicializa o contador

8 para $k \leftarrow p$ até r faça

9 se $B[i] \leq B[j]$

10 então $A[k] \leftarrow B[i]$

11 $i \leftarrow i + 1$

12 senão $A[k] \leftarrow B[j]$

13 $j \leftarrow j - 1$

14 $c \leftarrow c + (q - i + 1)$

▷ conta inversões

15 devolva c

Consumo de tempo

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

linha	consumo de todas as execuções da linha
1	$O(n)$
2	$O(n)$
3	$O(n)$
4	$O(n)$
5–7	$O(1)$
8	$O(n)$
9	$O(n)$
10–14	$O(n)$
15	$O(1)$
total	$O(7n + 2) = O(n)$

Conclusão

O algoritmo **CONTA-E-INTERCALA** consome $O(n)$ unidades de tempo.

Também escreve-se

O algoritmo **CONTA-E-INTERCALA** consome tempo $O(n)$.

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo **CONTA-E-ORDENA**.

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo **CONTA-E-ORDENA**.

Vale a seguinte recorrência para $T(n)$:

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

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo **CONTA-E-ORDENA**.

Vale a seguinte recorrência para $T(n)$:

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

Solução: $T(n) = O(n \lg n)$.

Prova?

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo **CONTA-E-ORDENA**.

Vale a seguinte recorrência para $T(n)$:

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

Solução: $T(n) = O(n \lg n)$.

Prova?

Considera-se a recorrência simplificada

$$T(n) = 2T(n/2) + n$$

definida apenas para n potência de 2.

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo **CONTA-E-ORDENA**.

Vale a seguinte recorrência para $T(n)$:

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

Solução: $T(n) = O(n \lg n)$.

Prova?

Considera-se a recorrência simplificada

$$T(n) = 2T(n/2) + n$$

definida apenas para n potência de 2.

Prova-se por indução em n que $T(n) = n + n \lg n = O(n \lg n)$.

Prova

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de } 2 \end{cases}$$

tem como solução $T(n) = n + n \lg n$.

Prova

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de } 2 \end{cases}$$

tem como solução $T(n) = n + n \lg n$.

Prova: Por indução em n .

Base: $n = 1$

$$T(1) = 1 = 1 + 1 \cdot 0 = 1 + 1 \lg 1.$$

Prova

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de } 2 \end{cases}$$

tem como solução $T(n) = n + n \lg n$.

Prova: Por indução em n .

Base: $n = 1$

$$T(1) = 1 = 1 + 1 \cdot 0 = 1 + 1 \lg 1.$$

Passo: $n \geq 2$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(n/2 + (n/2) \lg(n/2)) + n && \text{por indução} \\ &= 2n + n \lg(n/2) \\ &= 2n + n(\lg n - 1) \\ &= n + n \lg n. \end{aligned}$$

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$?

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$? No **chute!**

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$? No chute!

Uma maneira de se obter um “chute” de solução de recorrência é desenrolando a recorrência.

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$? No **chute!**

Uma maneira de se obter um “chute” de solução de recorrência é **desenrolando a recorrência**.

$$\begin{aligned}T(n) &= 2T(n/2) + n \\&= 2(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \\&= 2^2(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n \\&= 2^3(2T(n/2^4) + n/2^3) + 3n = 2^4T(n/2^4) + 4n \\&= \dots \\&= 2^k T(n/2^k) + kn,\end{aligned}$$

onde $k = \lg n$. Disso concluímos que

$$T(n) = n + n \lg n.$$