AULA 9

Quicksort

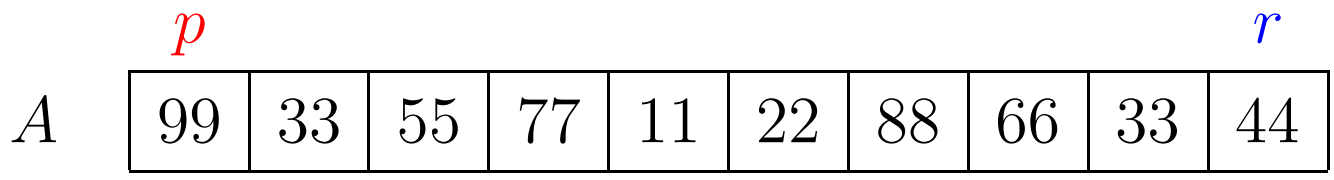
CLRS 7

Partição

Problema: Rearranjar um dado vetor $A[p\,..\,r]$ e devolver um índice q, $p \leq q \leq r$, tais que

$$A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,r]$$

Entra:



Partição

Problema: Rearranjar um dado vetor $A[p\,..\,r]$ e devolver um índice q, $p \leq q \leq r$, tais que

$$A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,r]$$

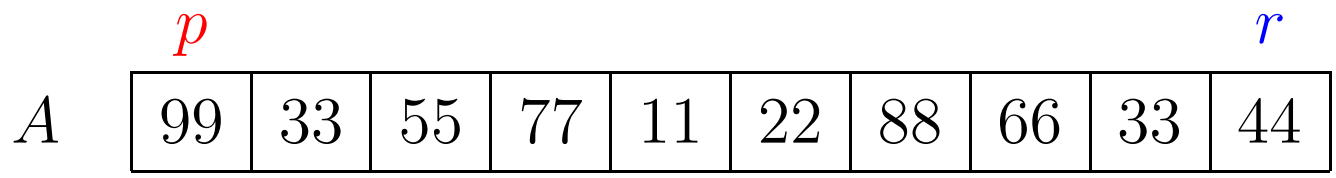
Entra:

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

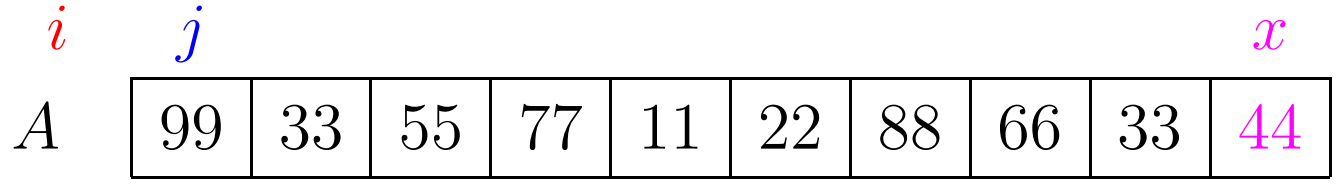
Sai:

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

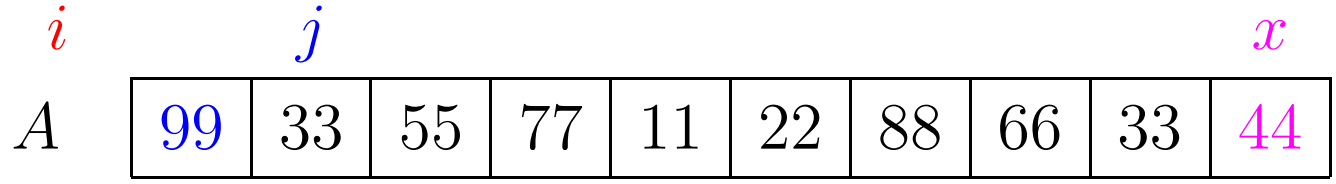
Particione



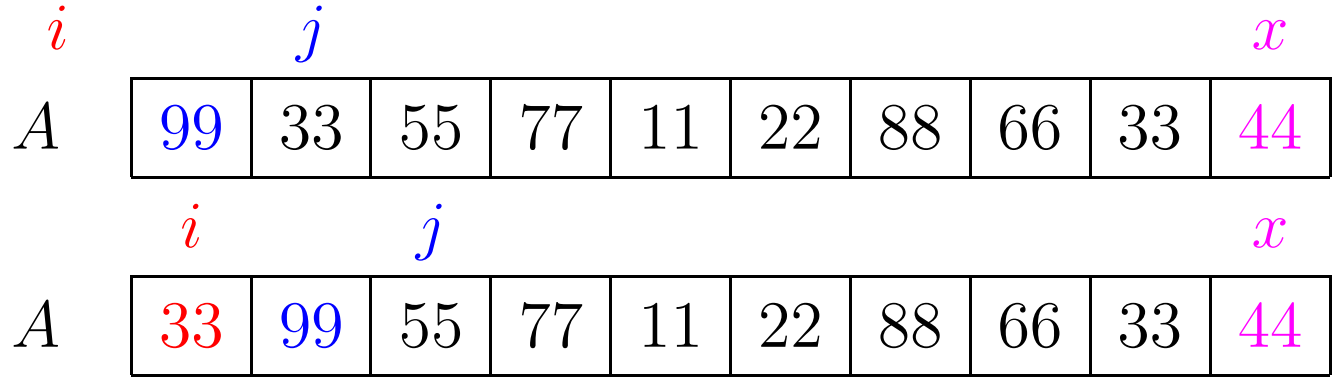
Particione



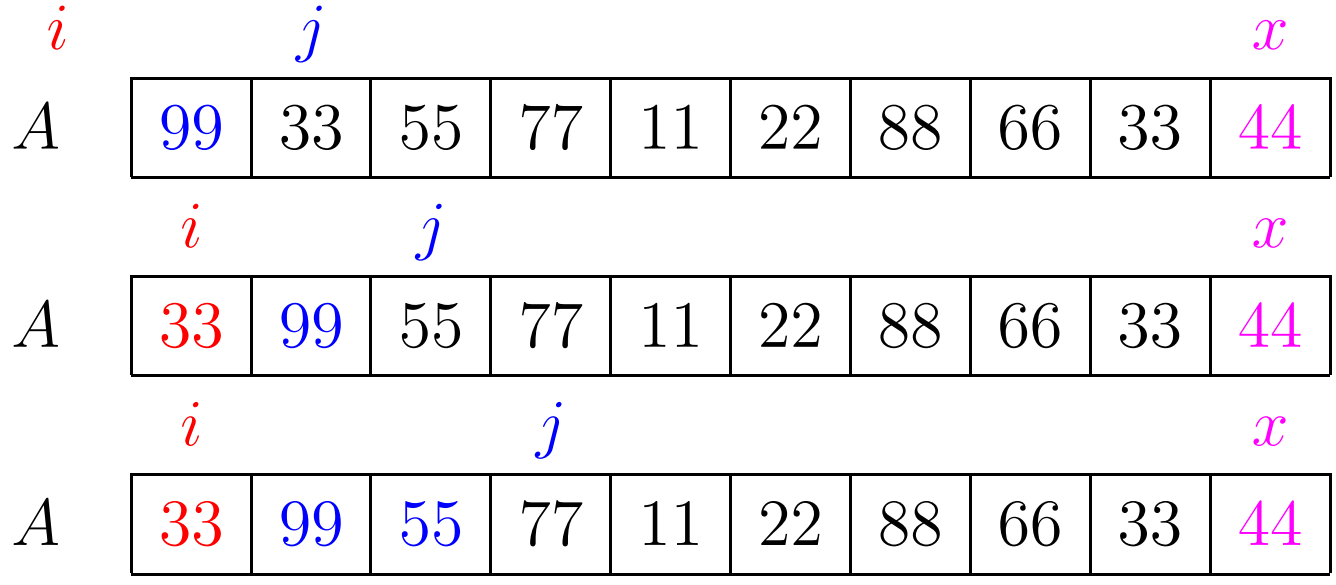
Particione



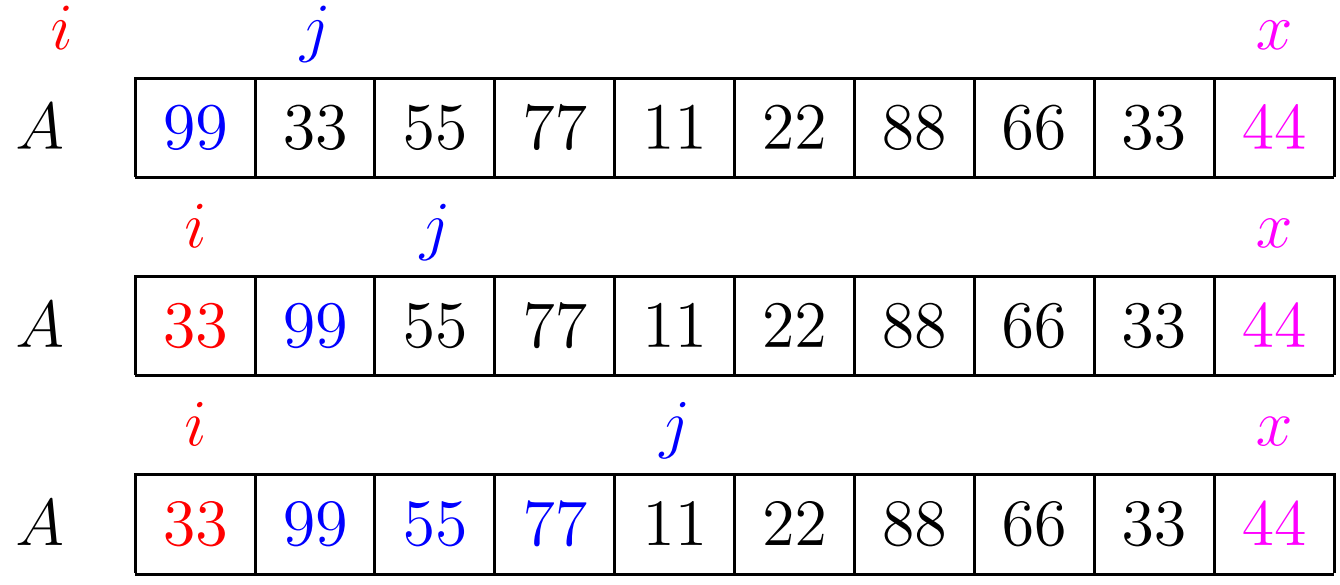
Particione



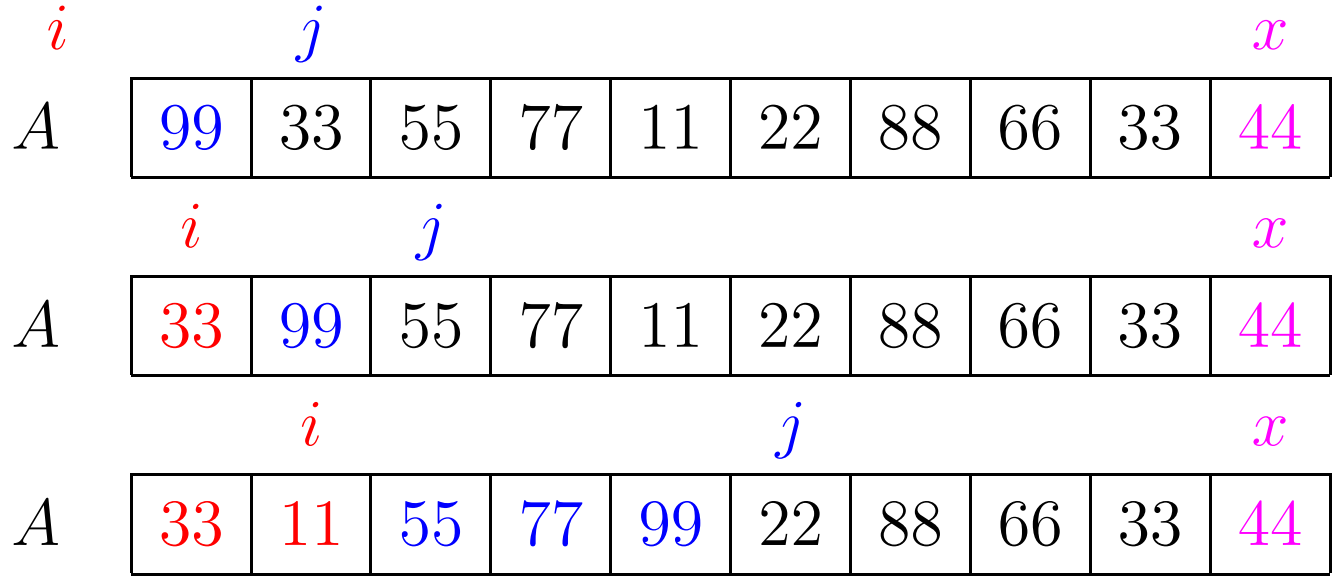
Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44

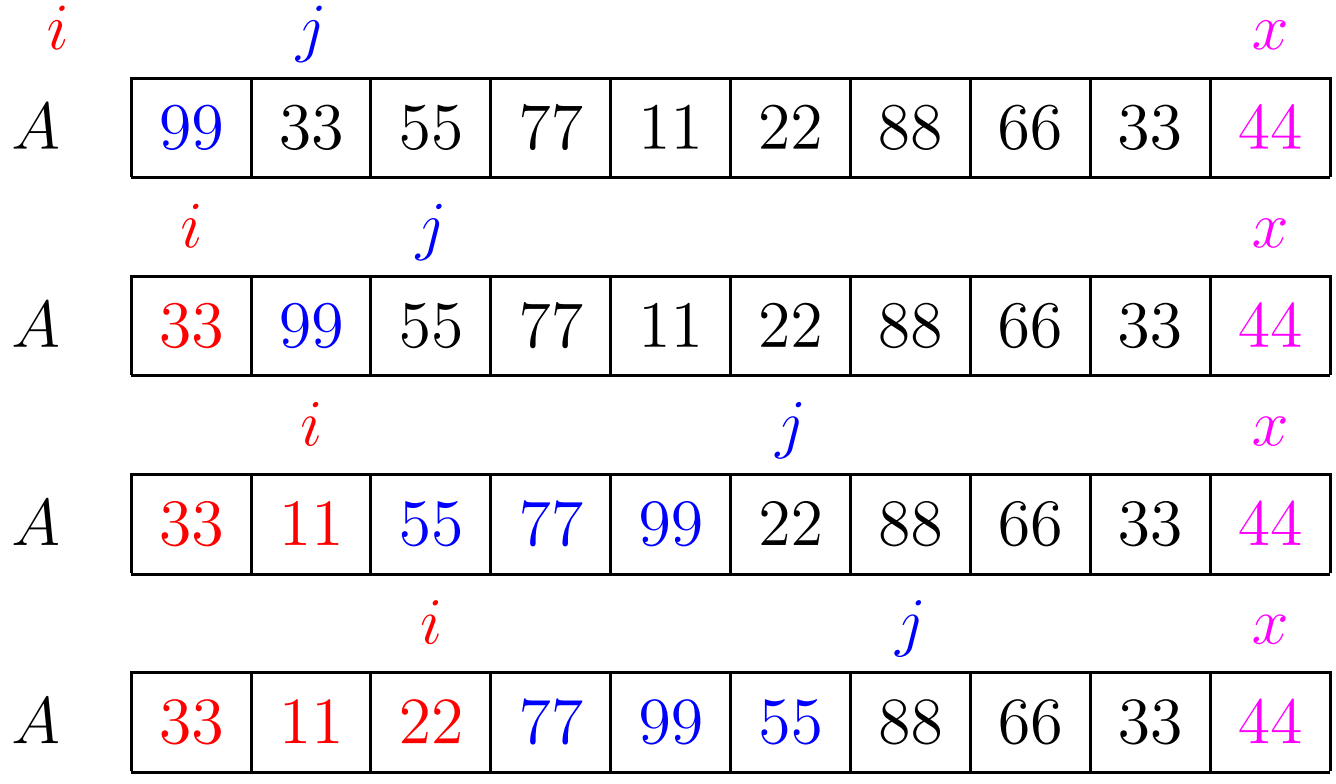
Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44

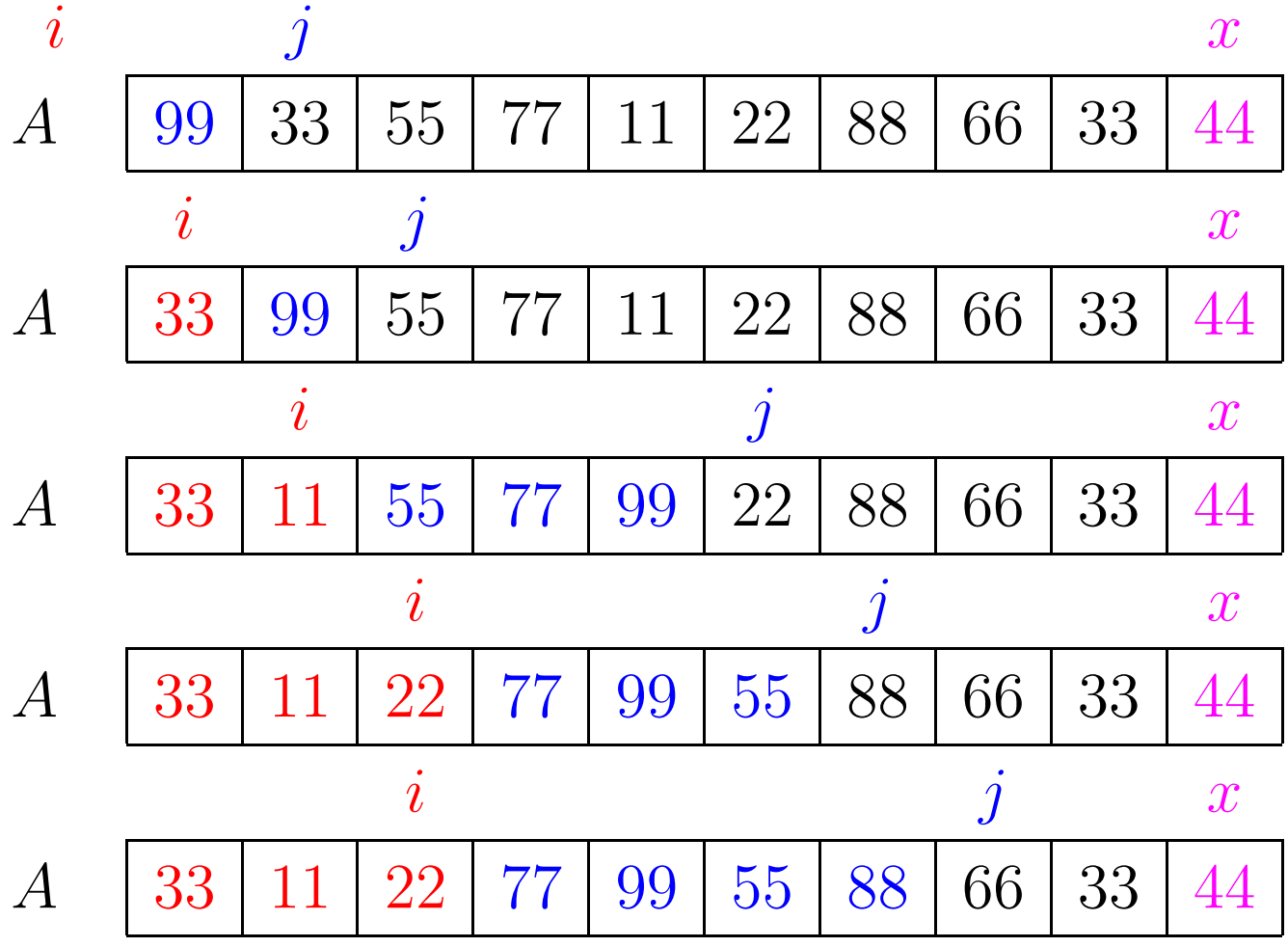
Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 11 55 77 99 22 88 66 33 44

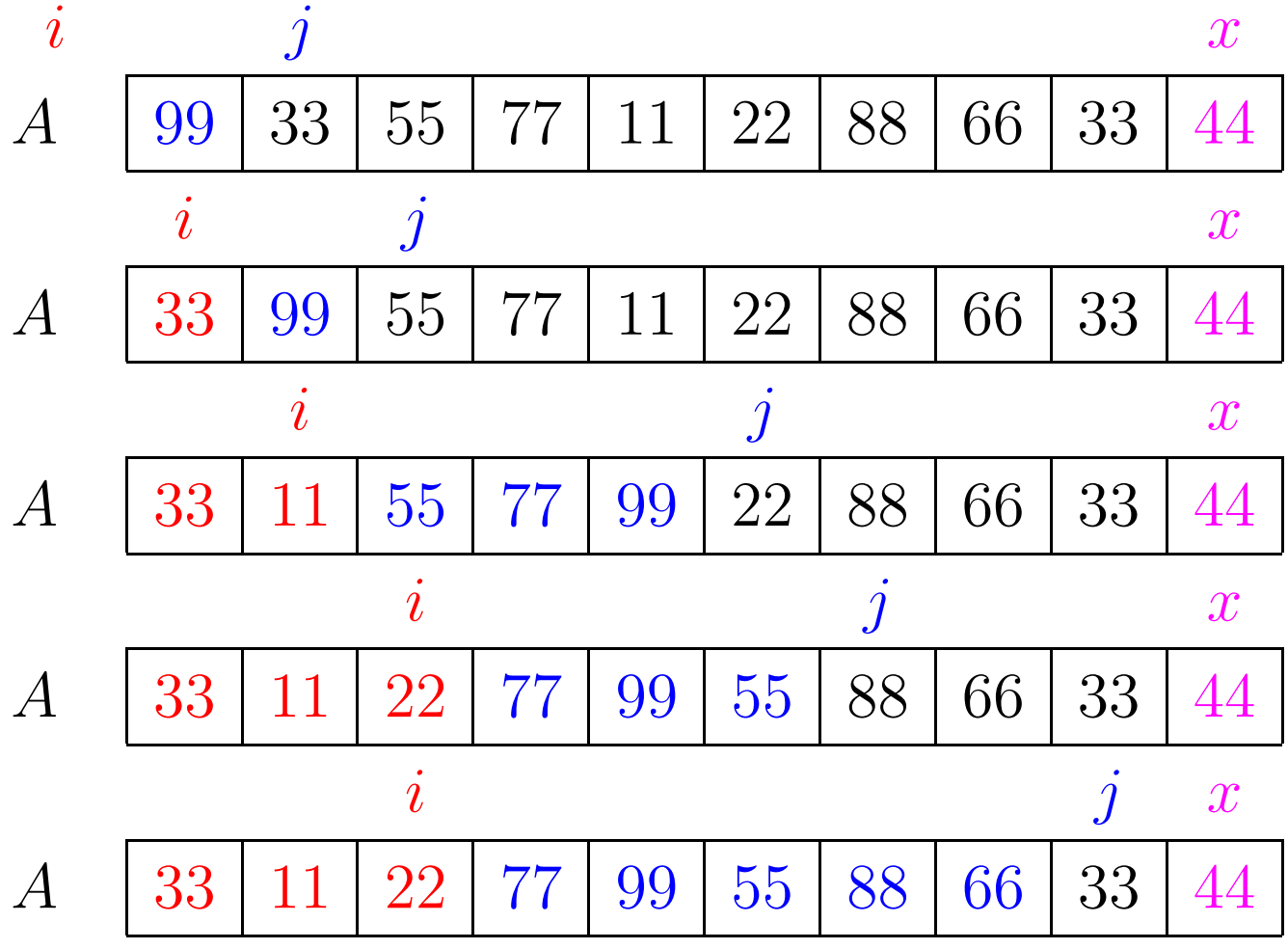
Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 11 55 77 99 22 88 66 33 44
i
j
x
A
33 11 22 77 99 55 88 66 33 44

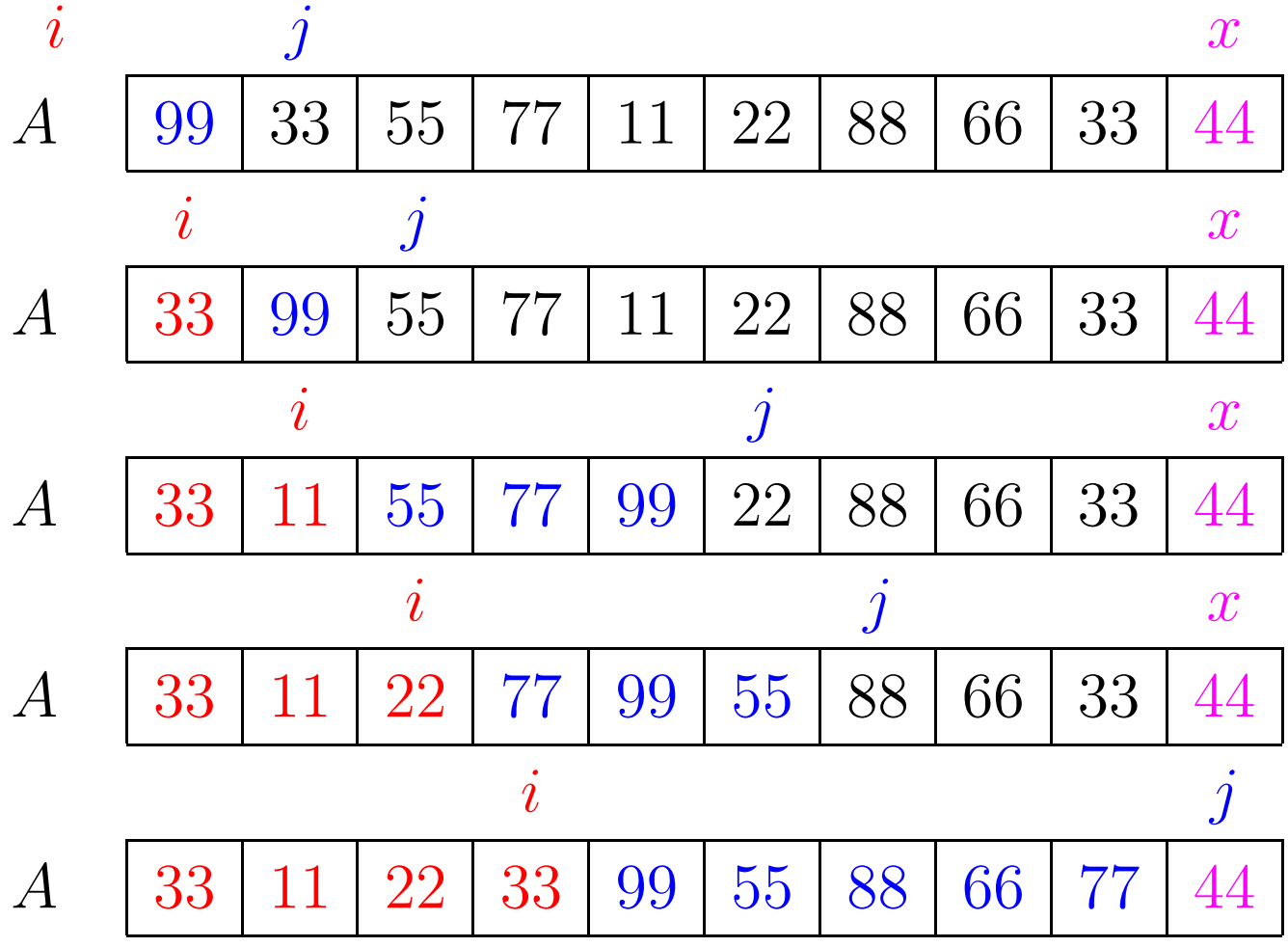
Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 11 55 77 99 22 88 66 33 44
i
j
x
A
33 11 22 77 99 55 88 66 33 44
i
j
x
A
33 11 22 77 99 55 88 66 33 44

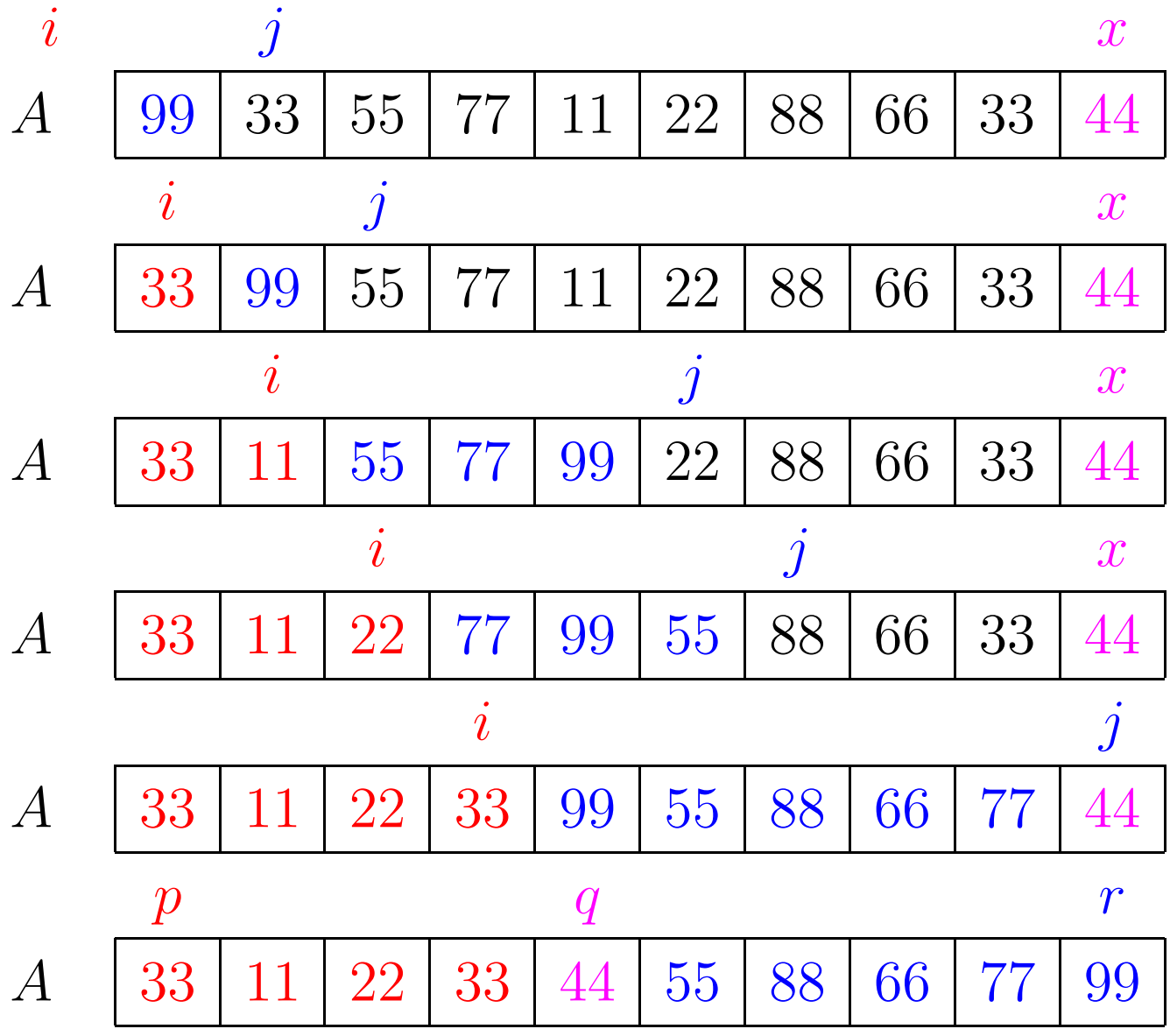
Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 11 55 77 99 22 88 66 33 44
i
j
x
A
33 11 22 77 99 55 88 66 33 44
i
j
x
A
33 11 22 77 99 55 88 66 33 44

Particione


i
j
x
A
99 33 55 77 11 22 88 66 33 44
i
j
x
A
33 99 55 77 11 22 88 66 33 44
i
j
x
A
33 11 55 77 99 22 88 66 33 44
i
j
x
A
33 11 22 77 99 55 88 66 33 44
i
j
A
33 11 22 33 99 55 88 66 77 44

Particione


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

Particione

Rearranja $A[p\,..\,r]$ de modo que $p \le q \le r$ e
$A[p\,..\,q{-}1] \le A[q] < A[q{+}1\,..\,r]$

```
PARTICIONE (A, p, r)
1   x ← A[r]          ▷ x é o "pivô"
2   i ← p−1
3   para j ← p até r − 1 faça
4       se A[j] ≤ x
5           então i ← i + 1
6                 A[i] ↔ A[j]
7   A[i+1] ↔ A[r]
8   devolva i + 1
```

Invariantes: no começo de cada iteração de 3–6,

(i0) $A[p\,..\,i] \le x$ (i1) $A[i{+}1\,..\,j{-}1] > x$ (i2) $A[r] = x$

Consumo de tempo

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

linha	consumo de todas as execuções da linha
1-2	$= 2\,\Theta(1)$
3	$= \Theta(n)$
4	$= \Theta(n)$
5-6	$= 2\,\mathrm{O}(n)$
7-8	$= 2\,\Theta(1)$
total	$= \Theta(2n+4) + \mathrm{O}(2n) \qquad = \Theta(n)$

Conclusão:

O algoritmo PARTICIONE consome tempo $\Theta(n)$.

Quicksort

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

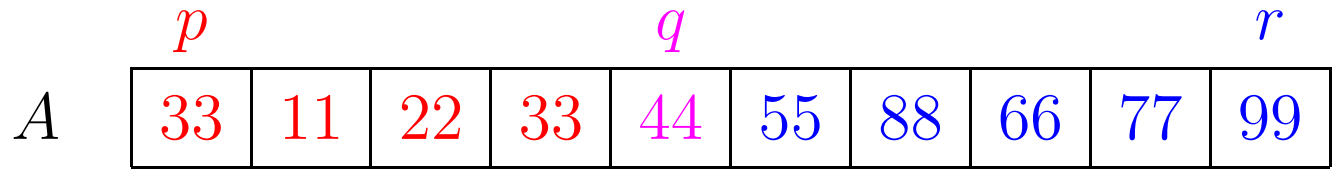
```
QUICKSORT (A, p, r)
1   se p < r
2       então q ← PARTICIONE (A, p, r)
3             QUICKSORT (A, p, q − 1)
4             QUICKSORT (A, q + 1, r)
```

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

Quicksort

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

```
QUICKSORT (A, p, r)
1    se p < r
2         então q ← PARTICIONE (A, p, r)
3                 QUICKSORT (A, p, q − 1)
4                 QUICKSORT (A, q + 1, r)
```



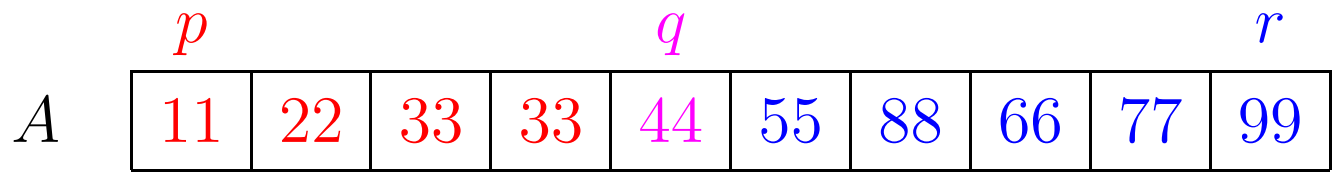
No começo da linha 3,

$$A[p\,..\,q{-}1] \leq A[q] \leq A[q{+}1\,..\,r]$$

Quicksort

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

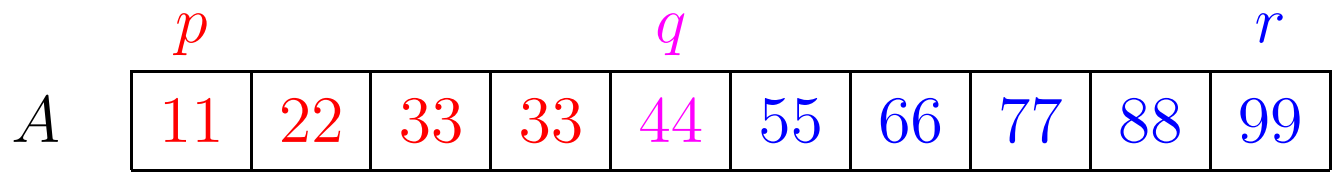
```
QUICKSORT (A, p, r)
1   se p < r
2       então q ← PARTICIONE (A, p, r)
3             QUICKSORT (A, p, q − 1)
4             QUICKSORT (A, q + 1, r)
```



Quicksort

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

```
QUICKSORT (A, p, r)
1    se p < r
2         então q ← PARTICIONE (A, p, r)
3               QUICKSORT (A, p, q − 1)
4               QUICKSORT (A, q + 1, r)
```



Quicksort

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

```
QUICKSORT (A, p, r)
1    se p < r
2         então q ← PARTICIONE (A, p, r)
3               QUICKSORT (A, p, q − 1)
4               QUICKSORT (A, q + 1, r)
```

No começo da linha 3,

$$A[p\,..\,q{-}1] \leq A[q] \leq A[q{+}1\,..\,r]$$

Consumo de tempo?

Quicksort

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

```
QUICKSORT (A, p, r)
1    se p < r
2         então q ← PARTICIONE (A, p, r)
3               QUICKSORT (A, p, q − 1)
4               QUICKSORT (A, q + 1, r)
```

No começo da linha 3,

$$A[p\,..\,q{-}1] \leq A[q] \leq A[q{+}1\,..\,r]$$

Consumo de tempo?

$T(n) :=$ consumo de tempo no pior caso sendo

$$n := r - p + 1$$

Consumo de tempo

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

linha		consumo de todas as execuções da linha
1	$=$	?
2	$=$	?
3	$=$	?
4	$=$	?
total	$=$	????

Consumo de tempo

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

linha		consumo de todas as execuções da linha
1	$=$	$\Theta(1)$
2	$=$	$\Theta(n)$
3	$=$	$T(k)$
4	$=$	$T(n-k-1)$
total	$=$	$T(k) + T(n-k-1) + \Theta(n+1)$

$$0 \leq k := q - p \leq n - 1$$

Recorrência

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

$$T(0) = \Theta(1)$$
$$T(1) = \Theta(1)$$
$$T(n) = T(k) + T(n - k - 1) + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Recorrência

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

$$T(0) = \Theta(1)$$
$$T(1) = \Theta(1)$$
$$T(n) = T(k) + T(n - k - 1) + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Recorrência grosseira:

$$T(n) = T(0) + T(n - 1) + \Theta(n)$$

$T(n)$ é $\Theta(???)$.

Recorrência

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

$$T(0) = \Theta(1)$$
$$T(1) = \Theta(1)$$
$$T(n) = T(k) + T(n - k - 1) + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Recorrência grosseira:

$$T(n) = T(0) + T(n - 1) + \Theta(n)$$

$T(n)$ é $\Theta(n^2)$.

Demonstração: ...

Recorrência cuidadosa

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

$$T(0) = \Theta(1)$$

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

$$T(n) = \max_{0 \leq k \leq n-1} \{T(k) + T(n-k-1)\} + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Exemplo concreto

$$S(0) = 1$$

$$S(1) = 1$$

$$S(n) = \max_{0 \leq k \leq n-1} \{S(k) + S(n-k-1)\} + n \quad \text{para } n = 3, 4, \ldots$$

n	0	1	2	3	4	5
$S(n)$	1	1	$2+2$	$5+3$	$9+4$	$14+5$

Exemplo concreto

$$S(0) = 1$$

$$S(1) = 1$$

$$S(n) = \max_{0 \leq k \leq n-1} \{S(k) + S(n-k-1)\} + n \quad \text{para } n = 3, 4, \ldots$$

n	0	1	2	3	4	5
$S(n)$	1	1	$2+2$	$5+3$	$9+4$	$14+5$

Vou mostrar que $S(n) \leq n^2 + 1$ para $n \geq 0$.

Demonstração

Prova: Trivial para $n \leq 1$. Se $n \geq 2$ então

$$
\begin{aligned}
S(n) &= \max_{0 \leq k \leq n-1} \left\{ S(k) + S(n-k-1) \right\} + n \\
&\overset{\text{hi}}{\leq} \max_{0 \leq k \leq n-1} \left\{ k^2 + 1 + (n-k-1)^2 + 1 \right\} + n \\
&= (n-1)^2 + 2 + n \qquad \triangleright \text{ exerc 14.E} \\
&= n^2 - n + 3 \\
&\leq n^2 + 1 \,.
\end{aligned}
$$

Prove que $S(n) \geq \frac{1}{2}\, n^2$ para $n \geq 1$.

Algumas conclusões

$T(n)$ é $\Theta(n^2)$.

O consumo de tempo do QUICKSORT no pior caso é $\mathrm{O}(n^2)$.

O consumo de tempo do QUICKSORT é $\mathrm{O}(n^2)$.

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = r - p + 1$

$$M(0) = \Theta(1)$$

$$M(1) = \Theta(1)$$

$$M(n) = \min_{0 \leq k \leq n-1} \{M(k) + M(n-k-1)\} + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = r - p + 1$

$$M(0) = \Theta(1)$$

$$M(1) = \Theta(1)$$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Mostre que, para $n \geq 1$,

$$M(n) \geq \frac{(n-1)}{2} \lg \frac{n-1}{2}.$$

Isto implica que no melhor caso o QUICKSORT é $\Omega(n \lg n)$.

Que é o mesmo que dizer que o QUICKSORT é $\Omega(n \lg n)$.

Quicksort no melhor caso

No melhor caso k é aproximadamente $(n-1)/2$.

$$R(n) = R(\left\lfloor \frac{n-1}{2} \right\rfloor) + R(\left\lceil \frac{n-1}{2} \right\rceil) + \Theta(n)$$

Solução: $R(n)$ é $\Theta(n \lg n)$.

Mais algumas conclusões

$M(n)$ é $\Theta(n \lg n)$.

O consumo de tempo do QUICKSORT no melhor caso é $\Omega(n \log n)$.

Na verdade . . .

O consumo de tempo do QUICKSORT no melhor caso é $\Theta(n \log n)$.

Discussão geral

Pior caso, melhor caso, todos os casos?!?!

Dado um algoritmo $\mathcal{A}$ o que significam as expressões:

- $\mathcal{A}$ é $\mathrm{O}(n^2)$ no pior caso.
- $\mathcal{A}$ é $\mathrm{O}(n^2)$ no melhor caso.
- $\mathcal{A}$ é $\mathrm{O}(n^2)$.
- $\mathcal{A}$ é $\Omega(n^2)$ no melhor caso.
- $\mathcal{A}$ é $\Omega(n^2)$ no pior caso.
- $\mathcal{A}$ é $\Omega(n^2)$

Análise experimental

Algoritmos implementados:

`shell`	`shellsort` original (Knuth).
`merge_r`	MERGE-SORT recursivo.
`merge_i`	MERGE-SORT iterativo.
`heap`	HEAPSORT.
`quick`	QUICKSORT recursivo.
`qsort`	`quicksort` da biblioteca do C.

Compilador: `gcc -O2`.

Computador: Pentium II, 233Mhz, 128Mb.

Estudo empírico (aleatório)

n	shell	merge_r	merge_i	heap	quick	qsort
4096	0.01	0.01	0.00	0.01	0.00	0.01
8192	0.01	0.01	0.01	0.01	0.01	0.03
16384	0.03	0.02	0.02	0.02	0.02	0.06
32768	0.07	0.05	0.05	0.04	0.04	0.12
65536	0.16	0.11	0.11	0.10	0.07	0.25
131072	0.38	0.25	0.33	0.23	0.16	0.54
262144	0.92	0.54	0.73	0.54	0.34	1.15
524288	2.13	1.18	1.55	1.31	0.74	2.44
1048576	4.97	2.52	3.32	3.06	1.60	5.20
2097152	11.38	5.42	6.96	7.07	3.38	10.88
4194304	26.50	11.40	14.46	15.84	7.16	22.78
8388608	61.68	24.35	29.76	35.85	15.24	47.85

Estudo empírico (decrescente)

n	shell	merge_r	merge_i	heap	quick	qsort
4096	0.00	0.01	0.00	0.00	0.22	0.01
8192	0.00	0.01	0.01	0.01	0.90	0.02
16384	0.02	0.02	0.02	0.01	3.70	0.04
32768	0.03	0.04	0.04	0.03	15.29	0.08
65536	0.07	0.09	0.09	0.07	62.23	0.18
131072	0.15	0.21	0.28	0.16	261.26	0.40

Estudo empírico (crescente)

n	shell	merge_r	merge_i	heap	quick	qsort
4096	0.01	0.00	0.01	0.00	0.29	0.00
8192	0.00	0.01	0.01	0.01	1.16	0.02
16384	0.01	0.02	0.02	0.01	4.63	0.03
32768	0.02	0.05	0.04	0.03	18.47	0.06
65536	0.05	0.10	0.10	0.07	74.16	0.11
131072	0.12	0.23	0.28	0.16	372.45	0.23

Exercícios

Exercício 14.A
Submeta ao algoritmo PARTICIONE um vetor com n elementos iguais. Como o algoritmo permuta o vetor recebido? Quantas trocas faz (linhas 6 e 7) entre elementos do vetor?

Exercício 14.B [CLRS 7.2-2]
Qual o consumo de tempo do QUICKSORT quando aplicado a um vetor com n elementos iguais?

Exercício 14.C [CLRS 7.2-3]
Mostre que o consumo de tempo do QUICKSORT é $\Omega(n^2)$ quando aplicado a um vetor crescente com n elementos distintos.

Exercício 14.D [CLRS 7.4-1, modificado]
Seja S a função definida sobre os inteiros positivos pela seguinte recorrência:
$S(0) = S(1) = 1$ e

$$S(n) = \max_{0 \le k \le n-1} \{S(k) + S(n-k-1)\} + n$$

quando $n \geq 2$. Mostre que $S(n) \geq \frac{1}{2}n^2$ para $n \geq 1$.

Mais exercícios

Exercício 14.E [CLRS 7.4-3]

Mostre que $k^2 + (n - k - 1)^2$ atinge o máximo para $0 \leq k \leq n - 1$ quando $k = 0$ ou $k = n - 1$.

Exercício 14.F

É verdade que $\lceil 2\lceil 2n/3 \rceil / 3 \rceil = \lceil 4n/9 \rceil$? É verdade que $\lfloor 2\lfloor 2n/3 \rfloor / 3 \rfloor = \lfloor 4n/9 \rfloor$?

Exercício 14.G [CLRS 7-4]

Considere a seguinte variante do algoritmo Quicksort:

QUICKSORT′ (A, p, r)
  enquanto $p < r$ faça
    $q \leftarrow$ PARTICIONE (A, p, r)
    QUICKSORT′ $(A, p, q - 1)$
    $p \leftarrow q + 1$

Mostre que a pilha de recursão pode atingir altura proporcional a n, onde $n := r - p + 1$. Modifique o código de modo que a pilha de recursão tenha altura $\mathrm{O}(\lg n)$. (Veja enunciado completo em CLRS p.162.)