

i -ésimo menor elemento

CLRS 9

i -ésimo menor

Problema: Encontrar o i -ésimo menor elemento de $A[1 \dots n]$.

Suponha $A[1 \dots n]$ sem elementos repetidos.

Exemplo: 33 é o 4o. menor elemento de:

1									10
22	99	32	88	34	33	11	97	55	66

A

1			4						10
11	22	32	33	34	55	66	88	97	99

ordenado

Mediana

Mediana é o $\lfloor \frac{n+1}{2} \rfloor$ -ésimo menor ou o $\lceil \frac{n+1}{2} \rceil$ -ésimo menor elemento.

Exemplo: a mediana é 34 ou 55:

1									10
22	99	32	88	34	33	11	97	55	66

A

1				5	6				10
11	22	32	33	34	55	66	88	97	99

ordenado

i -ésimo menor

Recebe $A[1..n]$ e i tal que $1 \leq i \leq n$
e devolve valor do i -ésimo menor elemento de $A[1..n]$.

```
SELECT-ORD ( $A, n, i$ )  
1  ORDENE ( $A, n$ )  
2  devolva  $A[i]$ 
```

O consumo de tempo do SELECT-ORD é $\Theta(n \lg n)$.

i -ésimo menor

Recebe $A[1..n]$ e i tal que $1 \leq i \leq n$
e devolve valor do i -ésimo menor elemento de $A[1..n]$.

```
SELECT-ORD ( $A, n, i$ )  
1  ORDENE ( $A, n$ )  
2  devolva  $A[i]$ 
```

O consumo de tempo do SELECT-ORD é $\Theta(n \lg n)$.

Dá para fazer melhor?

Menor

Recebe um vetor $A[1..n]$ e devolve o valor do **menor** elemento.

MENOR (A, n)

```
1  menor  $\leftarrow A[1]$ 
2  para  $k \leftarrow 2$  até  $n$  faça
3      se  $A[k] < \text{menor}$ 
4          então menor  $\leftarrow A[k]$ 
5  devolva menor
```

O consumo de tempo do algoritmo **MENOR** é $\Theta(n)$.

Segundo menor

Recebe um vetor $A[1..n]$ e devolve o valor do **segundo menor** elemento, supondo $n \geq 2$.

SEG-MENOR (A, n)

```
1  menor ← min{A[1], A[2]}    segmenor ← max{A[1], A[2]}
2  para k ← 3 até n faça
3      se A[k] < menor
4          então segmenor ← menor
5              menor ← A[k]
6      senão se A[k] < segmenor
7          então segmenor ← A[k]
8  devolva segmenor
```

O consumo de tempo do **SEG-MENOR** é $\Theta(n)$.

Algoritmo linear?

Será que conseguimos fazer um **algoritmo linear**
para a mediana?
para o i -ésimo menor?

Algoritmo linear?

Será que conseguimos fazer um **algoritmo linear**
para a mediana?
para o i -ésimo menor?

Sim!

Usaremos o PARTICIONE do QUICKSORT!

Particione

Rearranja $A[p..r]$ de modo que $p \leq q \leq r$ e
 $A[p..q-1] \leq A[q] < A[q+1..r]$

PARTICIONE (A, p, r)

- 1 $x \leftarrow A[r]$ $\triangleright x$ é o “pivô”
- 2 $i \leftarrow p-1$
- 3 **para** $j \leftarrow p$ até $r-1$ **faça**
- 4 **se** $A[j] \leq x$
- 5 **então** $i \leftarrow i + 1$
- 6 $A[i] \leftrightarrow A[j]$
- 7 $A[i+1] \leftrightarrow A[r]$
- 8 **devolva** $i + 1$

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

Particione

Rearranja $A[p..r]$ de modo que $p \leq q \leq r$ e
 $A[p..q-1] \leq A[q] < A[q+1..r]$

PARTICIONE (A, p, r)

- 1 $x \leftarrow A[r]$ $\triangleright x$ é o “pivô”
- 2 $i \leftarrow p-1$
- 3 para $j \leftarrow p$ até $r-1$ faça
- 4 se $A[j] \leq x$
- 5 então $i \leftarrow i + 1$
- 6 $A[i] \leftrightarrow A[j]$
- 7 $A[i+1] \leftrightarrow A[r]$
- 8 devolva $i + 1$

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

Particione

Rearranja $A[p..r]$ de modo que $p \leq q \leq r$ e
 $A[p..q-1] \leq A[q] < A[q+1..r]$

PARTICIONE (A, p, r)

```
1   $x \leftarrow A[r]$       ▷  $x$  é o “pivô”
2   $i \leftarrow p-1$ 
3  para  $j \leftarrow p$  até  $r-1$  faça
4      se  $A[j] \leq x$ 
5          então  $i \leftarrow i+1$ 
6               $A[i] \leftrightarrow A[j]$ 
7   $A[i+1] \leftrightarrow A[r]$ 
8  devolva  $i+1$ 
```

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

Algoritmo SELECT

Recebe $A[p..r]$ e i tal que $1 \leq i \leq r-p+1$
e devolve valor do i -ésimo menor elemento de $A[p..r]$.

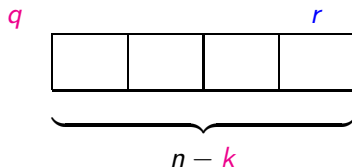
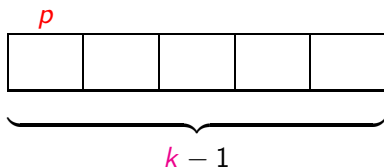
SELECT(A, p, r, i)

```
1  se  $p = r$ 
2      então devolva  $A[p]$ 
3   $q \leftarrow$  PARTICIONE( $p, r$ )
4   $k \leftarrow q - p + 1$ 
5  se  $k = i$ 
6      então devolva  $A[q]$ 
7  se  $k > i$ 
8      então devolva SELECT( $A, p, q - 1, i$ )
9      senão devolva SELECT( $A, q + 1, r, i - k$ )
```

Algoritmo SELECT

SELECT(A, p, r, i)

```
1  se  $p = r$ 
2      então devolva  $A[p]$ 
3   $q \leftarrow$  PARTICIONE( $A, p, r$ )
4   $k \leftarrow q - p + 1$ 
5  se  $k = i$ 
6      então devolva  $A[q]$ 
7  se  $k > i$ 
8      então devolva SELECT( $A, p, q - 1, i$ )
9  senão devolva SELECT( $A, q + 1, r, i - k$ )
```



Consumo de tempo

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

linha	consumo de todas as execuções da linha
1-2	$= \Theta(1)$
3	$= \Theta(n)$
4-7	$= \Theta(1)$
8	$= T(k - 1)$
9	$= T(n - k)$

$$T(n) = \Theta(n) + \max\{T(k - 1), T(n - k)\}$$

Consumo de tempo

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

linha	consumo de todas as execuções da linha
1-2	$= \Theta(1)$
3	$= \Theta(n)$
4-7	$= \Theta(1)$
8	$= T(k - 1)$
9	$= T(n - k)$

$$T(n) = \Theta(n) + \max\{T(k - 1), T(n - k)\}$$

Pior caso: $T(n) = \Theta(n) + T(n - 1)$

Consumo de tempo

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

linha	consumo de todas as execuções da linha
1-2	$= \Theta(1)$
3	$= \Theta(n)$
4-7	$= \Theta(1)$
8	$= T(k - 1)$
9	$= T(n - k)$

$$T(n) = \Theta(n) + \max\{T(k - 1), T(n - k)\}$$

Pior caso: $T(n) = \Theta(n) + T(n - 1) = \Theta(n^2)$

Seleção em tempo linear

Como fazer algo melhor?

Seleção em tempo linear

Como fazer algo melhor?

Vamos usar de novo **divisão e conquista**.

Veremos o algoritmo **BFPRT**,
de Blum, Floyd, Pratt, Rivest e Tarjan.

Seleção em tempo linear

Como fazer algo melhor?

Vamos usar de novo **divisão e conquista**.

Veremos o algoritmo **BFPRT**,
de Blum, Floyd, Pratt, Rivest e Tarjan.

Se o pivô do PARTICIONE for a mediana do vetor,
qual seria o consumo de tempo do **SELECT**?

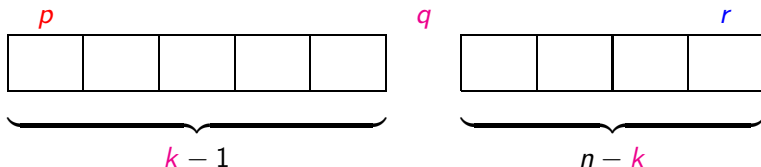
Select-BFPRT

Recebe $A[p..r]$ e i tal que $1 \leq i \leq r-p+1$ e devolve
um índice q tal que $A[q]$ é o i -ésimo menor elemento de $A[p..r]$

SELECT-BFPRT(A, p, r, i)

```
1  se  $p = r$ 
2      então devolva  $p$   $\triangleright p$  e não  $A[p]$ 
3   $q \leftarrow$  PARTICIONE-BFPRT( $A, p, r$ )
4   $k \leftarrow q - p + 1$ 
5  se  $k = i$ 
6      então devolva  $q$   $\triangleright q$  e não  $A[q]$ 
7  se  $k > i$ 
8      então devolva SELECT-BFPRT( $A, p, q - 1, i$ )
9  senão devolva SELECT-BFPRT( $A, q + 1, r, i - k$ )
```

Particione-BFPRT



Rearranja $A[p..r]$ e devolve um índice q com $p \leq q \leq r$, tal que $A[p..q-1] \leq A[q] < A[q+1..r]$ e

$$\max\{k - 1, n - k\} \leq \frac{7n}{10} + 3,$$

onde $n = r - p + 1$ e $k = q - p + 1$.

Suponha que $P(n) :=$ consumo de tempo máximo do algoritmo PARTICIONE-BFPRT quando $n = r - p + 1$

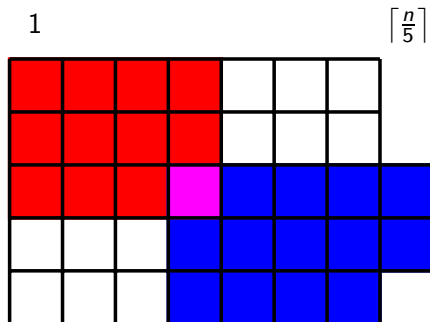
Consumo de tempo

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

linha	consumo de todas as execuções da linha
1-2	$= \Theta(1)$
3	$= P(n)$
4-7	$= \Theta(1)$
8	$= T(k - 1)$
9	$= T(n - k)$

$$\begin{aligned} T(n) &= \Theta(1) + P(n) + \max\{T(k - 1), T(n - k)\} \\ &\leq \Theta(1) + P(n) + T(\lceil \frac{7n}{10} \rceil + 3) \end{aligned}$$

Partizione-BFPRT



$$\begin{aligned}\max\{k - 1, n - k\} &\leq n - \left(3 \left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 3 \right) \\ &\leq n - \left(\frac{3n}{10} - 3 \right) = \frac{7n}{10} + 3\end{aligned}$$

Particione-BFPRT

$n := r - p + 1$

PARTICIONE-BFPRT (A, p, r)

1 **para** $j \leftarrow p, p+5, p+5 \cdot 2, \dots$ até $p+5(\lceil n/5 \rceil - 1)$ **faça**

2 ORDENE ($A, j, j+4$)

3 ORDENE ($A, p+5\lfloor n/5 \rfloor, r$)

4 **para** $j \leftarrow 1$ até $\lceil n/5 \rceil - 1$ **faça**

5 $B[j] \leftarrow A[p+5j-3]$

6 $B[\lceil n/5 \rceil] \leftarrow A[(p+5\lfloor n/5 \rfloor + r)/2]$

7 $k \leftarrow \text{SELECT-BFPRT}(B, 1, \lceil n/5 \rceil, \lfloor (\lceil n/5 \rceil + 1)/2 \rfloor)$

8 $A[r] \leftrightarrow B[k]$

9 **devolva** PARTICIONE (A, p, r)

Particione-BFPRT

$n := r - p + 1$

PARTICIONE-BFPRT (A, p, r)

1 **para** $j \leftarrow p, p+5, p+5 \cdot 2, \dots$ até $p+5(\lceil n/5 \rceil - 1)$ **faça**

2 ORDENE ($A, j, j+4$)

3 ORDENE ($A, p+5\lfloor n/5 \rfloor, r$)

4 **para** $j \leftarrow 1$ até $\lceil n/5 \rceil - 1$ **faça**

5 $B[j] \leftarrow A[p+5j-3]$

6 $B[\lceil n/5 \rceil] \leftarrow A[(p+5\lfloor n/5 \rfloor + r)/2]$

7 $k \leftarrow \text{SELECT-BFPRT}(B, 1, \lceil n/5 \rceil, \lfloor (\lceil n/5 \rceil + 1)/2 \rfloor)$

8 $A[r] \leftrightarrow B[k]$ ▷ Não dá certo...

9 **devolva** PARTICIONE (A, p, r)

Particione-BFPRT

$n := r - p + 1$

PARTICIONE-BFPRT (A, p, r)

1 **para** $j \leftarrow p, p+5, p+5 \cdot 2, \dots$ até $p+5(\lceil n/5 \rceil - 1)$ **faça**

2 ORDENE ($A, j, j+4$)

3 ORDENE ($A, p+5\lfloor n/5 \rfloor, r$)

4 **para** $j \leftarrow 1$ até $\lceil n/5 \rceil - 1$ **faça**

5 $A[p-1+j] \leftrightarrow A[p+5j-3]$

6 $A[p-1+\lceil n/5 \rceil] \leftrightarrow A[\lfloor (p+5\lfloor n/5 \rfloor + r)/2 \rfloor]$

7 $k \leftarrow \text{SELECT-BFPRT}(A, p, p+\lceil n/5 \rceil - 1, \lfloor (\lceil n/5 \rceil + 1)/2 \rfloor)$

8 $A[k] \leftrightarrow A[r]$

9 **devolva** PARTICIONE (A, p, r)

Consumo de tempo do Particione-BFPRT

$P(n) :=$ consumo de tempo **máximo** do algoritmo
PARTICIONE-BFPRT quando $n = r - p + 1$

linha	consumo de todas as execuções da linha
1-3	$= \lceil n/5 \rceil \Theta(1) = \Theta(n)$
4-6	$= \lceil n/5 \rceil \Theta(1) = \Theta(n)$
7	$= T(\lceil n/5 \rceil)$
8	$= \Theta(1)$
9	$= \Theta(n)$

$$P(n) = \Theta(n) + T(\lceil n/5 \rceil)$$

Consumo de tempo do Select-BFPRT

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

Temos que

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

$$\begin{aligned} T(n) &\leq \Theta(1) + P(n) + T\left(\left\lceil \frac{7n}{10} \right\rceil + 3\right) \\ &\leq \Theta(1) + \Theta(n) + T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\left\lceil \frac{7n}{10} \right\rceil + 3\right) \\ &= \Theta(n) + T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\left\lceil \frac{7n}{10} \right\rceil + 3\right) \end{aligned}$$

para $n = 2, 3, \dots$

Consumo de tempo do Select-BFPRT

$T(n)$ pertence a mesma classe O que:

$$S(n) = 1 \text{ para } n < 30$$

$$S(n) \leq S\left(\left\lceil \frac{n}{5} \right\rceil\right) + S\left(\left\lceil \frac{7n}{10} \right\rceil + 3\right) + n \text{ para } n \geq 30$$

n	30	60	90	120	150	180	210	240	270	300
$S(n)$	32	144	280	362	514	640	802	940	1114	1261

Vamos verificar que $S(n) < 80n$ para $n = 1, 2, 3, 4, \dots$

Prova: Se $n = 1, \dots, 29$, então $S(n) = 1 < 80 < 80n$.

Se $n = 30, \dots, 99$, então

$$S(n) < S(120) = 362 < 80 \times 30 \leq 80n.$$

Recorrência

Se $n \geq 100$, então

$$\begin{aligned} S(n) &\leq s\left(\left\lceil \frac{n}{5} \right\rceil\right) + s\left(\left\lceil \frac{7n}{10} \right\rceil + 3\right) + n \\ &\stackrel{\text{hi}}{<} 80 \left\lceil \frac{n}{5} \right\rceil + 80 \left(\left\lceil \frac{7n}{10} \right\rceil + 3\right) + n \\ &\leq 80 \left(\frac{n}{5} + 1\right) + 80 \left(\frac{7n}{10} + 4\right) + n \\ &= 80 \frac{n}{5} + 80 + 80 \frac{7n}{10} + 320 + n \\ &= 16n + 56n + n + 400 \\ &= 73n + 400 \\ &< 80n \quad (\text{pois } n \geq 100). \end{aligned}$$

Logo, $T(n)$ é $O(n)$.

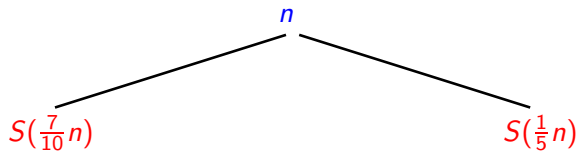
Como adivinhei classe O?

Árvore da recorrência:

$$S(n)$$

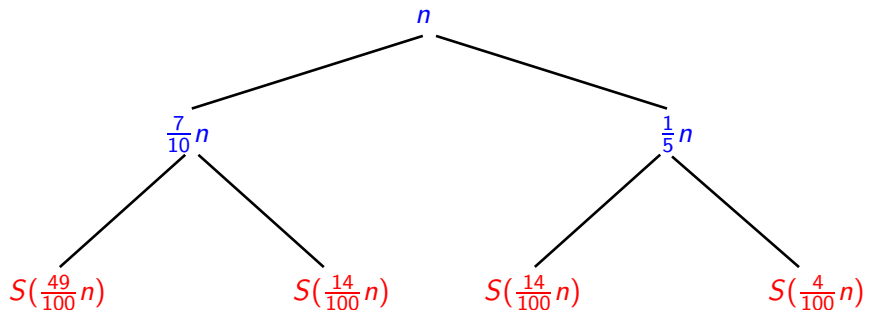
Como adivinhei classe O?

Árvore da recorrência:



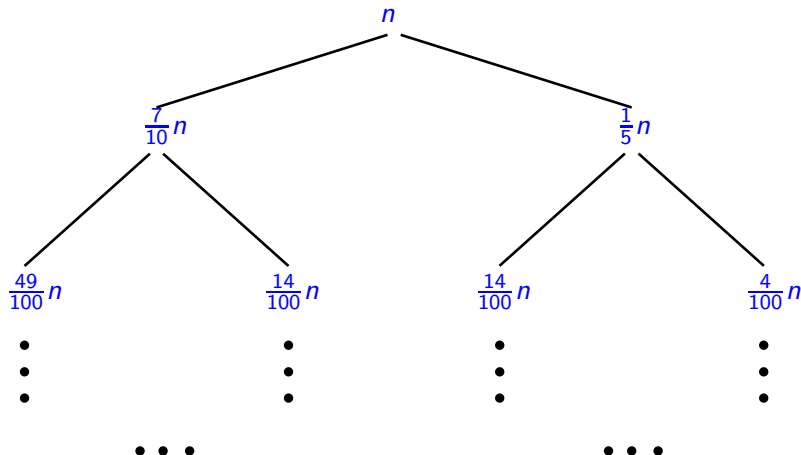
Como adivinhei classe O?

Árvore da recorrência:



Como adivinhei classe O?

Árvore da recorrência:



Contas

nível	0	1	2	...	$k-1$	k
soma	n	$\frac{9}{10}n$	$\frac{9^2}{10^2}n$	...	$\frac{9^{k-1}}{10^{k-1}}n$	$\frac{9^k}{10^k}n$

$$\frac{10^{k-1}}{9^{k-1}} < n \leq \frac{10^k}{9^k} \Rightarrow k = \lceil \log_{\frac{10}{9}} n \rceil$$

$$\begin{aligned} S(n) &= n + \frac{9}{10}n + \dots + \frac{9^{k-1}}{10^{k-1}}n + \frac{9^k}{10^k}n \\ &= \left(1 + \frac{9}{10} + \dots + \frac{9^k}{10^k}\right)n \\ &= 10\left(1 - \frac{9^{k+1}}{10^{k+1}}\right)n \\ &< 10n \end{aligned}$$

Conclusão

O consumo de tempo do **SELECT-BFPRT** é $O(n)$.