Tópicos de Análise de Algoritmos

Parte destes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Aulas 3 e 4

Complexidade computacional para algoritmos numéricos

Modelos de computação

Custo uniforme: toda instrução consome uma unidade de tempo e todo registrador usa uma unidade de memória.

Custo logarítmico: Toda instrução consome tempo logarítmico em relação ao valor dos operandos e cada registrador requer um número logarítmico de unidades de memória no valor armazenado.

Modelos de computação

Custo uniforme: toda instrução consome uma unidade de tempo e todo registrador usa uma unidade de memória.

Custo logarítmico: Toda instrução consome tempo logarítmico em relação ao valor dos operandos e cada registrador requer um número logarítmico de unidades de memória no valor armazenado.

Analisamos a maioria dos algoritmos que vemos no modelo de custo uniforme. Porém, para alguns, a análise deve ser feita no modelo de custo logarítmico.

Modelos de computação

Custo uniforme: toda instrução consome uma unidade de tempo e todo registrador usa uma unidade de memória.

Custo logarítmico: Toda instrução consome tempo logarítmico em relação ao valor dos operandos e cada registrador requer um número logarítmico de unidades de memória no valor armazenado.

Analisamos a maioria dos algoritmos que vemos no modelo de custo uniforme. Porém, para alguns, a análise deve ser feita no modelo de custo logarítmico.

FFT faz $\mathrm{O}(n \lg n)$ operações com números complexos (custo unitário).

Modelos de computação

Custo uniforme: toda instrução consome uma unidade de tempo e todo registrador usa uma unidade de memória.

Custo logarítmico: Toda instrução consome tempo logarítmico em relação ao valor dos operandos e cada registrador requer um número logarítmico de unidades de memória no valor armazenado.

Analisamos a maioria dos algoritmos que vemos no modelo de custo uniforme. Porém, para alguns, a análise deve ser feita no modelo de custo logarítmico.

FFT faz $\mathrm{O}(n \lg n)$ operações com números complexos (custo unitário).

Os tempos mostrados para os algoritmos de multiplicação de inteiros grandes são em função do número de operações com os dígitos (bits) dos números (custo logarítmico).

Algoritmo de Karatsuba e Ofman

Algoritmo recebe inteiros $X[1\,..\,n]$ e $Y[1\,..\,n]$ e devolve $X \cdot Y$.

KARATSUBA (X, Y, n)

```
 1   se n ≤ 3 devolva X · Y
 2   q ← ⌈n/2⌉
 3   A ← X[q + 1 .. n]        B ← X[1 .. q]
 4   C ← Y[q + 1 .. n]        D ← Y[1 .. q]
 5   E ← KARATSUBA(A, C, ⌊n/2⌋)
 6   F ← KARATSUBA(B, D, ⌈n/2⌉)
 7   G ← KARATSUBA(A + B, C + D, ⌈n/2⌉ + 1)
 8   H ← G − F − E
 9   R ← E × 10^{2⌈n/2⌉} + H × 10^{⌈n/2⌉} + F
10   devolva R
```

$T(n) =$ consumo de tempo do algoritmo para multiplicar dois inteiros com n algarismos.

Consumo de tempo

linha		todas as execuções da linha
1	$=$	$\Theta(1)$
2	$=$	$\Theta(1)$
3	$=$	$\Theta(n)$
4	$=$	$\Theta(n)$
5	$=$	$T(\lfloor n/2 \rfloor)$
6	$=$	$T(\lceil n/2 \rceil)$
7	$=$	$T(\lceil n/2 \rceil + 1) + \Theta(n)$
8	$=$	$\Theta(n)$
9	$=$	$\Theta(n)$
10	$=$	$\Theta(n)$
total	$=$	$T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n)$

Transformada rápida de Fourier

```
DFT (a, n)                                          ▷ n é uma potência de 2
 1   se n = 1
 2         então devolva a
 3   a^0 ← (a_0, a_2, ..., a_{n-2})
 4   a^1 ← (a_1, a_3, ..., a_{n-1})
 5   y^0 ← DFT (a^0, n/2)
 6   y^1 ← DFT (a^1, n/2)
 7   ω_n ← e^{2πi/n}
 8   ω ← 1
 9   para k ← 0 até n/2 − 1 faça
10         y[k] ← y^0[k] + ω y^1[k]
11         y[k + n/2] ← y^0[k] − ω y^1[k]
12         ω ← ω ω_n
13   devolva y
```

Recorrências

Como resolver $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + T(\lceil \frac{n}{2} \rceil + 1) + \Theta(n)$?

Recorrências

Como resolver $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + T(\lceil \frac{n}{2} \rceil + 1) + \Theta(n)$?

Inspire-se na prova do Teorema Mestre.

Considere a recorrência

$$\begin{array}{rcll} T(n) & = & 1 & \text{para } n \leq 5 \\ T(n) & = & 3T(\lceil \frac{n}{2} \rceil + 1) + n & \text{para } n \geq 6. \end{array}$$

Vamos provar que $T(n) = \mathrm{O}(n^{\lg 3})$.

Recorrências

Como resolver $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + T(\lceil \frac{n}{2} \rceil + 1) + \Theta(n)$?

Inspire-se na prova do Teorema Mestre.

Considere a recorrência

$$\begin{array}{rcll} T(n) & = & 1 & \text{para } n \leq 5 \\ T(n) & = & 3T(\lceil \frac{n}{2} \rceil + 1) + n & \text{para } n \geq 6. \end{array}$$

Vamos provar que $T(n) = \mathrm{O}(n^{\lg 3})$.

Defina

$$n_i = \begin{cases} n & \text{se } i = 0 \\ \lceil \frac{n_{i-1}}{2} \rceil + 1 & \text{se } i \geq 1 \end{cases}$$

e prove por indução que $n_i \leq \frac{n-4}{2^i} + 4$.

Recorrências

Como resolver $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + T(\lceil \frac{n}{2} \rceil + 1) + \Theta(n)$?

Inspire-se na prova do Teorema Mestre.

Considere a recorrência

$$\begin{array}{rcll} T(n) & = & 1 & \text{para } n \leq 5 \\ T(n) & = & 3T(\lceil \frac{n}{2} \rceil + 1) + n & \text{para } n \geq 6. \end{array}$$

Vamos provar que $T(n) = \mathrm{O}(n^{\lg 3})$.

Defina

$$n_i = \begin{cases} n & \text{se } i = 0 \\ \lceil \frac{n_{i-1}}{2} \rceil + 1 & \text{se } i \geq 1 \end{cases}$$

e prove por indução que $n_i \leq \frac{n-4}{2^i} + 4$.

Base: $i = 0$

$$n_0 = n = (n-4) + 4$$

Recorrências

Como resolver $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + T(\lceil \frac{n}{2} \rceil + 1) + \Theta(n)$?

Inspire-se na prova do Teorema Mestre.

Considere a recorrência

$$\begin{array}{rcll} T(n) & = & 1 & \text{para } n \leq 5 \\ T(n) & = & 3T(\lceil \frac{n}{2} \rceil + 1) + n & \text{para } n \geq 6. \end{array}$$

Vamos provar que $T(n) = \mathrm{O}(n^{\lg 3})$.

Defina

$$n_i = \begin{cases} n & \text{se } i = 0 \\ \lceil \frac{n_{i-1}}{2} \rceil + 1 & \text{se } i \geq 1 \end{cases}$$

e prove por indução que $n_i \leq \frac{n-4}{2^i} + 4$.

Base: $i = 0$

$$n_0 = n = (n-4) + 4$$

Passo: $i \geq 1$

$$n_i = \lceil \frac{n_{i-1}}{2} \rceil + 1 \leq \lceil \frac{1}{2}(\frac{n-4}{2^{i-1}} + 4) \rceil + 1 \leq \lceil \frac{n-4}{2^i} + 2 \rceil + 1 \leq \frac{n-4}{2^i} + 4.$$

Recorrências

Como resolver $T(n) = 3T(\lceil \frac{n}{2} \rceil + 1) + n$?

Desenrole a recorrência:

$$
\begin{aligned}
T(n) &= 3T(n_1) + n_0 \\
&= 3(3T(n_2) + n_1) + n_0 \;=\; 3^2 T(n_2) + 3n_1 + n_0 \\
&= 3^2(3T(n_3) + n_2) + 3n_1 + n_0 \;=\; 3^3 T(n_3) + 3^2 n_2 + 3n_1 + n_0 \\
&= \cdots \\
&= 3^i T(n_i) + 3^{i-1} n_{i-1} + \cdots + 3^2 n_2 + 3n_1 + n_0 \\
&= 3^i T(n_i) + \sum_{j=0}^{i-1} 3^j n_j .
\end{aligned}
$$

Recorrências

Como resolver $T(n) = 3T(\lceil \frac{n}{2} \rceil + 1) + n$?

Desenrole a recorrência:

$$\begin{aligned}
T(n) &= 3T(n_1) + n_0 \\
&= 3(3T(n_2) + n_1) + n_0 \;=\; 3^2 T(n_2) + 3n_1 + n_0 \\
&= 3^2(3T(n_3) + n_2) + 3n_1 + n_0 \;=\; 3^3 T(n_3) + 3^2 n_2 + 3n_1 + n_0 \\
&= \cdots \\
&= 3^i T(n_i) + 3^{i-1} n_{i-1} + \cdots + 3^2 n_2 + 3n_1 + n_0 \\
&= 3^i T(n_i) + \sum_{j=0}^{i-1} 3^j n_j.
\end{aligned}$$

Seja i menor possível tal que $n_i \leq 5$.

Note que neste caso $T(n_i) = 1$.

Recorrências

Como resolver $T(n) = 3T(\lceil \frac{n}{2} \rceil + 1) + n$?

Desenrole a recorrência, obtendo para i menor possível tal que $n_i \leq 5$:

$$T(n) = 3^i + \sum_{j=0}^{i-1} 3^j n_j.$$

Usando que $n_j \leq \frac{n-4}{2^j} + 4$ para todo j,

$$\begin{aligned} T(n) &\leq 3^i + \sum_{j=0}^{i-1} 3^j \frac{n-4}{2^j} + 4\sum_{j=0}^{i-1} 3^j \\ &\leq 3^i + \sum_{j=0}^{i-1} 3^j \frac{n-4}{2^j} + 4\frac{(3^i-1)}{2} \\ &\leq 3 \cdot 3^i - 2 + (n-4)\sum_{j=0}^{i-1} (\frac{3}{2})^j. \end{aligned}$$

Recorrências

Como resolver $T(n) = 3T(\lceil \frac{n}{2} \rceil + 1) + n$?

Desenrole a recorrência, obtendo para i menor possível tal que $n_i \leq 5$:

$$T(n) = 3^i + \sum_{j=0}^{i-1} 3^j n_j.$$

Usando que $n_j \leq \frac{n-4}{2^j} + 4$ para todo j, e que $i \leq \lg n$,

$$\begin{aligned}
T(n) &\leq 3 \cdot 3^i - 2 + (n-4) \sum_{j=0}^{i-1} (\frac{3}{2})^j \\
&\leq 3 \cdot 3^i - 2 + 2(n-4)((\frac{3}{2})^i - 1) \\
&\leq 3 \cdot n^{\lg 3} + 2(n-4)\frac{n^{\lg 3}}{n} - 2n + 6 \\
&\leq 5 \cdot n^{\lg 3} - 2n + 6.
\end{aligned}$$

Recorrências

Como resolver $T(n) = 3T(\lceil \frac{n}{2} \rceil + 1) + n$?

Desenrole a recorrência, obtendo para i menor possível tal que $n_i \leq 5$:

$$T(n) \;\; = \;\; 3^i + \sum_{j=0}^{i-1} 3^j n_j.$$

Usando que $n_i \leq \frac{n-4}{2^i} + 4$, conclua que

$$T(n) \;\leq\; 5n^{\lg 3} - 2n + 6.$$

Recorrências

Como resolver $T(n) = 3T(\lceil \frac{n}{2} \rceil + 1) + n$?

Desenrole a recorrência, obtendo para i menor possível tal que $n_i \leq 5$:

$$T(n) \quad = \quad 3^i + \sum_{j=0}^{i-1} 3^j n_j.$$

Usando que $n_i \leq \frac{n-4}{2^i} + 4$, conclua que

$$T(n) \ \leq \ 5n^{\lg 3} - 2n + 6.$$

Portanto

$$T(n) = \mathrm{O}(n^{\lg 3}).$$

Aula 5

Seleção do i-ésimo menor elemento

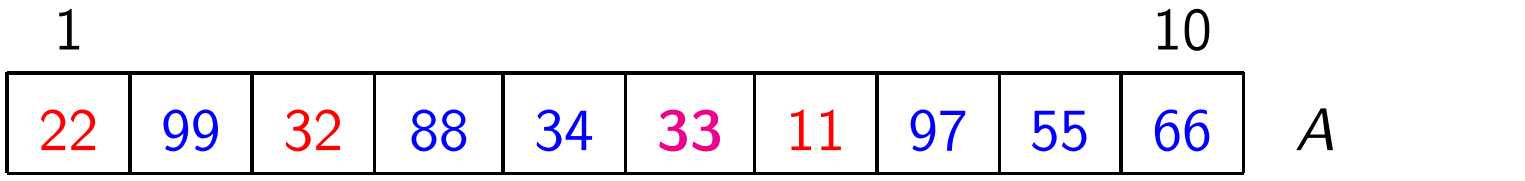
Cap 9 do CLRS.

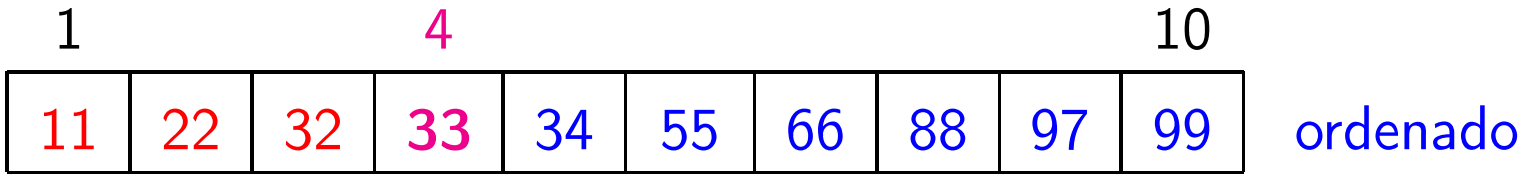
i-ésimo menor

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

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

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

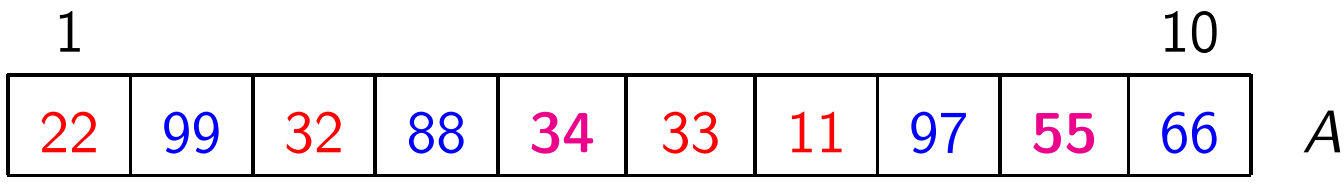


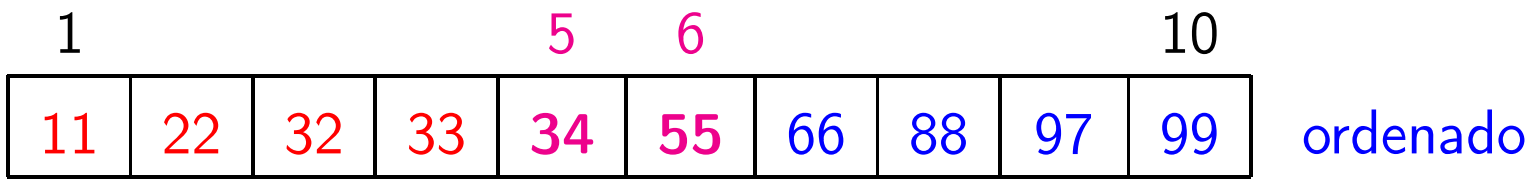


Mediana

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:





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 ← A[1]
2    para k ← 2 até n faça
3         se A[k] < menor
4              então menor ← 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?

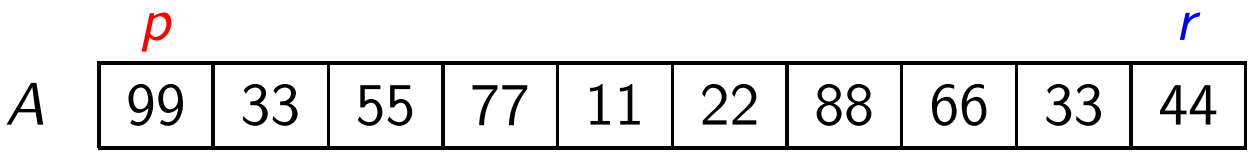
Sim!

Usaremos o PARTICIONE do QUICKSORT!

Particione

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

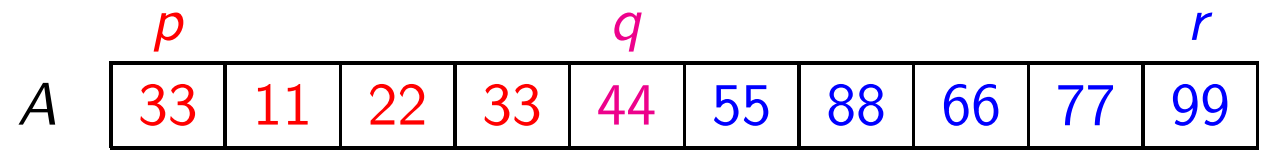
```
PARTICIONE (A, p, r)
1    x ← A[r]          ▷ x é o "pivô"
2    i ← p−1
3    para j ← p até r − 1 faça
4         se A[j] ≤ x
5              então i ← i + 1
6                    A[i] ↔ A[j]
7    A[i+1] ↔ A[r]
8    devolva i + 1
```



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
```



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
```

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

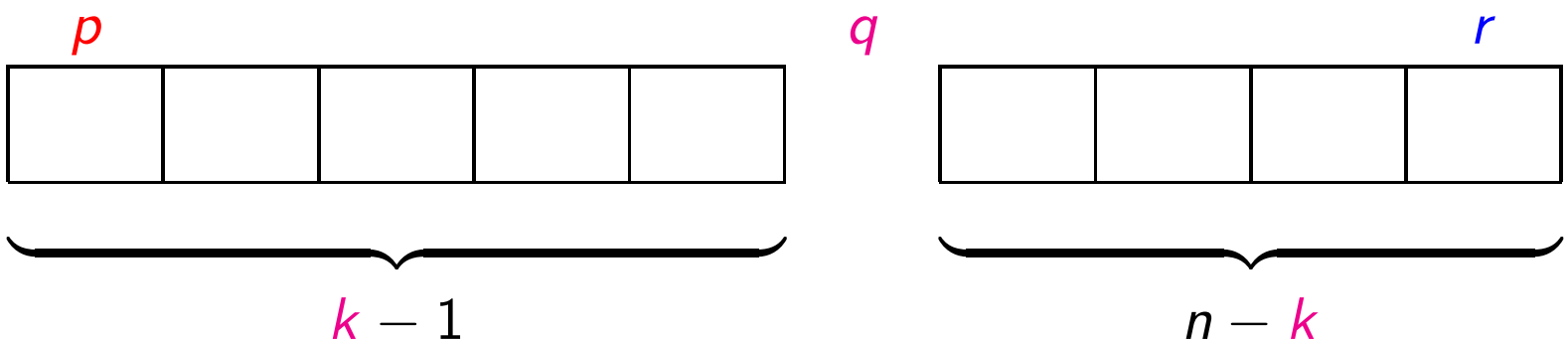
Algoritmo SELECT

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 ← PARTICIONE (p, r)
4    k ← 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 ← PARTICIONE (A, p, r)
4    k ← 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.

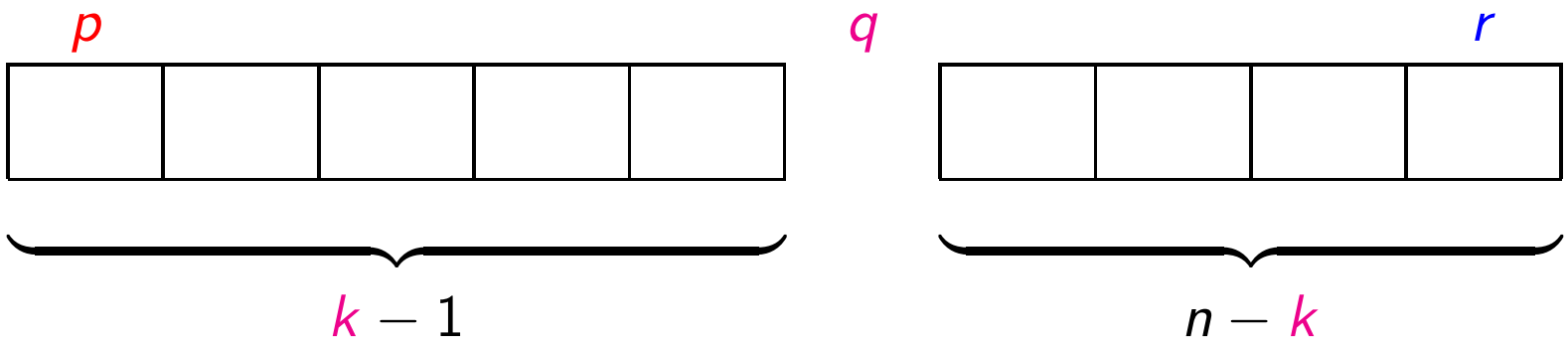
Se o pivô do PARTICIONE for a mediana do vetor,
qual seria o consumo de tempo do SELECT?

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      ▷ p e não A[p]
3    q ← PARTICIONE-BFPRT (A, p, r)
4    k ← q − p + 1
5    se k = i
6          então devolva q      ▷ 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 $p \le q \le r$,
tal que $A[p\,..\,q{-}1] \le A[q] < A[q{+}1\,..\,r]$ **e**

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

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

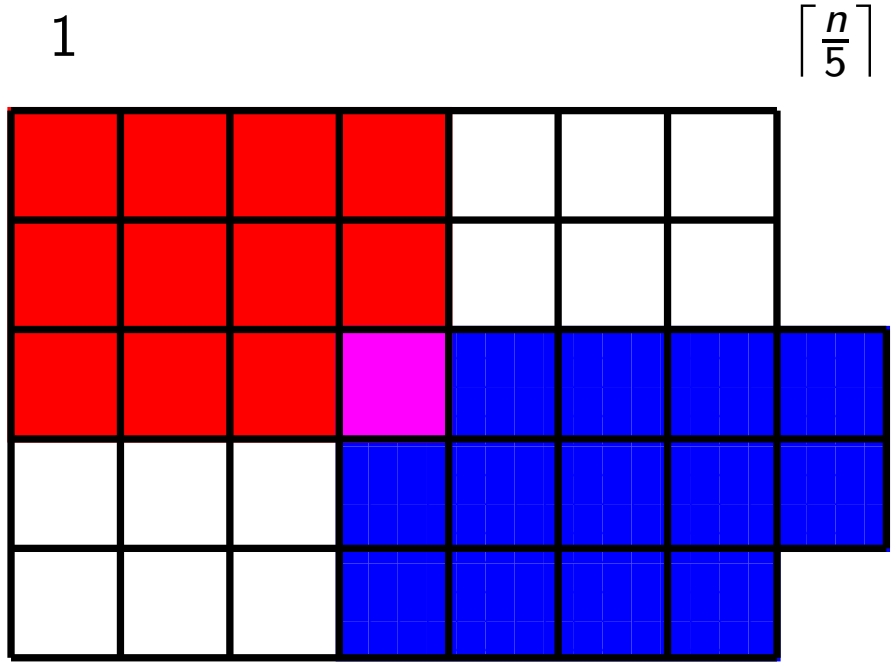
$T(n) :=$ consumo de tempo máximo do algoritmo SELECT-BFPRT quando $n = r - p + 1$

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

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

$$\leq \Theta(1) + P(n) + T(\lceil \tfrac{7n}{10} \rceil + 3)$$

Particione-BFPRT



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

Particione-BFPRT

$n := r - p + 1$

PARTICIONE-BFPRT (A, p, r)

1 **para** $j \leftarrow p, p{+}5, p{+}5 \cdot 2, \ldots$ **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 $B[\lceil n/5 \rceil] \leftarrow A[\lfloor (p{+}5\lfloor n/5 \rfloor {+} r)/2 \rfloor]$

7 $k \leftarrow$ 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** $j \leftarrow p, p{+}5, p{+}5 \cdot 2, \ldots$ **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 $B[\lceil n/5 \rceil] \leftarrow A[\lfloor (p{+}5\lfloor n/5 \rfloor {+} r)/2 \rfloor]$

7 $k \leftarrow$ 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** $j \leftarrow p, p{+}5, p{+}5 \cdot 2, \ldots$ **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 $k \leftarrow$ 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

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

para $n = 2, 3, \ldots$

Consumo de tempo do Select-BFPRT

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

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

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

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 $S(n) < 80\,n$ para $n = 1, 2, 3, 4, \ldots$

Prova: Se $n = 1, \ldots, 29$, então $S(n) = 1 < 80 < 80n$.
Se $n = 30, \ldots, 99$, então

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

Recorrência

Se $n \geq 100$, então

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

Logo, $T(n)$ é $\mathrm{O}(n)$.

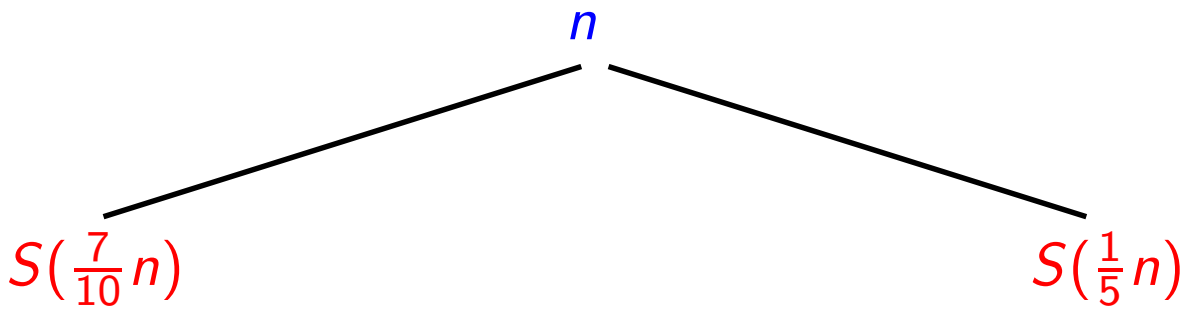
Como adivinhei classe O?

Árvore da recorrência:

$S(n)$

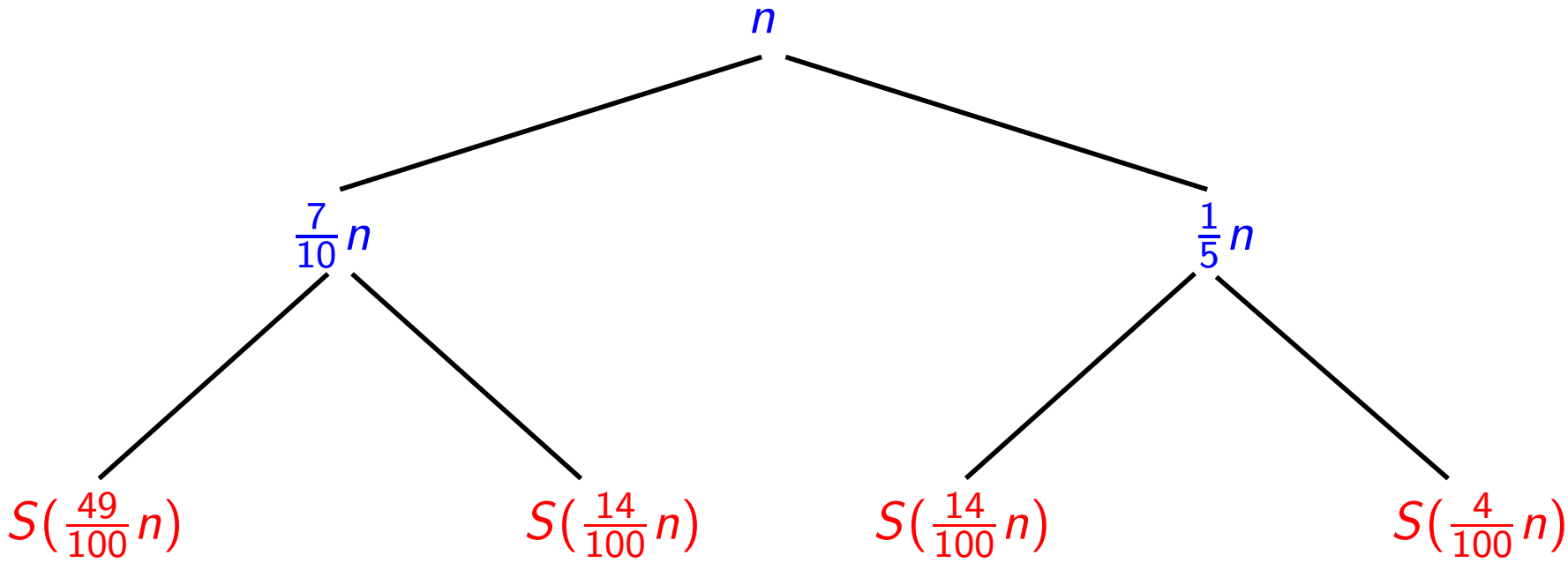
Como adivinhei classe O?

Árvore da recorrência:



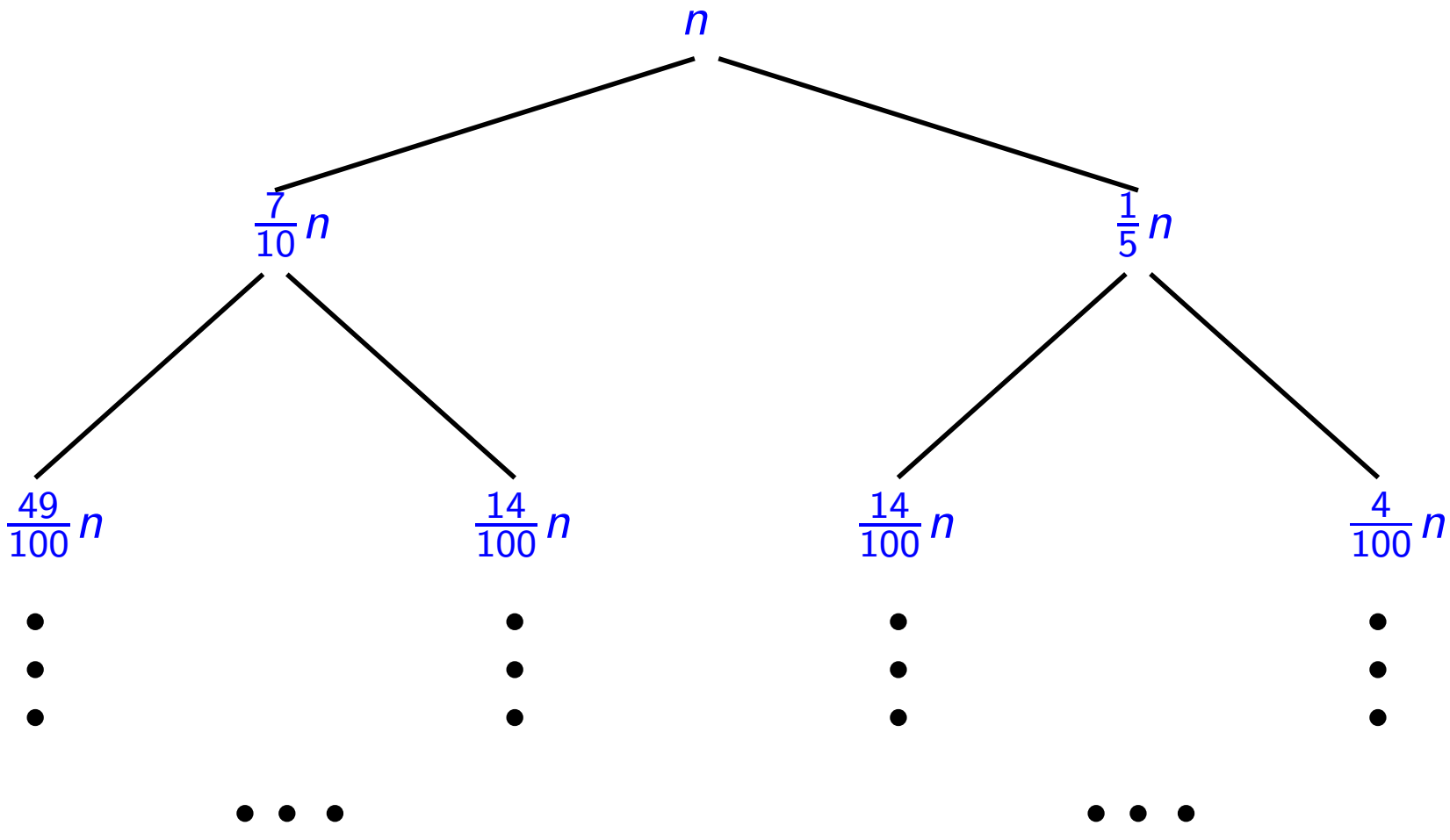
Como adivinhei classe O?

Árvore da recorrência:



Como adivinhei classe O?

Árvore da recorrência:



Contas

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

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

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

Conclusão

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