MC458 — Projeto e Análise de Algoritmos I

C.C. de Souza   C.N. da Silva   O. Lee

Antes de mais nada...

- Uma versão anterior deste conjunto de slides foi preparada por Cid Carvalho de Souza e Cândida Nunes da Silva para uma instância anterior desta disciplina.
- O que vocês tem em mãos é uma versão modificada preparada para atender a meus gostos.
- Nunca é demais enfatizar que o material é apenas um guia e não deve ser usado como única fonte de estudo. Para isso consultem a bibliografia (em especial o CLR ou CLRS).

Orlando Lee

Agradecimentos (Cid e Cândida)

- Várias pessoas contribuíram direta ou indiretamente com a preparação deste material.
- Algumas destas pessoas cederam gentilmente seus arquivos digitais enquanto outras cederam gentilmente o seu tempo fazendo correções e dando sugestões.
- Uma lista destes "colaboradores" (em ordem alfabética) é dada abaixo:
  - Célia Picinin de Mello
  - José Coelho de Pina
  - Orlando Lee
  - Paulo Feofiloff
  - Pedro Rezende
  - Ricardo Dahab
  - Zanoni Dias

Estatísticas de Ordem

Estatísticas de Ordem (Problema da Seleção)

Estatísticas de Ordem (Problema da Seleção)

- Estamos interessados em resolver o

Problema da Seleção:

Dado um vetor $A[1\,..\,n]$ de reais e um inteiro i, determinar o i-ésimo menor elemento de A.

- Estamos interessados em resolver o

Problema da Seleção:

Dado um vetor $A[1..n]$ de reais e um inteiro i, determinar o i-ésimo menor elemento de A.

- Casos particulares importantes:

Mínimo : $i = 1$

Máximo : $i = n$

Mediana : $i = \lfloor \frac{n+1}{2} \rfloor$ (mediana inferior)

Mediana : $i = \lceil \frac{n+1}{2} \rceil$ (mediana superior)

Mínimo

Recebe um vetor $A[1\,.\,.\,n]$ de reais e devolve o mínimo do vetor.

Mínimo(A, n)
1   min $\leftarrow A[1]$
2   **para** $j \leftarrow 2$ **até** n **faça**
3   **se** min $> A[j]$
4   **então** min $\leftarrow A[j]$
5   **devolva** min

Mínimo

Recebe um vetor $A[1\,..\,n]$ de reais e devolve o mínimo do vetor.

$\textsc{Mínimo}(A, n)$

    min $\leftarrow A[1]$
    para $j \leftarrow 2$ **até** n **faça**
        se min $> A[j]$
            então min $\leftarrow A[j]$
    devolva min

Número de comparações: $n - 1 = \Theta(n)$

Mínimo

Recebe um vetor $A[1\,..\,n]$ de reais e devolve o mínimo do vetor.

```
Mínimo(A, n)
    min ← A[1]
    para j ← 2 até n faça
        se min > A[j]
            então min ← A[j]
    devolva  min
```

Número de comparações: $n - 1 = \Theta(n)$

Aula passada: não é possível fazer com menos comparações.

Recebe um vetor $A[1\,..\,n]$ de reais e devolve o mínimo e o máximo do vetor.

```
MinMax(A, n)
    min ← max ← A[1]
    para j ← 2 até n faça
        se A[j] < min
            então min ← A[j]
        se A[j] > max
            então max ← A[j]
    devolva (min, max)
```

Mínimo e máximo

Recebe um vetor $A[1\,.\,.\,n]$ de reais e devolve o mínimo e o máximo do vetor.

MinMax(A, n)
```
min ← max ← A[1]
para j ← 2 até n faça
    se A[j] < min
        então min ← A[j]
    se A[j] > max
        então max ← A[j]
devolva (min, max)
```

Número de comparações: $2(n-1) = 2n-2 = \Theta(n)$

Mínimo e máximo

Recebe um vetor $A[1\,..\,n]$ de reais e devolve o mínimo e o máximo do vetor.

```
MinMax(A, n)
    min ← max ← A[1]
    para j ← 2 até n faça
        se A[j] < min
            então min ← A[j]
        se A[j] > max
            então max ← A[j]
    devolva (min, max)
```

Número de comparações: $2(n-1) = 2n-2 = \Theta(n)$

É possível fazer melhor!

Mínimo e máximo

Mínimo e máximo

- Processe os elementos em pares. Para cada par, compare o menor com o mínimo atual e o maior com o máximo atual. Isto resulta em 3 comparações para cada 2 elementos.

Mínimo e máximo

- Processe os elementos em pares. Para cada par, compare o menor com o mínimo atual e o maior com o máximo atual. Isto resulta em 3 comparações para cada 2 elementos.
- Se n for ímpar, inicialize o mínimo e o máximo como sendo o primeiro elemento.

Mínimo e máximo

- Processe os elementos em pares. Para cada par, compare o menor com o mínimo atual e o maior com o máximo atual. Isto resulta em 3 comparações para cada 2 elementos.
- Se n for ímpar, inicialize o mínimo e o máximo como sendo o primeiro elemento.
- Se n for par, inicialize o mínimo e o máximo comparando os dois primeiros elementos.

Mínimo e máximo

- Processe os elementos em pares. Para cada par, compare o menor com o mínimo atual e o maior com o máximo atual. Isto resulta em 3 comparações para cada 2 elementos.
- Se n for ímpar, inicialize o mínimo e o máximo como sendo o primeiro elemento.
- Se n for par, inicialize o mínimo e o máximo comparando os dois primeiros elementos.

Número de comparações:

$$\begin{array}{ll} 3\lfloor n/2 \rfloor & \text{se } n \text{ for ímpar} \\ 3\lfloor n/2 \rfloor - 2 & \text{se } n \text{ for par} \end{array}$$

Mínimo e máximo

- Processe os elementos em pares. Para cada par, compare o menor com o mínimo atual e o maior com o máximo atual. Isto resulta em 3 comparações para cada 2 elementos.
- Se n for ímpar, inicialize o mínimo e o máximo como sendo o primeiro elemento.
- Se n for par, inicialize o mínimo e o máximo comparando os dois primeiros elementos.

Número de comparações:

$$\begin{array}{ll} 3\lfloor n/2 \rfloor & \text{se } n \text{ for ímpar} \\ 3\lfloor n/2 \rfloor - 2 & \text{se } n \text{ for par} \end{array}$$

Pode-se mostrar que isto é o melhor possível.
(Exercício 9.1-2 $\star$ do CLRS)

Problema da Seleção – primeira solução

Recebe um vetor $A[1\,..\,n]$ de reais e um inteiro i ($1 \leq i \leq n$).
Devolve o i-ésimo menor elemento de $A[1\,..\,n]$.

Select-Ord(A, n, i)
1 Ordene(A, n)
2 **devolva** $A[i]$

Ordene pode ser *MergeSort* ou *HeapSort*.

Problema da Seleção – primeira solução

Recebe um vetor $A[1\,..\,n]$ de reais e um inteiro i $(1 \leq i \leq n)$.
Devolve o i-ésimo menor elemento de $A[1\,..\,n]$.

$\text{S}\textsc{elect}\text{-O}\textsc{rd}(A, n, i)$
1 $\text{O}\textsc{rdene}(A, n)$
2 **devolva** $A[i]$

$\text{O}\textsc{rdene}$ pode ser *MergeSort* ou *HeapSort*.

A complexidade de tempo de $\text{S}\textsc{elect}\text{-O}\textsc{rd}$ é $O(n \lg n)$.

Recebe um vetor $A[1\,..\,n]$ de reais e um inteiro i $(1 \leq i \leq n)$.
Devolve o i-ésimo menor elemento de $A[1\,..\,n]$.

Select-Ord(A, n, i)
1 Ordene(A, n)
2 **devolva** $A[i]$

Ordene pode ser *MergeSort* ou *HeapSort*.

A complexidade de tempo de Select-Ord é $O(n \lg n)$.

Será que não dá para fazer melhor que isso?

Afinal, consigo achar o mínimo e/ou máximo em tempo $O(n)$.

Problema: rearranjar um dado vetor $A[p \mathinner{..} r]$ de reais e devolver um índice q, $p \leq q \leq r$, tais que

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

Entrada:

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

Problema: rearranjar um dado vetor $A[p \mathrel{.\,.} r]$ de reais e devolver um índice q, $p \leq q \leq r$, tais que

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

Entrada:

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

Saída:

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

Relembrando – Particione

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

$\textsc{Particione}(A, p, r)$

$x \leftarrow A[r]$ $\quad\triangleright$ x é o "pivô"
$i \leftarrow p{-}1$
para $j \leftarrow p$ **até** $r - 1$ **faça**
    se $A[j] \le x$
        então $i \leftarrow i + 1$
            $A[i] \leftrightarrow A[j]$
$A[i{+}1] \leftrightarrow A[r]$
devolva $i + 1$

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[1\,..\,n]$.

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[1\,..\,n]$.

- PARTICIONE rearranja o vetor A e devolve um índice k tal que $A[1\,..\,k-1] \leq A[k] < A[k+1\,..\,n]$.

	1				k					n
A	33	11	22	33	44	55	99	66	77	88

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[1\,.\,.\,n]$.

- PARTICIONE rearranja o vetor A e devolve um índice k tal que $A[1\,.\,.\,k-1] \leq A[k] < A[k+1\,.\,.\,n]$.

	1				k					n
A	33	11	22	33	44	55	99	66	77	88

- Se $i = k$, então o pivô $A[k]$ é o i-ésimo menor! (Yeesss!)

Suponha que queremos achar o i-ésimo menor de $A[1\,..\,n]$.

- PARTICIONE rearranja o vetor A e devolve um índice k tal que $A[1\,..\,k-1] \leq A[k] < A[k+1\,..\,n]$.

	1				k					n
A	33	11	22	33	44	55	99	66	77	88

- Se $i = k$, então o pivô $A[k]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[1\,..\,k-1]$:

Suponha que queremos achar o i-ésimo menor de $A[1 \mathinner{.\,.} n]$.

- PARTICIONE rearranja o vetor A e devolve um índice k tal que $A[1 \mathinner{.\,.} k-1] \leq A[k] < A[k+1 \mathinner{.\,.} n]$.

	1				k					n
A	33	11	22	33	44	55	99	66	77	88

- Se $i = k$, então o pivô $A[k]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[1 \mathinner{.\,.} k-1]$: procuramos o $\underline{i\text{-ésimo menor}}$ em $A[1 \mathinner{.\,.} k-1]$.

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[1\,..\,n]$.

- PARTICIONE rearranja o vetor A e devolve um índice k tal que $A[1\,..\,k-1] \le A[k] < A[k+1\,..\,n]$.

	1				k					n
A	33	11	22	33	44	55	99	66	77	88

- Se $i = k$, então o pivô $A[k]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[1\,..\,k-1]$: procuramos o <u>i-ésimo menor</u> em $A[1\,..\,k-1]$.
- Se $i > k$, então o i-ésimo menor está em $A[k+1\,..\,n]$:

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[1\,..\,n]$.

- PARTICIONE rearranja o vetor A e devolve um índice k tal que $A[1\,..\,k-1] \leq A[k] < A[k+1\,..\,n]$.

	1				k					n
A	33	11	22	33	44	55	99	66	77	88

- Se $i = k$, então o pivô $A[k]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[1\,..\,k-1]$: procuramos o i-ésimo menor em $A[1\,..\,k-1]$.
- Se $i > k$, então o i-ésimo menor está em $A[k+1\,..\,n]$: procuramos o $(i-k)$-ésimo menor em $A[k+1\,..\,n]$.

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p \,..\, r]$.

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p\,..\,r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,r]$.

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

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p\,.\,.\,r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p\,.\,.\,q-1] \leq A[q] < A[q+1\,.\,.\,r]$.

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

- Seja $k = q - p + 1$. Assim, $A[q]$ está na k-ésima posição de A.

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p \,..\, r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p \,..\, q-1] \leq A[q] < A[q+1 \,..\, r]$.

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

- Seja $k = q - p + 1$. Assim, $A[q]$ está na k-ésima posição de A.
- Se $i = k$, então o pivô $A[q]$ é o i-ésimo menor! (Yeesss!)

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p\,..\,r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,r]$.

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

- Seja $k = q - p + 1$. Assim, $A[q]$ está na k-ésima posição de A.
- Se $i = k$, então o pivô $A[q]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[p\,..\,q-1]$:

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p\,..\,r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,r]$.

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

- Seja $k = q - p + 1$. Assim, $A[q]$ está na k-ésima posição de A.
- Se $i = k$, então o pivô $A[q]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[p\,..\,q-1]$: procuramos o i-ésimo menor em $A[p\,..\,q-1]$.

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p\,..\,r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,r]$.

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

- Seja $k = q - p + 1$. Assim, $A[q]$ está na k-ésima posição de A.
- Se $i = k$, então o pivô $A[q]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[p\,..\,q-1]$: procuramos o <u>i-ésimo menor</u> em $A[p\,..\,q-1]$.
- Se $i > k$, então o i-ésimo menor está em $A[q+1\,..\,r]$:

Problema da Seleção – segunda solução

Suponha que queremos achar o i-ésimo menor de $A[p\,.\,.\,r]$.

- PARTICIONE rearranja o vetor A e devolve um índice q tal que $A[p\,.\,.\,q-1] \leq A[q] < A[q+1\,.\,.\,r]$.

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

- Seja $k = q - p + 1$. Assim, $A[q]$ está na k-ésima posição de A.
- Se $i = k$, então o pivô $A[q]$ é o i-ésimo menor! (Yeesss!)
- Se $i < k$, então o i-ésimo menor está em $A[p\,.\,.\,q-1]$: procuramos o $\underline{i\text{-ésimo menor}}$ em $A[p\,.\,.\,q-1]$.
- Se $i > k$, então o i-ésimo menor está em $A[q+1\,.\,.\,r]$: procuramos o $\underline{(i-k)\text{-ésimo menor}}$ em $A[q+1\,.\,.\,r]$.

Problema da Seleção – segunda solução

Recebe $A[p \mathinner{..} r]$ e i tal que $1 \leq i \leq r-p+1$
e devolve o i-ésimo menor elemento de $A[p \mathinner{..} r]$.

$\textsc{Select-NL}(A, p, r, i)$

    se $p = r$
        então devolva $A[p]$
    $q \leftarrow \textsc{Particione}(A, p, r)$
    $k \leftarrow q - p + 1$
    se $i = k$ $\triangleright$ pivô é o i-ésimo menor!
        então devolva $A[q]$
        senão se $i < k$
            então devolva $\textsc{Select-NL}(A, p, q-1, i)$
            senão devolva $\textsc{Select-NL}(A, q+1, r, i-k)$

Segunda solução – complexidade

SELECT-NL(A, p, r, i)	Tempo
se $p = r$	?
então devolva $A[p]$	?
$q \leftarrow$ PARTICIONE(A, p, r)	?
$k \leftarrow q - p + 1$	?
se $i = k$	?
então devolva $A[q]$	?
senão se $i < k$	?
então devolva SELECT-NL($A, p, q - 1, i$)	?
senão devolva SELECT-NL($A, q + 1, r, i - k$)	?

$T(n)$ = complexidade de tempo no pior caso quando $n = r - p + 1$

Segunda solução – complexidade

SELECT-NL(A, p, r, i)	Tempo
se $p = r$	$\Theta(1)$
então devolva $A[p]$	$O(1)$
$q \leftarrow$ PARTICIONE(A, p, r)	$\Theta(n)$
$k \leftarrow q - p + 1$	$\Theta(1)$
se $i = k$	$\Theta(1)$
então devolva $A[q]$	$O(1)$
senão se $i < k$	$O(1)$
então devolva SELECT-NL$(A, p, q-1, i)$	$T(k-1)$
senão devolva SELECT-NL$(A, q+1, r, i-k)$	$T(n-k)$

$$T(n) = \max_{1 \leq k \leq n} \{T(k-1), T(n-k)\} + \Theta(n).$$

Segunda solução – complexidade

SELECT-NL(A, p, r, i)	Tempo
se $p = r$	$\Theta(1)$
então devolva $A[p]$	$O(1)$
$q \leftarrow$ PARTICIONE(A, p, r)	$\Theta(n)$
$k \leftarrow q - p + 1$	$\Theta(1)$
se $i = k$	$\Theta(1)$
então devolva $A[q]$	$O(1)$
senão se $i < k$	$O(1)$
então devolva SELECT-NL$(A, p, q-1, i)$	$T(k-1)$
senão devolva SELECT-NL$(A, q+1, r, i-k)$	$T(n-k)$

$$T(n) = \max_{1 \leq k \leq n} \{T(k-1), T(n-k)\} + \Theta(n).$$

Exercício. Mostre que $T(n) \in \Theta(n^2)$.

Segunda solução – complexidade

Segunda solução – complexidade

- A complexidade de Select-NL no pior caso é $\Theta(n^2)$.

Segunda solução – complexidade

- A complexidade de SELECT-NL no pior caso é $\Theta(n^2)$.
- Então é melhor usar SELECT-ORD?

Segunda solução – complexidade

- A complexidade de SELECT-NL no pior caso é $\Theta(n^2)$.
- Então é melhor usar SELECT-ORD?
- Não, SELECT-NL é muito eficiente na prática.

Segunda solução – complexidade

- A complexidade de SELECT-NL no pior caso é $\Theta(n^2)$.
- Então é melhor usar SELECT-ORD?
- Não, SELECT-NL é muito eficiente na prática.
- Vamos mostrar que no caso médio SELECT-NL tem complexidade $O(n)$.

SELECT aleatorizado

O pior caso do SELECT-NL ocorre devido a uma escolha infeliz do pivô. Um modo de evitar isso é usar aleatoriedade (como no QUICKSORT-ALEATÓRIO).

PARTICIONE-ALEATÓRIO(A, p, r)

1 $j \leftarrow$ RANDOM(p, r)

2 $A[j] \leftrightarrow A[r]$

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

Algoritmo SELECT-ALEAT

Recebe um vetor $A[p\,..\,r]$ e um inteiro i tal que $1 \leq i \leq r-p+1$.
Devolve o i-ésimo menor elemento de $A[p\,..\,r]$.

```
SELECT-ALEAT(A, p, r, i)
    se p = r
        então devolva A[p]
    q ← PARTICIONE-ALEATÓRIO(A, p, r)
    k ← q − p + 1
    se i = k    ▷ pivô é o i-ésimo menor
        então devolva A[q]
        senão se i < k
            então devolva SELECT-ALEAT(A, p, q − 1, i)
            senão devolva SELECT-ALEAT(A, q + 1, r, i − k)
```

Análise do caso médio

Recorrência para o caso médio de SELECT-ALEAT.

$T(n)$ = complexidade de tempo médio de SELECT-ALEAT.

$$T(0) = \Theta(1)$$
$$T(1) = \Theta(1)$$
$$T(n) \leq \frac{1}{n}\sum_{k=1}^{n} T(\max\{k-1, n-k\}) + \Theta(n).$$

$T(n)$ é $\Theta(???)$.

$$\begin{aligned} T(n) &\leq \frac{1}{n}\sum_{k=1}^{n} T(\max\{k-1, n-k\}) + an \\ &\leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} T(k) + an \end{aligned}$$

pois

$$\max\{k-1, n-k\} = \begin{cases} k-1 & \text{se } k > \lceil n/2 \rceil, \\ n-k & \text{se } k \leq \lceil n/2 \rceil. \end{cases}$$

Se n é par, cada termo de $T(\lceil n/2 \rceil)$ a $T(n-1)$ aparece exatamente duas vezes na somatória.

Se n é ímpar, esses termos aparecem duas vezes e $T(\lfloor n/2 \rfloor)$ aparece uma vez.

Demonstração: $T(n) \le cn$

$$
\begin{aligned}
T(n) &\le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} T(k) + an \\
&\overset{\text{hi}}{\le} \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an \\
&= \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor -1} k \right) + an \\
&= \frac{2c}{n} \left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2 \rfloor - 1)\lfloor n/2 \rfloor}{2} \right) + an
\end{aligned}
$$

Demonstração: $T(n) \leq cn$

$$
\begin{aligned}
T(n) &= \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2\rfloor - 1)\lfloor n/2\rfloor}{2}\right) + an \\
&\leq \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(n/2-2)(n/2-1)}{2}\right) + an \\
&= \frac{c}{n}\left(\frac{3n^2}{4} + \frac{n}{2} - 2\right) + an \\
&\leq \frac{3cn}{4} + \frac{c}{2} + an \\
&= cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right) \leq cn.
\end{aligned}
$$

Isto funciona se $c > 4a$ e $n \geq 2c/(c-4a)$.
Logo, $T(n) = O(n)$.

Conclusão

A complexidade de tempo de SELECT-ALEAT no caso médio é $O(n)$.

Conclusão

A complexidade de tempo de SELECT-ALEAT no caso médio é $O(n)$.

Na verdade,

A complexidade de tempo de SELECT-ALEAT no caso médio é $\Theta(n)$.

Conclusão

A complexidade de tempo de SELECT-ALEAT no caso médio é $O(n)$.

Na verdade,

A complexidade de tempo de SELECT-ALEAT no caso médio é $\Theta(n)$.

Próxima aula:
Algoritmo que resolve o Problema da Seleção em tempo linear (no pior caso).

Mediana de dois vetores ordenados

Exercício (CLRS 9.3-8). Sejam $X[1\,..\,n]$ e $Y[1\,..\,n]$ vetores **ordenados**. Projete um algoritmo de complexidade $O(\lg n)$ para encontrar a mediana (inferior) dos $2n$ elementos de X e Y.

Mediana de dois vetores ordenados

Exercício (CLRS 9.3-8). Sejam $X[1 \,..\, n]$ e $Y[1 \,..\, n]$ vetores **ordenados**. Projete um algoritmo de complexidade $O(\lg n)$ para encontrar a mediana (inferior) dos $2n$ elementos de X e Y.

Sugestão: para obter a complexidade exigida, o algoritmo deveria ser capaz de descartar frações grandes de cada vetor...

Problema da Seleção – terceira solução

Problema da Seleção:

Dado um vetor $A[1\,..\,n]$ de reais e um inteiro i, determinar o i-ésimo menor elemento de A.

Descreveremos um algoritmo linear para o Problema da Seleção.

BFPRT $=$ Blum, Floyd, Pratt, Rivest e Tarjan

Problema da Seleção – terceira solução

Problema da Seleção:

Dado um vetor $A[1..n]$ de reais e um inteiro i, determinar o i-ésimo menor elemento de A.

Descreveremos um algoritmo linear para o Problema da Seleção.

BFPRT $=$ Blum, Floyd, Pratt, Rivest e Tarjan

Para simplificar a exposição, vamos supor que os elementos em A são distintos.

Problema da Seleção – terceira solução

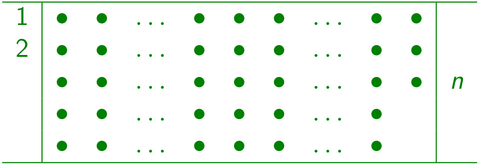
Problema da Seleção – terceira solução

1. Divida os n elementos em $\left\lfloor \frac{n}{5} \right\rfloor$ subconjuntos de 5 elementos e um subconjunto de $n \bmod 5$ elementos.

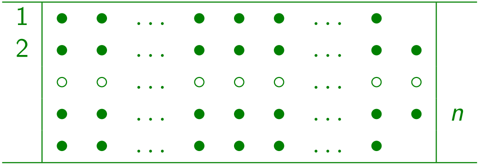
$$
\begin{array}{c|cccccccc|c}
\hline
1 & \bullet & \bullet & \dots & \bullet & \bullet & \bullet & \dots & \bullet & \bullet & \\
2 & \bullet & \bullet & \dots & \bullet & \bullet & \bullet & \dots & \bullet & \bullet & \\
 & \bullet & \bullet & \dots & \bullet & \bullet & \bullet & \dots & \bullet & \bullet & n \\
 & \bullet & \bullet & \dots & \bullet & \bullet & \bullet & \dots & \bullet & & \\
 & \bullet & \bullet & \dots & \bullet & \bullet & \bullet & \dots & \bullet & & \\
\hline
\end{array}
$$

Problema da Seleção – terceira solução

1. Divida os n elementos em $\left\lfloor \frac{n}{5} \right\rfloor$ subconjuntos de 5 elementos e um subconjunto de $n \bmod 5$ elementos.



2. Encontre a mediana de cada um dos $\left\lceil \frac{n}{5} \right\rceil$ subconjuntos.

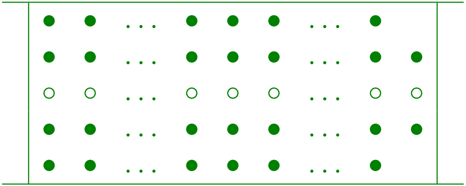


Na figura acima, cada subconjunto está em ordem crescente, de cima para baixo.

Problema da Seleção – terceira solução

Problema da Seleção – terceira solução

3. Determine, recursivamente, a mediana x das medianas dos subconjuntos de no máximo 5 elementos.



A figura acima é a mesma que a anterior, com as colunas ordenadas pela mediana de cada grupo. A ordem dos elementos em cada coluna permanece inalterada.
Por simplicidade de exposição, supomos que a última coluna permanece no mesmo lugar.
A figura serve apenas como auxílio visual: o algoritmo **não** ordena as medianas!

Problema da Seleção - terceira solução

Problema da Seleção - terceira solução

4. Usando x como pivô, particione o conjunto original A criando dois subconjuntos $A_<$ e $A_>$, onde

Problema da Seleção - terceira solução

4. Usando x como pivô, particione o conjunto original A criando dois subconjuntos $A_<$ e $A_>$, onde
   - $A_<$ contém os elementos $< x$ e

Problema da Seleção - terceira solução

4. Usando x como pivô, particione o conjunto original A criando dois subconjuntos $A_<$ e $A_>$, onde
   - $A_<$ contém os elementos $< x$ e
   - $A_>$ contém os elementos $> x$.

   Se a posição final de x após o particionamento é k, então $|A_<| = k - 1$ e $|A_>| = n - k$.

Problema da Seleção - terceira solução

Problema da Seleção - terceira solução

5. Finalmente, para encontrar o i-ésimo menor elemento do conjunto, compare i com a posição k de x após o particionamento:

Problema da Seleção - terceira solução

5. Finalmente, para encontrar o i-ésimo menor elemento do conjunto, compare i com a posição k de x após o particionamento:
   - Se $i = k$, x é o elemento procurado;

Problema da Seleção - terceira solução

5. Finalmente, para encontrar o i-ésimo menor elemento do conjunto, compare i com a posição k de x após o particionamento:
   - Se $i = k$, x é o elemento procurado;
   - Se $i < k$, então determine recursivamente o i-ésimo menor elemento do subconjunto $A_<$;

Problema da Seleção - terceira solução

5. Finalmente, para encontrar o i-ésimo menor elemento do conjunto, compare i com a posição k de x após o particionamento:
   - Se $i = k$, x é o elemento procurado;
   - Se $i < k$, então determine recursivamente o i-ésimo menor elemento do subconjunto $A_<$;
   - Senão, determine recursivamente o $(i - k)$-ésimo menor elemento do subconjunto $A_>$.

Problema da Seleção - terceira solução

5. Finalmente, para encontrar o i-ésimo menor elemento do conjunto, compare i com a posição k de x após o particionamento:
   - Se $i = k$, x é o elemento procurado;
   - Se $i < k$, então determine recursivamente o i-ésimo menor elemento do subconjunto $A_<$;
   - Senão, determine recursivamente o $(i - k)$-ésimo menor elemento do subconjunto $A_>$.

Note que esta parte é idêntica ao feito em SELECT-NL e em SELECT-ALEAT. O que diferencia este algoritmo dos outros é a escolha do pivô. Escolhendo-se a mediana das medianas, podemos garantir que nenhum lado é muito "grande".

Terceira solução – complexidade

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

1. Divisão em subconjuntos de 5 elementos. $\Theta(n)$

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

1. Divisão em subconjuntos de 5 elementos. $\Theta(n)$
2. Encontrar a mediana de cada subconjunto. $\Theta(n)$

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

1. Divisão em subconjuntos de 5 elementos. $\Theta(n)$
2. Encontrar a mediana de cada subconjunto. $\Theta(n)$
3. Encontrar x, a mediana das medianas. $T(\lceil n/5 \rceil)$

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

1. Divisão em subconjuntos de 5 elementos. $\Theta(n)$
2. Encontrar a mediana de cada subconjunto. $\Theta(n)$
3. Encontrar x, a mediana das medianas. $T(\lceil n/5 \rceil)$
4. Particionamento com pivô x. $O(n)$

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

1. Divisão em subconjuntos de 5 elementos. $\Theta(n)$
2. Encontrar a mediana de cada subconjunto. $\Theta(n)$
3. Encontrar x, a mediana das medianas. $T(\lceil n/5 \rceil)$
4. Particionamento com pivô x. $O(n)$
5. Encontrar o i-ésimo menor de $A_<$ $T(k-1)$
   OU encontrar o $(i-k)$-ésimo menor de $A_>$. $T(n-k)$

Terceira solução – complexidade

$T(n)$: complexidade de tempo no pior caso

1. Divisão em subconjuntos de 5 elementos. $\Theta(n)$
2. Encontrar a mediana de cada subconjunto. $\Theta(n)$
3. Encontrar x, a mediana das medianas. $T(\lceil n/5 \rceil)$
4. Particionamento com pivô x. $O(n)$
5. Encontrar o i-ésimo menor de $A_<$ $T(k-1)$
   OU encontrar o $(i-k)$-ésimo menor de $A_>$. $T(n-k)$

Temos então a recorrência
$T(n) = T(\lceil n/5 \rceil) + T(\max\{k-1, n-k\}) + \Theta(n)$

Terceira Solução - Complexidade

Terceira Solução - Complexidade

O diagrama abaixo classifica os elementos da última figura.

□	□	...	□	□	●	...	●	
□	□	...	□	□	●	...	●	●
□	□	...	□	x	△	...	△	△
●	●	...	●	△	△	...	△	△
●	●	...	●	△	△	...	△	

□ $= \ < x$

△ $= \ > x$

● $=$?

Terceira Solução - Complexidade

O diagrama abaixo classifica os elementos da última figura.

$$
\begin{array}{|ccccccccc|ccc}
\hline
\square & \square & \dots & \square & \square & \bullet & \dots & \bullet & & & & \\
\square & \square & \dots & \square & \square & \bullet & \dots & \bullet & \bullet & \square & = & < x \\
\square & \square & \dots & \square & x & \triangle & \dots & \triangle & \triangle & \triangle & = & > x \\
\bullet & \bullet & \dots & \bullet & \triangle & \triangle & \dots & \triangle & \triangle & \bullet & = & ? \\
\bullet & \bullet & \dots & \bullet & \triangle & \triangle & \dots & \triangle & & & & \\
\hline
\end{array}
$$

O número de elementos $> x$, isto é $\triangle$s, é pelo menos $\frac{3n}{10} - 6$.

Terceira Solução - Complexidade

O diagrama abaixo classifica os elementos da última figura.

□	□	...	□	□	●	...	●	
□	□	...	□	□	●	...	●	●
□	□	...	□	x	△	...	△	△
●	●	...	●	△	△	...	△	△
●	●	...	●	△	△	...	△	

□ $=$ $< x$
△ $=$ $> x$
● $=$?

O número de elementos $> x$, isto é △s, é pelo menos $\frac{3n}{10} - 6$.

De fato, pelo menos $\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil$ grupos contribuem com 3 elementos $> x$, exceto possivelmente o último e aquele que contém x. Portanto,
$3\left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$.

Terceira Solução - Complexidade

Terceira Solução - Complexidade

Similarmente, vale o seguinte.

O número de elementos $< x$, isto é $\square$s, é pelo menos $\frac{3n}{10} - 6$.

Terceira Solução - Complexidade

Similarmente, vale o seguinte.

O número de elementos $< x$, isto é □s, é pelo menos $\frac{3n}{10} - 6$.

Assim, no passo 5 do algoritmo,

$$\max\{k-1, n-k\} \leq n - \left(\frac{3n}{10} - 6\right) \leq \frac{7n}{10} + 6.$$

Terceira Solução - Complexidade

Similarmente, vale o seguinte.

O número de elementos $< x$, isto é □s, é pelo menos $\frac{3n}{10} - 6$.

Assim, no passo 5 do algoritmo,

$$\max\{k-1, n-k\} \leq n - \left(\frac{3n}{10} - 6\right) \leq \frac{7n}{10} + 6.$$

A recorrência $T(n)$ está agora completa:

$$T(n) \leq \begin{cases} \Theta(1), & n \leq 140, \\ T(\lceil n/5 \rceil) + T(\lfloor 7n/10 \rfloor + 6) + \Theta(n), & n > 140. \end{cases}$$

140 é um "número mágico" que faz as contas funcionarem...

Terceira Solução - Complexidade

Similarmente, vale o seguinte.

O número de elementos $< x$, isto é □s, é pelo menos $\frac{3n}{10} - 6$.

Assim, no passo 5 do algoritmo,

$$\max\{k-1, n-k\} \leq n - \left(\frac{3n}{10} - 6\right) \leq \frac{7n}{10} + 6.$$

A recorrência $T(n)$ está agora completa:

$$T(n) \leq \begin{cases} \Theta(1), & n \leq 140, \\ T(\lceil n/5 \rceil) + T(\lfloor 7n/10 \rfloor + 6) + \Theta(n), & n > 140. \end{cases}$$

140 é um "número mágico" que faz as contas funcionarem...

A solução é $T(n) \in \Theta(n)$. (prova no próximo slide)

Solução da recorrência: $T(n) \le cn$

$$
\begin{aligned}
T(n) &\le T(\lceil n/5 \rceil) + T(\lfloor 7n/10 \rfloor + 6) + an \\
&\overset{\text{hi}}{\le} c\lceil n/5 \rceil + c(\lfloor 7n/10 \rfloor + 6) + an \\
&\le c(n/5 + 1) + c(7n/10 + 6) + an \\
&= 9cn/10 + 7c + an \\
&= cn + (-cn/10 + 7c + an) \\
&\le cn
\end{aligned}
$$

onde a última desigualdade vale se $(-cn/10 + 7c + an) \le 0$ para todo $n > 140$.

Isto equivale a exigir que $c \ge 10a(n/(n-70))$. Como $n > 140$, temos que $n/(n-70) \le 2$ e assim basta escolher $c \ge 20a$.

Algoritmo SELECT

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(A, p, r, i)

```
se p = r
    então devolva p    ▷ p e não A[p]
q ← PARTICIONE-BFPRT(A, p, r)
k ← q − p + 1
se i = k
    então devolva q    ▷ q e não A[q]
    senão se i < k
        então devolva SELECT(A, p, q − 1, i)
        senão devolva SELECT(A, q + 1, r, i − k)
```

Particione-BFPRT

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

$$\max\{k-1, n-k\} \le \left\lfloor \frac{7n}{10} \right\rfloor + 6,$$

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

PARTICIONE-BFPRT

- Divida o vetor em $\lfloor n/5 \rfloor$ grupos de tamanho 5 e um grupo ≤ 5,
- ordene cada grupo e determine a mediana de cada um deles,
- determine a mediana das medianas chamando SELECT (!!)
- e particione o vetor em torno desse valor.

Particione-BFPRT

Particione-BFPRT(A, p, r) $\quad \triangleright\ n := r - p + 1$

para $j \leftarrow p, p+5, p+5\cdot 2, \ldots$ **até** $p+5(\lceil n/5\rceil - 1)$ **faça**
    Ordene$(A, j, j+4)$
Ordene$(A, p+5\lfloor n/5\rfloor, n)$

para $j \leftarrow 1$ **até** $\lceil n/5\rceil - 1$ **faça**
    $A[j] \leftrightarrow A[p+5j-3]$
$A[\lceil n/5\rceil] \leftrightarrow A[\lfloor (p+5\lfloor n/5\rfloor + n)/2\rfloor]$

$k \leftarrow$ Select$(A, p, p+\lceil n/5\rceil - 1, \lfloor(\lceil n/5\rceil + 1)/2\rfloor)$

$A[k] \leftrightarrow A[r]$
devolva Particione(A, p, r)

Exercícios

Exercício 0. Em algumas aplicações, quando o vetor A possui elementos repetidos, é mais conveniente ter uma forma mais refinada de fazer a partição. Escreva uma versão de PARTICIONE que recebe um vetor $A[p\,..\,r]$, rearranja A e devolve índices q e t tais que:

- $A[p\,..\,q-1] < x$,
- $A[q\,..\,t-1] = x$ e
- $A[t\,..\,r] > x$.

Exercícios

Exercício 1 (CLRS 9.3-3). Mostre como modificar QUICKSORT de modo que tenha complexidade de tempo $\Theta(n \lg n)$ no pior caso, supondo que todos os elementos são distintos.

Exercício 2 (CLRS 9.3-5). Suponha que você tenha uma subrotina do tipo "caixa-preta" que determina a mediana em tempo linear (no pior caso). Descreva um algoritmo linear simples que resolve o problema da seleção para qualquer i dado na entrada.

Exercícios.

Exercício 3 (CLRS 9-1). Dado um conjunto de n números, queremos imprimir em ordem crescente os i maiores elementos deste usando um algoritmo baseado em comparações.

Compare a complexidade dos seguinte métodos em função de n e i.

- Ordene o vetor e liste os i maiores elementos.
- Construa uma fila de prioridade (max-heap) e chame a rotina EXTRACT-MAX i vezes.
- Use um algoritmo de seleção para encontrar o i-ésimo maior elemento, particione o vetor em torno dele e ordene os i maiores elementos.

Observação: nesta questão você deve pensar que i pode ser uma função de n. Por exemplo, $i = 43$, $i = \lceil \lg n \rceil$ ou $i = \lfloor n/2 \rfloor$.