Aula 07: Algoritmos de busca em arranjos lineares

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada

Entrada
- uma sequência linear $A = \langle A_1, \ldots A_n \rangle$,
- um valor v

Saída
- se $\exists j \mid 1 \leq j \leq n \cdot A_j = v$ então o menor $1 \leq i \leq n$ tal que $A_i = v$, ou
- se $\forall j \mid 1 \leq j \leq n \cdot A_j \neq v$ então NIL.

Algoritmos
- busca linear em arranjo
- busca em arranjo ordenado
  - busca linear
  - busca binária

Algoritmo de busca linear

LINEAR-SEARCH(A, v)

$\quad$ // $\{A = \langle a_1, a_2, \ldots, a_n \rangle\}$

$\quad j = 1$

$\quad$ **while** $A[j] \neq v$ and $j \leq \mathit{length}(A)$

$\qquad\quad j = j + 1$

$\quad$ **if** $j \leq \mathit{length}(A)$

$\qquad\quad$ **return** j

$\quad$ **else return** NIL

$\quad$ // $\{\ (1 \leq r \leq n \Leftrightarrow a_r = v \wedge (\forall i : 1 \leq i < r \Rightarrow a_i \neq v)) \wedge$

$\qquad (r = \text{NIL} \Leftrightarrow (\forall i : 1 \leq i \leq n \Rightarrow a_i \neq v))\}$

Algoritmo de busca linear

- Complexidade no melhor caso
- Complexidade no pior caso
- Complexidade no caso médio

Algoritmo de busca linear

Assumindo o arranjo ordenado

LINEAR-SEARCH(A, v)

// $\{A = \langle a_1, a_2, \ldots, a_n \rangle \wedge (\forall i \mid 1 \leq i < n \cdot a_i \leq a_{i+1})\}$

$j = 1$

while $j \leq \mathit{length}(A)$ and $A[j] < v$

$\quad j = j + 1$

if $j \leq \mathit{length}(A)$ and $A[j] = v$

$\quad$ **return** j

else return NIL

// $\{\ (1 \leq r \leq n \Leftrightarrow a_r = v \wedge (\forall i : 1 \leq i < r \Rightarrow a_i \neq v)) \wedge$
$\quad (r = \text{NIL} \Leftrightarrow (\forall i : 1 \leq i \leq n \Rightarrow a_i \neq v))\}$

Algoritmo não recursivo de busca binária em arranjo ordenado

Assumindo o arranjo ordenado

$\textsc{Binary-Search}(A, v)$

   // $\{A = \langle a_1, a_2, \ldots, a_n \rangle \land (\forall i \mid 1 \leq i < n \cdot a_i \leq a_{i+1})\}$

   $l = 1$

   $u = length(A)$

   $m = \lfloor (length(A)/2 \rfloor$

   while $l \leq u$ and $v \neq A[m]$

      if $v < A[m]$

         $u = m - 1$

      else $l = m + 1$

      $m = (u + l)/2$

   if $v = A[m]$

      return m

   else return NIL

   // $\{ \ (1 \leq r \leq n \Leftrightarrow a_r = v) \land$

      $(r = \text{NIL} \Leftrightarrow (\forall i : 1 \leq i \leq n \Rightarrow a_i \neq v))\}$

Algoritmo não recursivo de busca binária em arranjo ordenado

Assumindo o arranjo ordenado

BINARY-SEARCH(A, v)

// $\{A = \langle a_1, a_2, \ldots, a_n\rangle \wedge (\forall i \mid 1 \leq i < n \cdot a_i \leq a_{i+1})\}$

$l = 1$
$u = length(A)$
$m = \lfloor (length(A)/2 \rfloor$
while $l \leq u$ and $v \neq A[m]$
    if $v < A[m]$
        $u = m - 1$
    else $l = m + 1$
    $m = (u + l)/2$ $\Longleftarrow$ Bug (aritmética em hardware)
if $v = A[m]$
    return m
else return NIL

// $\{\ (1 \leq r \leq n \Leftrightarrow a_r = v) \wedge$
$(r = \text{NIL} \Leftrightarrow (\forall i : 1 \leq i \leq n \Rightarrow a_i \neq v))\}$

Algoritmo não recursivo de busca binária em arranjo ordenado

Assumindo o arranjo ordenado

$\textsc{Binary-Search}(A, v)$

$/\!/ \; \{A = \langle a_1, a_2, \ldots, a_n \rangle \wedge (\forall i \mid 1 \leq i < n \cdot a_i \leq a_{i+1})\}$

$l = 1$

$u = length(A)$

$m = \lfloor (length(A)/2 \rfloor$

while $l \leq u$ and $v \neq A[m]$

    if $v < A[m]$

        $u = m - 1$

    else $l = m + 1$

    $m = l + \lfloor (u + 1 - l)/2 \rfloor$

if $v = A[m]$

    return m

else return $\textsc{nil}$

$/\!/ \; \{ \; (1 \leq r \leq n \Leftrightarrow a_r = v) \wedge$

$\quad (r = \textsc{nil} \Leftrightarrow (\forall i : 1 \leq i \leq n \Rightarrow a_i \neq v))\}$

Algoritmo não recursivo de busca binária

- Complexidade no melhor caso
- Complexidade no pior caso

Algoritmo recursivo de busca binária

Assumindo o arranjo ordenado

$\textsc{Binary-Search}(A, v)$

  // $\{A = \langle a_1, a_2, \ldots, a_n \rangle \wedge (\forall i \mid 1 \leq i < n \cdot a_i \leq a_{i+1})\}$

  return $\textsc{Binary-Search-Range}(A, v, 1, \mathit{length}(A))$

  // $\{\ (1 \leq r \leq n \Leftrightarrow a_r = v) \wedge$

    $(r = \text{NIL} \Leftrightarrow (\forall i : 1 \leq i \leq n \Rightarrow a_i \neq v))\}$

Algoritmo de busca binária

Assumindo o arranjo ordenado

Binary-Search-Range(A, v, l, u)

// $\{A = \langle a_1, a_2, \ldots, a_n \rangle \wedge (\forall i \mid 1 \leq i < n \cdot a_i \leq a_{i+1})\} \wedge$
$1 \leq u, l \leq n$

if $l > u$
    return NIL
else
    $m = l + \lfloor (u + 1 - l)/2 \rfloor$
    if $v = A[m]$
        return m
    elseif $v < A[m]$
        return Binary-Search-Range$(A, v, l, m - 1)$
    else
        return Binary-Search-Range$(A, v, m + 1, u)$

// $\{ \; (l \leq r \leq u \Leftrightarrow a_r = v) \wedge$
$(r = \text{NIL} \Leftrightarrow (\forall i : l \leq i \leq u \Rightarrow a_i \neq v))\}$

Algoritmo de busca binária

Assumindo o arranjo ordenado

- Complexidade no melhor caso
- Complexidade no pior caso
  - $T(n) = T(n/2) + f(n)$, onde $f \in \Theta(1)$.
  - Teorema Master, caso 2
  - $T(n) \in \Theta(\log n)$

PPgSC

Exercício

Considere o problema de contar o número de ocorrências de um valor v em uma sequência A.

1. Assume A não ordenado.
   1.1 Escreva um algoritmo que solucione o problema.
   1.2 Determine a complexidade no melhor caso, e no pior caso.
2. Assume A ordenado.
   2.1 Adapte o algoritmo anterior para levar esta hipótese em conta.
   2.2 Determine a complexidade no melhor caso, e no pior caso.
   2.3 Escreva um novo algoritmo, inspirado do algoritmo da busca binária.
   2.4 Determine a complexidade, no pior caso, do novo algoritmo.

PPgSC