"Reivindicações extraordinárias requerem evidências extraordinárias" (Marcello Truzzi).

Busca em Vetores Ordenados

Paulo Ricardo Lisboa de Almeida

DSBD

Alguns conteúdos e exemplos foram baseados nas aulas do Professor Renato Carmo - www.inf.ufpr.br/renato

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Vetor ordenado

Um vetor v indexado por $[a..b]$ é ordenado se seus elementos estão em **ordem não decrescente**.

Vetor ordenado

Um vetor v indexado por $[a..b]$ é ordenado se seus elementos estão em **ordem não decrescente**.

Às vezes na matemática parece que complicamos as coisas de propósito (o que não é o caso). Por que dizer “não decrescente”, ao invés de simplesmente dizer “crescente”?

Vetor ordenado

Um vetor v indexado por $[a..b]$ é ordenado se seus elementos estão em **ordem não decrescente**.

A ordem crescente implica que valores repetidos não são permitidos.

Vetor ordenado

Um vetor v indexado por $[a..b]$ é ordenado se seus elementos estão em **ordem não decrescente.** Ou seja, $v[i] \leq v[j], \forall i < j$.

Vetor ordenado

Um vetor v indexado por $[a..b]$ é ordenado se seus elementos estão em **ordem não decrescente.** Ou seja, $v[i] \leq v[j], \forall i < j$.

Criar um algoritmo que dado um valor *x*, busca a posição onde x deve ocupar no vetor de forma a manter o vetor ordenado.

Vetor ordenado

Um vetor v indexado por $[a..b]$ é ordenado se seus elementos estão em **ordem não decrescente.** Ou seja, $v[i] \leq v[j], \forall i < j$.

função busca (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o **menor** m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]

Exemplo

i	1	2	3	4	5	6	7	8
v[i]	4	8	8	15	16	16	23	42

buscaPos(15, v, 1, 8) = 4
buscaPos(16, v, 1, 8) = 6
buscaPos(8, v, 1, 8) =3
buscaPos(10, v, 1, 8) = 3
buscaPos(4, v, 1, 8) = 1
buscaPos(1, v, 1, 8) = 0
buscaPos(42, v, 1, 8) = 8
buscaPos(50, v, 1, 8) = 8

Implementação Ingênua

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
     retorne a-1
se v[b] ≤ x
     retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Faça você mesmo

i	1	2	3	4	5	6	7	8
v[i]	4	8	8	15	16	16	23	42

Faça um teste de mesa considerando o vetor *v* e a seguinte chamada

buscaSequencial(20,v,1,8)

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
     retorne a-1
se v[b] ≤ x
     retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Implementação Ingênua

Como fica a função de custo para o número de comparações com elementos do vetor para a implementação ingênua?

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
      retorne a-1
se v[b] ≤ x
      retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Implementação Ingênua

Como fica a função de custo para o número de comparações com elementos do vetor para a implementação ingênua?

$$C(x, v, a, b) = \begin{cases} 0, & \text{se } a > b, \\ 1, & \text{se } a \leq b \text{ e } v[b] \leq x, \\ 1 + C(x, v, a, b-1), & \text{se } a \leq b \text{ e } v[b] > x, \end{cases}$$

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
      retorne a-1
se v[b] ≤ x
      retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Pior caso

$$C^+(n) = max\{C(x, v, a, b) | b - a + 1 = n\}$$

$$C^+(n) = 1 + C^+(n-1) = 1 + (1 + C^+(n-2)) = \ldots$$
$$= \mu + C^+(n - \mu)$$
$$C^+(n) = n + C^+(n-n) = n + C^+(0) = n$$

Melhor Caso

$$C^-(n) = \begin{cases} 0, & \text{se } n \leq 0, \\ 1, & \text{se } n > 0 \end{cases}$$

$$C^-(n) = 1$$

Então

Sabemos que o melhor caso é $C^{-}(n) = 1$ e o pior caso é $C^{+}(n) = n$. Sendo assim, para uma instância qualquer do problema $buscaSequencial(x, v, a, b)$, onde $n = b - a + 1$, o custo $C(x, v, a, b)$ vai ser um valor $1 \leq C(x, v, a, b) \leq n$

Considerações

Note que o problema é muito parecido com o problema de buscar o índice do valor *x* no vetor *v*, onde *v* é ordenado.

Podemos trivialmente usar o algoritmo desenvolvido para indicar a posição de *x* no vetor, ou se *x* não existe.

Fica como exercício.

Considerações

Note que o problema é muito parecido com o problema de buscar o índice do valor *x* no vetor *v*, onde *v* é ordenado.

Podemos trivialmente usar o algoritmo desenvolvido para indicar a posição de *x* no vetor, ou se *x* não existe.

Fica como exercício.

Como o vetor é ordenado, tão logo os valores se tornem menores que x, podemos parar de verificar.

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
      retorne a-1
se v[b] ≤ x
      retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Considerações

Note que o problema é muito parecido com o problema de buscar o índice do valor *x* no vetor *v*, onde *v* é ordenado.

Podemos trivialmente usar o algoritmo desenvolvido para indicar a posição de *x* no vetor, ou se *x* não existe.

Fica como exercício.

Como o vetor é ordenado, tão logo os valores se tornem menores que x, podemos parar de verificar.

Mas isso **não nos ajudou em nada no pior caso.**

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
      retorne a-1
se v[b] ≤ x
      retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Considerações

Note que o problema é muito parecido com o problema de buscar o índice do valor *x* no vetor *v*, onde *v* é ordenado.

Podemos trivialmente usar o algoritmo desenvolvido para indicar a posição de *x* no vetor, ou se *x* não existe.

Fica como exercício.

Como o vetor é ordenado, tão logo os valores se tornem menores que x, podemos parar de verificar.

Mas isso **não nos ajudou em nada no pior caso.**

Obs.: na verdade isso nos ajuda no caso médio, mas vamos nos contentar com pior e melhor casos na disciplina.

```
função buscaSequencial (x,v,a,b)
entrada: vetor v ordenado, indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: o menor m ∈ [a-1..b], tal que x < v[i] ∀i ∈ [m+1..b]
se a > b
      retorne a-1
se v[b] ≤ x
      retorne b
retorne buscaSequencial(x,v,a,b-1)
```

Exercícios

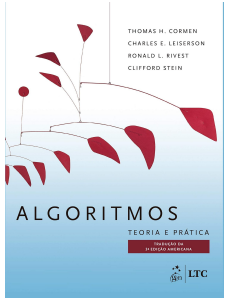
1. Implemente a versão ingênua (dada em aula) do algoritmo de busca de posição em C
2. Crie uma função que usa a resposta da busca ingênua para dizer se *x* está ou não no vetor
3. Mostre a versão iterativa do algoritmo de busca de posição e
    a. Encontre a invariante e demonstre através dela que o algoritmo está correto
    b. Implemente em C
4. Considere o algoritmo de Euclides para encontrar o máximo divisor comum entre dois inteiros, dado pela recorrência

$$mdc(a,b) = \begin{cases} a, & \text{se } b = 0, \\ mdc(b, a \mod b), & \text{se } b \neq 0 \end{cases}$$

    a. Crie um algoritmo recursivo para o mdc
        i. Implemente em C
    b. Criem um algoritmo iterativo para o mdc
        i. Implemente em C

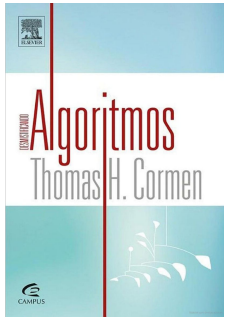
Referências

T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012

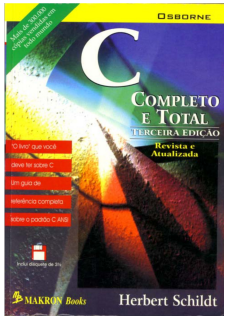


R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2014

T. Cormen. Desmistificando algoritmos. 2017.



H. Schildt. C completo e total. 1996



Licença