> Os humanos são pensadores lentos, desleixados e brilhantes.
> Computadores são rápidos, cuidadosos e estúpidos.

Busca em Vetores

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Faça você mesmo

Considere o problema de busca em um vetor não ordenado

Seja um vetor v indexado por $[a..b]$, com $a < b$, e um valor x. Obtenha um índice $m \in [a..b]$ tal que $v[m] = x$, ou nao caso o tal m não exista.

1. Crie um algoritmo iterativo para resolver o problema
    a. Implemente o algoritmo também em C
2. Crie um algoritmo recursivo para resolver o problema
    a. Implemente o algoritmo também em C

Dica: no seu algoritmo, use *retorne nao* caso o valor não seja encontrado. Na implementação em C, retorne -1 caso o valor não seja encontrado.

Faça você mesmo

Considere o problema de busca em um vetor não ordenado

Seja um vetor v indexado por $[a..b]$, com $a < b$, e um valor x. Obtenha um índice $m \in [a..b]$ tal que $v[m] = x$, ou nao caso o tal m não exista.

```
função busca(x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
     retorne nao
se v[b] = x
     retorne b
retorne busca(x,v,a,b-1)
```

Melhor e pior caso

Melhor caso: No melhor caso, quantos "recursos" (e.g., comparações, joules, tempo, ...) precisamos gastar para resolver o problema?

Temos um piso. Sabemos que na melhor das hipóteses, o algoritmo vai gastar a quantidade de recursos do melhor caso

Pior caso: No pior caso, quantos "recursos" precisamos gastar para resolver o problema?

Temos um teto. Sabemos que na pior das hipóteses, o algoritmo vai gastar a quantidade de recursos do pior caso

Melhor e pior caso

Intuitivamente, considerando o número de comparações com elementos do vetor, qual o melhor e o pior caso para o algoritmo de busca?

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
     retorne nao
se v[b] = x
     retorne b
retorne busca(x,v,a,b-1)
```

Melhor e pior caso

Intuitivamente, considerando o número de comparações com elementos do vetor, qual o melhor e o pior caso para o algoritmo de busca?

Melhor caso: encontramos o item na primeira chamada da função, então uma comparação é feita

Pior caso: passamos por todos os itens do vetor e comparamos, sem encontrar *x*

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
     retorne nao
se v[b] = x
     retorne b
retorne busca(x,v,a,b-1)
```

Análise

Seja *C(x, v, a, b)* a função que computa o número de comparações (de x com o vetor) realizadas em *busca (x,v,a,b)*.

$$C(x, v, a, b) = \begin{cases} ? \end{cases}$$

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
      retorne nao
se v[b] = x
      retorne b
retorne busca(x,v,a,b-1)
```

Análise

Seja *C(x, v, a, b)* a função que computa o número de comparações (de x com o vetor) realizadas em *busca (x,v,a,b)*.

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

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
      retorne nao
se v[b] = x
      retorne b
retorne busca(x,v,a,b-1)
```

Análise

Seja *C(x, v, a, b)* a função que computa o número de comparações (de x com o vetor) realizadas em *busca (x,v,a,b)*.

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

Problema: a quantidade de comparações não depende (apenas) do tamanho do vetor

Análise

Seja *C(x, v, a, b)* a função que computa o número de comparações (de x com o vetor) realizadas em *busca (x,v,a,b)*.

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

C(n-1) vai ser 0, 1, ou 1 + C(n -1). Precisamos analisar como melhor ou pior caso para resolver o problema.

Problema: a quantidade de comparações não depende (apenas) do tamanho do vetor

Pior Caso

Vamos definir a função de custo de pior caso como $C^*(n)$

Quando o pior caso acontece no algoritmo?

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
      retorne nao
se v[b] = x
      retorne b
retorne busca(x,v,a,b-1)
```

Pior Caso

Vamos definir a função de custo de pior caso como $C^*(n)$

Quando o pior caso acontece no algoritmo?

Quando passamos por todos elementos sem encontrar x (o retorno é **nao**)

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
      retorne nao
se v[b] = x
      retorne b
retorne busca(x,v,a,b-1)
```

Pior Caso

Vamos definir a função de custo de pior caso como $C^+(n)$

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

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
      retorne nao
se v[b] = x
      retorne b
retorne busca(x,v,a,b-1)
```

Pior Caso

Vamos definir a função de custo de pior caso como $C^+(n)$

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

"$C^+(n)$ é o máximo valor no conjunto de todos os valores possíveis de $C(x,v,a,b)$ tal que $b-a+1=n$, ou seja, para todos os vetores de tamanho n."

```
função busca (x,v,a,b)
entrada: vetor v indexado por [a..b], com a ≤ b, e um valor x a ser buscado
saída: m ∈ [a..b], tal que v[m] = x, ou nao se x não existe no vetor
se a > b
      retorne nao
se v[b] = x
      retorne b
retorne busca(x,v,a,b-1)
```

Pior Caso

Como o pior caso é dado quando o valor x não é encontrado:

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

Pior Caso

Como o pior caso é dado quando o valor x não é encontrado:

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

Agora basta resolver a recorrência

Pior Caso

Como o pior caso é dado quando o valor x não é encontrado:

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

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

No caso base, $n - \mu = 0$, logo $\mu = n$. Então

$$C^+(n) = n + C^+(n-n) = n + C^+(0) = n$$

Melhor caso

Usando um raciocínio análogo, a função de custo $C^-(n)$ de melhor caso ocorre quando encontramos *x* na primeira chamada da função. Definimos

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

Melhor caso

O melhor caso ocorre quando encontramos *x* na primeira chamada da função

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

Logo, $C^{-}(n) = 1$

Resumindo

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 $busca(x, v, a, b)$, onde $n = b - a + 1$, o custo $C(x, v, a, b)$vai ser um valor

$$1 \leq C(x, v, a, b) \leq n$$

Exercícios

1. Considere o problema

   Seja um vetor v indexado por $[a..b]$, com $a < b$, e dois valores x e y. Obtenha um índice $m \in [a..b]$ tal que $x < v[m] < y$, ou nao caso o tal m não exista.

   a. Crie um algoritmo **recursivo** que recebe o vetor, e retorna *m*, ou *nao* caso *m* não exista
   b. Implemente o algoritmo em C. Retorne -1 para nao.
   c. Encontre a função de recorrência para o número de comparações entre elementos do vetor
   d. Resolva a recorrência para o melhor e para o pior caso

2. Um vetor v[a..b] é palíndromo se seu conteúdo é igual se o lermos da esquerda para a direita, ou da direita para a esquerda. Por exemplo, o vetor v = (1,2,3,2,1) é palíndromo. Ou seja

$$(v[a], v[a+1], ..., v[b-1], v[b]) = (v[b], v[b-1], \ldots, v[a+1], v[a])$$

   a. Crie um algoritmo **recursivo** que recebe o vetor, e retorna *sim* se o vetor é palíndromo, ou *nao* caso contrário
   b. Implemente o algoritmo em C. Retorne 1 para sim, e 0 para não.
   c. Encontre a função de recorrência para o número de comparações entre elementos do vetor
   d. Resolva a recorrência para o melhor e para o pior caso

Exercícios

3. Considere o problema para vetores ordenados

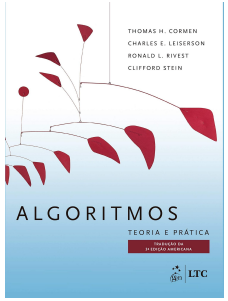
> Seja um vetor v indexado por $[a..b]$, com $a < b$, onde os elementos de v estão em **ordem não decrescente**, ou seja, $v[i] \leq v[j], \forall i < j$. Seja um valor x. Obtenha um índice $m \in [a..b]$ tal que $v[m] = x$, ou nao caso o tal m não exista.

Para ***a.*** e ***b.*** crie a **sua** melhor (mais eficiente) solução para o problema. **Não trapaceie** buscando na internet ou em livros. Vamos usar as ideias iniciais de solução na próxima aula.

a. Crie um algoritmo **iterativo** que recebe o vetor, e retorna *m*, ou *nao* caso *m* não exista
b. Crie um algoritmo **recursivo** que recebe o vetor, e retorna *m*, ou *nao* caso *m* não exista
c. Implemente a. e b. algoritmo em C. Retorne -1 para nao.
    i. Para facilitar, é dado um esqueleto de um programa em C que gera vetores aleatórios e os ordena. O algoritmo de ordenação dado é o Selection Sort. Vamos estudar esse algoritmo em detalhes no futuro.

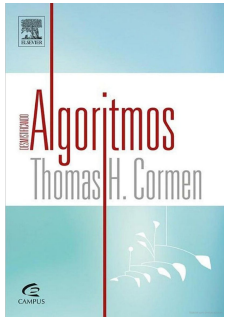
Referências

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

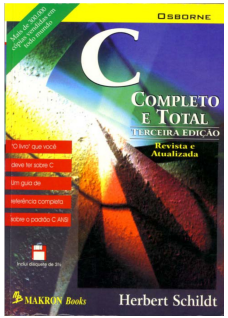


R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2014

T. Cormen. Desmistificando algoritmos. 2017.



H. Schildt. C completo e total. 1996



Licença