**Problema inicial**

**Dado um inteiro x e um vetor v[0..n-1] de inteiros, o problema da busca consiste em encontrar x em v, ou seja, encontrar um índice k tal que v[k] == x.**

**x=2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 5 | 3 | 7 | 4 | 2 | 8 | 4 | 6 |
|  |  |  |  |  |  |  |  |

v[k] = x

ideia de algoritmo da **busca linear**

começando na primeira posição do vetor comparar cada uma das posições (v[k]=x) para verificar se x está no vetor.

vamos considerar que iremos implementar a solução através de uma função.

public static int busca( int v[], int x ){

for( int k = 0; k < v.length; k++ )

if ( v[k] == x )

return k; // achou

// não achou

return -1;

}

Questões

**o que a função retorna quando encontrar o x no vetor ?**

retornamos o valor do índice k.

**o que a função retorna quando não encontrar o x no vetor ?**

retornamos -1 caso não encontro e elemento no vetor

**o que fazer se aparece o valor de x duas ou mais vezes no vetor ?**

retorna o índice da 1ª ocorrência de x no vetor,

As três seriam como se fosse 3 configurações de entrada

Exercícios:

1. modifique a busca linear de tal forma que ao invés de retornar o índice da 1ª ocorrência de um valor repetido retorne a última ocorrência.

public static int buscaUltimo( int v[], int x ){

int i=-1;

for( int k = 0; k < v.length; k++ )

if ( v[k] == x )

i = k;

// não achou

return i;

}

1. Critique a função busca abaixo ela funciona ? sempre ?

public static int buscaCriticar( int x, int v[]) {

int m = 0;

while (v[m] < x && m < v.length)

m++;

if (v[m] == x)

return m;

else

return -1;

}

não funciona pois a função considera que o vetor está em ordem crescente

Considere novamente o problema:

**Dado um inteiro x e um vetor v[0..n-1] de inteiros, o problema da busca consiste em encontrar x em v, ou seja, encontrar um índice k tal que v[k] == x.**

**Considere que agora o vetor está em ordem crescente.**

**x=40**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4 | 9 | 15 | 16 | 40 | 47 | 78 | 91 |
|  |  |  |  | i  f  m |  |  |  |

meio do vetor do vetor será o índice da primeira posição somado ao índice da última posição divido por 2.

m =(i+f)/2

v[k] = x

Como ficaria a função que implementa a busca feita no vídeo <https://www.youtube.com/watch?v=iDVH3oCTc2c>

public static int buscaBinaria( int v[], int x ){

int i = 0;

in f = v.length - 1;

while( i <= f ){

m = (i+f)/2;

if( x == v[m]) // achei

return m; // return o indice

if( x > v[m] ) // descarta tudo que está à esquerda

i = m+1;

else if ( x < v[m]) // descarta tudo que está à direita

f = m-1;

}

return -1;

}

Para saber mais veja a simulação no site abaixo

<https://www.cs.usfca.edu/~galles/visualization/Search.html>

Para praticar !!!

Reescreva a função de busca binária para que ela funcione em um vetor decrescente.

**Análise da Busca Linear e Busca Binária**

O **pior caso** da **busca linear** é quando **o elemento não está no vetor**, nesse caso a **busca linear** executa **N** passos para um vetor de tamanho N no seu **pior caso**.

O **pior caso** da **busca binária** é quando **o elemento não está no vetor**, nesse caso a **busca binária** executa **log2N + 1** passos para um vetor de tamanho N no seu **pior caso**.

Análise da busca binária:

|  |  |  |
| --- | --- | --- |
| tamanho do vetor = N | potencia de 2 | passos |
| **1** | **N=20** | **1** |
| **2** | **N=21** | **2** |
| **4** | **N=22** | **3** |
| **8** | **N=23** | **4** |
| 9 |  | 4 |
| 10 |  | 4 |
| ... |  | ... |
| 15 |  | 4 |
| **16** | **N=24** | **5** |
| ... |  | ... |
| 31 |  | 5 |
| **32** | **N=25** | **6** |
| ... |  | ... |
| 63 |  | 6 |
| **64** | **N=26** | **7** |
|  |  |  |
|  |  |  |
| 1024 | **N=210** | 11 |

Então o número de passos é **o expoente da potência** **mais 1**. Então o 1024 é 11.

Então o número de passos é **log2N + 1**. Então o 1024 é log21024=10 e 10 + 1 é 11.

Agora dá pra gente comparar quais dos dois algoritmos é melhor.

|  |  |  |
| --- | --- | --- |
| tamanho do vetor = N | passos busca linear  passos=N | passos da busca binária  passos = log2N+1 |
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 4 | 4 | 3 |
| 8 | 8 | 4 |
| 9 | 9 | 4 |
| 10 | 10 | 4 |
| ... |  | ... |
| 15 | 15 | 4 |
| 16 | 16 | 5 |
| ... |  | ... |
| 31 | 31 | 5 |
| 32 | 32 | 6 |
| ... |  | ... |
| 63 | 63 | 6 |
| 64 | 64 | 7 |
|  |  |  |
|  |  |  |
| 1024 | 1024 | 11 |

Dado a tabela acima notamos que conforme o tamanho do vetor aumenta o número de passos da busca linear aumenta mais que da busca binária.