Linguagem de Programação

Pesquisa em *Arrays*

ECT2303

helton.maia@ect.ufrn.br

Pesquisa em *Arrays* - Introdução

Motivação:

- O armazenamento de grandes quantidades de dados exige soluções rápidas para a busca de informações em um *array*;
- Determinar a frequência de ocorrência de um certo dado em um *array* é essencial para vários problemas de busca e acesso à informação.

Pesquisa em *Arrays* - Exemplos

- Pesquisar um número de identificação (Ex: RG/CPF) em um uma base de dados com milhões de registros;
- Encaminhar adequadamente um e-mail para o referido destinatário;
- Buscar a melhor rota para o deslocamento de um veículo;
- Pesquisar por uma palavra chave em um texto.

Pesquisa em *Arrays* - Exemplos

Considere o seguinte *array* e a chave para busca abaixo:

```
int arr[] = {2, 6, 25, 24, 35, 68, 85, 76, 10};
int chave = 25;
```

Temos como resultado, que a chave está na posição 2 do vetor de inteiros;

Pesquisa em *Arrays* - Tipos

Pesquisa Linear:

- O tempo gasto para pesquisar elementos continua aumentando à medida que o número de elementos aumenta;
- Possui complexidade de O(*n*) no pior caso.

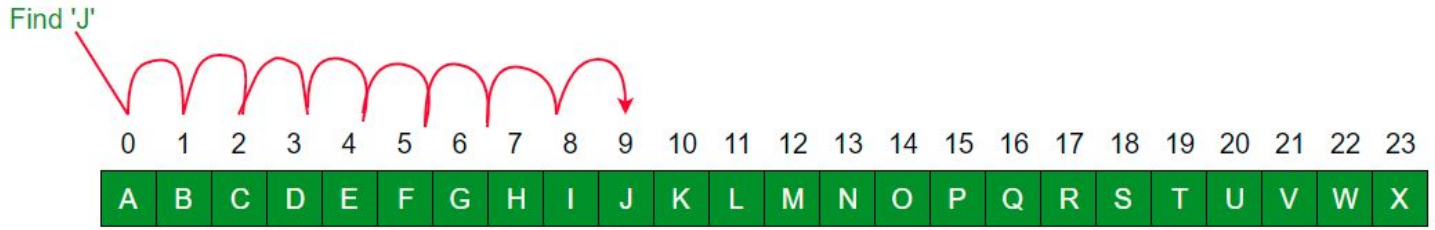
Pesquisa Binária:

- O *array* precisa estar ordenado;
- Bastante eficiente, a cada comparação, metade dos elementos do array é desconsiderado;
- Possui complexidade de O(log *n*).

Pesquisa em *Arrays* - Linear

Pesquisa Linear:

- Compara individualmente cada elemento do array com a chave de pesquisa definida;
- Funciona para *Arrays* ordenados e não ordenados;
- O valor a ser buscado pode aparecer em qualquer parte do array, sendo em média, pesquisado em metade dos elementos



fonte: https://www.geeksforgeeks.org/linear-search-vs-binary-search/

Pesquisa em *Arrays* - Linear (Code)

Dado um array **v[ ]** de n elementos, escreva uma função para procurar um dado elemento (**chave)** em **v[ ]**. Sua função deve retornar **-1** caso a chave não seja encontrada.

```
int pesquisaLinear(int v[], int chave, int tamanho);
```

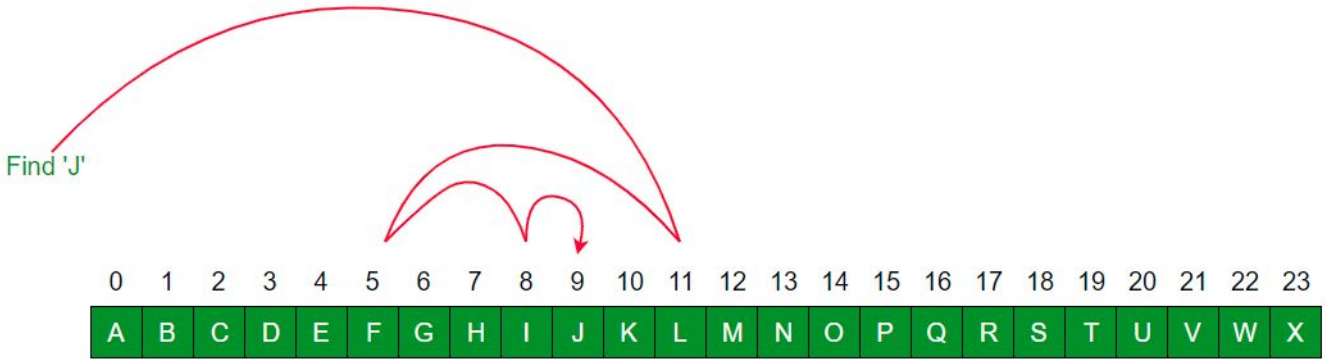
Pesquisa em *Arrays* - Linear (Code)

```cpp
#include <iostream>
using namespace std;

int buscaLinear(int v[], int chave, int tam){
    int i;
    for (i = 0; i < tam; i++)
        if (v[i] == chave)
            return i;
    return -1;
}

int main(){
    int arr[] = { 2, 3, 4, 10, 40 };
    int chave = 40;
    int tamanho = sizeof(arr) / sizeof(arr[0]);

    int resultado = buscaLinear(arr, chave, tamanho);

    (resultado == -1)? cout<<"Elemento não faz parte do array"
       : cout<<"Elemento faz parte do array, no indice: " << resultado;
return 0;
}
```

Pesquisa em *Arrays* - Binária

Pesquisa Binária:

- Mais eficiente do que a linear, contudo o array precisa estar ordenada;
- Eliminamos metade dos elementos do array (ordenado) à cada comparação, acelerando a busca.



fonte: https://www.geeksforgeeks.org/linear-search-vs-binary-search/

Pesquisa em *Arrays* - Binária (Problema)

Algoritmo: Deve-se localizar o elemento contido no meio do *array,* e compará-lo com a chave de busca já definida, se forem iguais, a busca é encerrada pois o elemento foi encontrado. Senão, se a chave for menor que o elemento do meio do *array*, repita o procedimento para a primeira metade do *array*. No entanto, se a chave for maior que o elemento do meio do *array*, repita a busca para a segunda metade do *array,* e assim por diante. Caso não seja encontrado nenhum elemento, sua função deve retornar -1.

```
int pesquisaBinaria(int v[], int chave, int primeiro, int ultimo);
```

Pesquisa em *Arrays* - Binária (Problema)

```cpp
#include <iostream>
using namespace std;

int buscaBinaria(int v[], int chave, int primeiro, int ultimo){
    int meio;
    while(primeiro < ultimo)
    {
        meio = (primeiro+ultimo)/2;
        if(v[meio]==chave)         {
            return meio;
        }else if(chave < v[meio]){
            ultimo = meio - 1;
        }
        else{ primeiro = meio + 1;
        }
    }
    return -1;
}
int main(){
    int arr[] = { 2, 3, 4, 10, 50 };
    int chave = 50;
    int tamanho = sizeof(arr) / sizeof(arr[0]);
    int resultado = buscaBinaria(arr, chave, 0, tamanho);

    (resultado == -1) ? cout << "Elemento nao faz parte do array"
    : cout << "Elemento faz parte do array, no indice: " << resultado;

    return 0;
}
```

Perguntas ?