Buscas

Prof. Marcel Hugo

Estruturas de Dados

Departamento de Sistemas e Computação
Universidade Regional de Blumenau – FURB

1

Introdução – Conceitos Básicos

- Estudo de como recuperar dados a partir de uma grande massa de dados previamente armazenada.
- Os dados estão divididos em registros ou objetos.
- Cada registro/objeto possui uma chave para ser usada na pesquisa.
- **Objetivo da pesquisa**: Encontrar uma ou mais ocorrências de registros com chaves iguais à chave de pesquisa.
- **Pesquisa com sucesso**

X **Pesquisa sem sucesso.**

2

Introdução – Conceitos Básicos

- **Tabelas**
  - Conjunto de registros ou arquivos
  - Tabela: associada a entidades de vida curta, criadas na memória interna durante a execução de um programa.
  - Arquivo: geralmente associado a entidades de vida mais longa, armazenadas em memória externa.
  - Distinção não é rígida:
    - tabela: arquivo de índices
    - arquivo: tabela de valores de funções.

3

Escolha do Método de Pesquisa mais Adequado a uma Determinada Aplicação

- Depende principalmente:
  1. Quantidade de dados envolvidos.
  2. Arquivo estar sujeito a inserções e retiradas freqüentes.
- Se conteúdo do arquivo é estável é importante minimizar o tempo de pesquisa, sem preocupação com o tempo necessário para estruturar o arquivo.

4

Buscas em vetor

- Considerar função em que:
  - Os parâmetros da função sejam:
    - Um vetor de dados/objetos denominado **`info`**.
    - Um *valor de busca* denominado **`valorBuscar`**.
  - O retorno da função seja:
    - `i`, se o elemento **`valorBuscar`** encontrar-se em **`info[i]`**
    - `-1`, se o elemento **`valorBuscar`** não constar em **`info`**.
- Dois algoritmos básicos:
  - Busca linear (pesquisa sequencial)
  - Busca binária (pesquisa binária)

5

Busca linear

- Percorre o vetor, partindo da primeira posição, até encontrar um elemento que armazena o *valor de busca* ou até atingir o final do vetor.

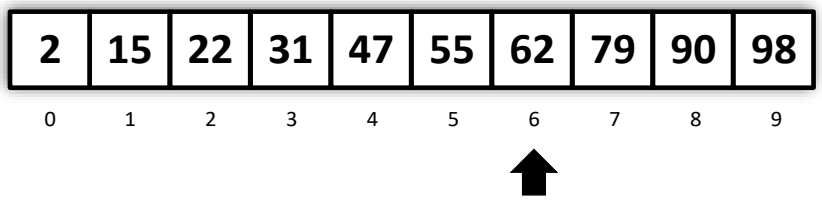
```
Algoritmo: buscaLinear(int[] info, int valorBuscar)

int n ← size(info);
para i ←0 até n-1 faça
  se info[i] = valorBuscar então // encontrado
    retornar i;
  fim-se;
fim-para;

retornar -1; // não encontrado
```

6

Busca linear em vetor ordenado



- Para buscar o valor 57, por exemplo, não é preciso percorrer toda a estrutura de dados

A partir desta posição não precisa mais procurar

Isto é, quando o *valor de busca* for inferior ao valor do vetor, desiste de procurar

8

Busca linear em vetor ordenado

```
Algoritmo: buscaLinearVetorOrdenado(int[] info, int valorBuscar)

int n ← size(info);
para i ←0 até n-1 faça
  se info[i] = valorBuscar então
    retornar i;
  senão
    se valorBuscar < info[i] então
      break;
    fim-se;
  fim-se;
fim-para;
retornar -1;
```

9

Busca binária

- Utilizada quando a estrutura de dados está ordenada.
- Compara o *valor de busca* com o elemento do *meio* do vetor e:
  - **Se o *valor de busca* for igual ao do vetor**:
    - elemento localizado. Finaliza o algoritmo
  - **Se o *valor de busca* for menor ao do vetor**:
    - procura novamente na primeira metade do vetor
  - **Se o *valor de busca* for maior ao do vetor**:
    - procura novamente na segunda metade do vetor
- O procedimento é repetido, subdividindo o vetor até encontrar o elemento ou o sub-vetor atingir tamanho 0.

Busca binária

- Procurar o número 55 no vetor:

2	15	22	31	47	55	62	79	90	98

$meio \leftarrow \lfloor (inicio + fim)/2 \rfloor;$
$meio \leftarrow \lfloor (0 + 9)/2 \rfloor;$
$meio \leftarrow 4;$

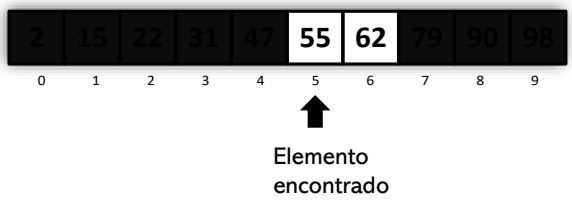
2	15	22	31	47	55	62	79	90	98
0	1	2	3	4	5	6	7	8	9

$meio \leftarrow \lfloor (inicio + fim)/2 \rfloor;$
$meio \leftarrow \lfloor (5 + 9)/2 \rfloor;$
$meio \leftarrow 7;$

2	15	22	31	47	55	62	79	90	98
0	1	2	3	4	5	6	7	8	9

Busca binária

meio ← ⌊(inicio + fim)/2⌋;
meio ← ⌊(5 + 6)/2⌋;
meio ← 5;



13

Algoritmo iterativo para busca binária

```
Algoritmo: buscaBinaria(int[] info, int valorBuscar)

n ← size(info);
inicio ← 0;
fim ← n-1;
enquanto inicio ≤ fim faça
  meio ← ⌊(inicio + fim)/2⌋;
  se valorBuscar < info[meio] então
    fim ← meio-1; // redefine posição final
  senão
    se valorBuscar > info[meio] então
      inicio ← meio+1; // redefine posição inicial
    senão
      retornar meio; // achou
    fim-se;
  fim-se;
fim-enquanto;
retornar -1;
```

14