Métodos de Busca em Arranjos de Elementos

Códigos de Alta Performance

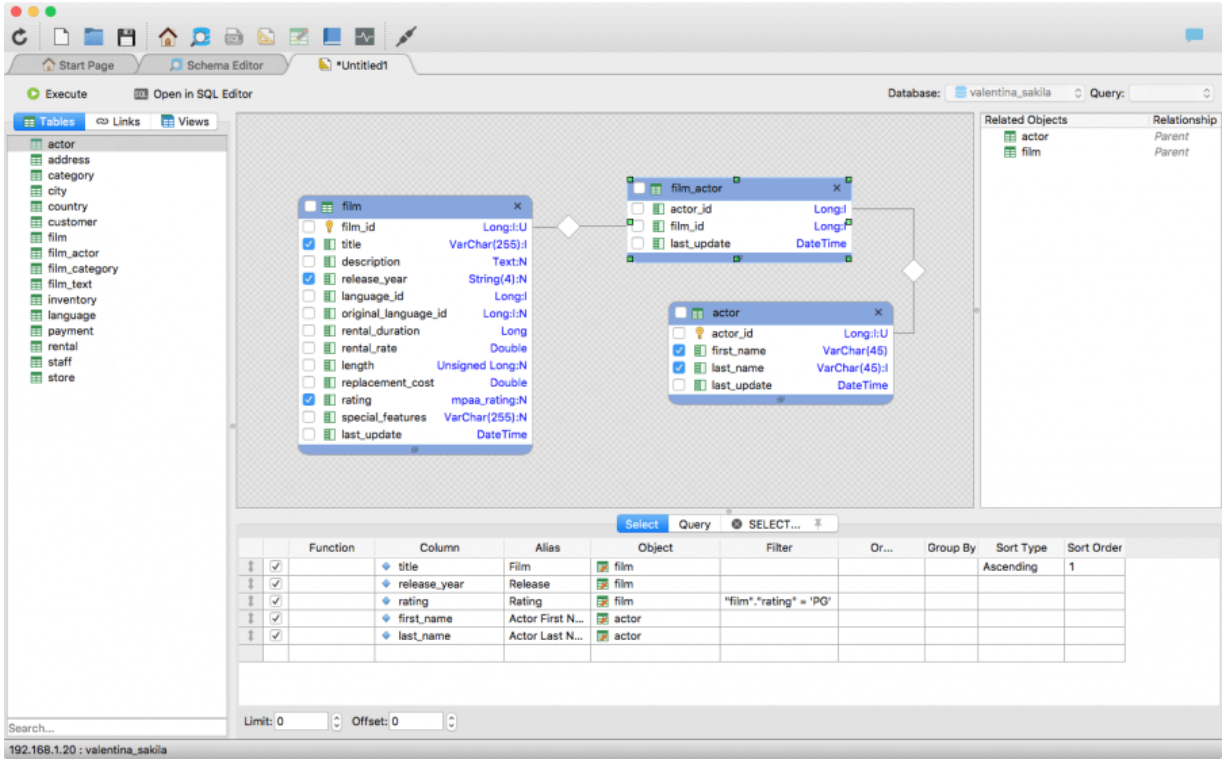
PROFa. PATRÍCIA MAGNA - profpatricia.magna@fiap.com.br

Motivação

- Um arquivo é formado por um conjunto de registros.
- Um registro é um conjunto de informações relacionadas entre si.
- A cada registro de uma coleção de registros associa-se uma chave, que é uma informação que distingue aquele registro dos demais.
  - Se a chave somente ocorre uma vez na coleção é chamada chave primária.
  - No caso da possibilidade de múltiplas ocorrências de uma chave, esta é dita chave secundária.

Método ou Função de Busca ou de Pesquisa

- Aceita uma chave como argumento
- Tenta descobrir um registro cuja chave seja coincidente com este argumento e retorna:
  - registro encontrado ou um ponteiro para o mesmo.
  - Sinalização na condição de inexistência de um registro com a chave recebida.



Escolha do método

- Escolha do melhor algoritmo é função de
  - Quantidade de dados
  - Arquivo com a inserções/remoções frequentes

- Se o arquivo for estável (ou seja, que sofre pouco operações de inserção e remoção de dados), é preferível minimizar o tempo de pesquisa, mesmo que para isso gaste-se muito tempo para organizar o arquivo

Algoritmos de Pesquisa ou Busca

- A escolha do melhor algoritmo depende, muitas vezes, do estado de ordenação do arquivo

- Tipos mais comuns de algoritmos de busca
  - Busca Sequencial Exaustiva
  - Busca Sequencial
  - Busca Binária

Busca Sequencial Exaustiva

Busca Sequencial Exaustiva

- A técnica de busca mais simples é a varredura sequencial exaustiva da coleção de registros.
- Percorre-se o conjunto até o final, independentemente da descoberta de registro ou registros que satisfaçam à condição de busca.
- A chave procurada pode não ser chave primária

Procurando todas as ocorrências da palavra NERD em um livro

Exemplo de Função Busca Sequencial Exaustiva

```
public static Registro[] buscaSequencialExaustiva (Registro bd[], int chave)
    {
    int i, ne = 0;int num = bd.length;
    Registro encontrados[] = new Registro[num];
    for (i = 0; i < num; i++)
    {
    if (bd[i].getChave() == chave){
            encontrados[ne] = bd[i];
          /*armazena registro da posição em que a chave foi encontrada */
     ne++;
    }
    }
    return(encontrados);/* registros com a chave procurada*/
    }
```

Projeto MétodosBusca pacote BuscaExaustiva

Considerações sobre o Algoritmo

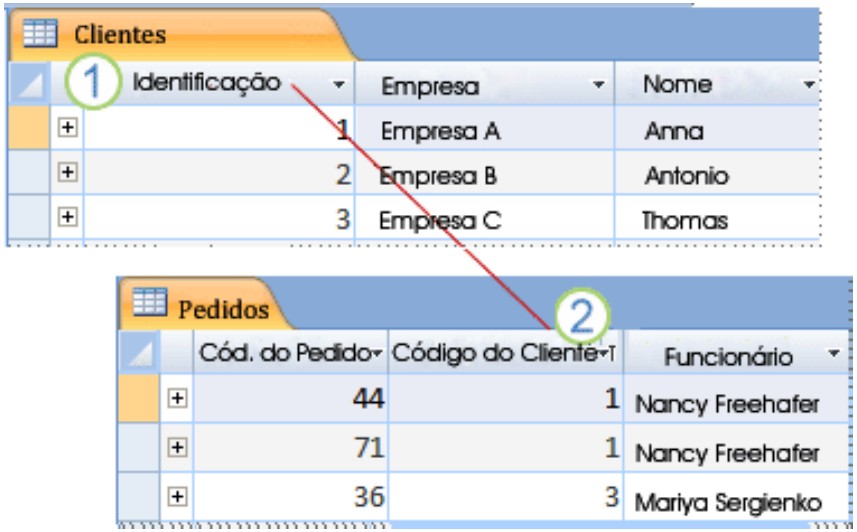
- É um método simples de se implementar
- Não necessita de hierarquização ou ordenação prévias
- Pode ser aplicado para dados em memória e em dispositivos auxiliares (HD, DVDs, etc.).
- As demais técnicas de busca tentam melhorar a eficiência do processo por meio da redução do conjunto de dados verificado:
  - baseadas em algum conhecimento sobre os dados e seu armazenamento
  - se pressupostos forem falhos invalidam o resultado da pesquisa.

FIAP

Busca Sequencial

Busca Sequencial

- Algoritmo básico que o arquivo seja varrido sequencialmente, até que:
  - seja encontrada a chave pesquisada (normalmente uma chave primária, ou seja, única no sistema)
  - Se chave não encontrada é atingido o final do arquivo.



Função de Busca Sequencial

```
public static int buscaSequencial(Registro baseDados[], int chaveproc) {
int pos = -1;
for (int i = 0; i< baseDados.length && pos == -1; i++)
       if((baseDados[i].getChave() == chaveproc))
           pos = i;

return pos;
}
```

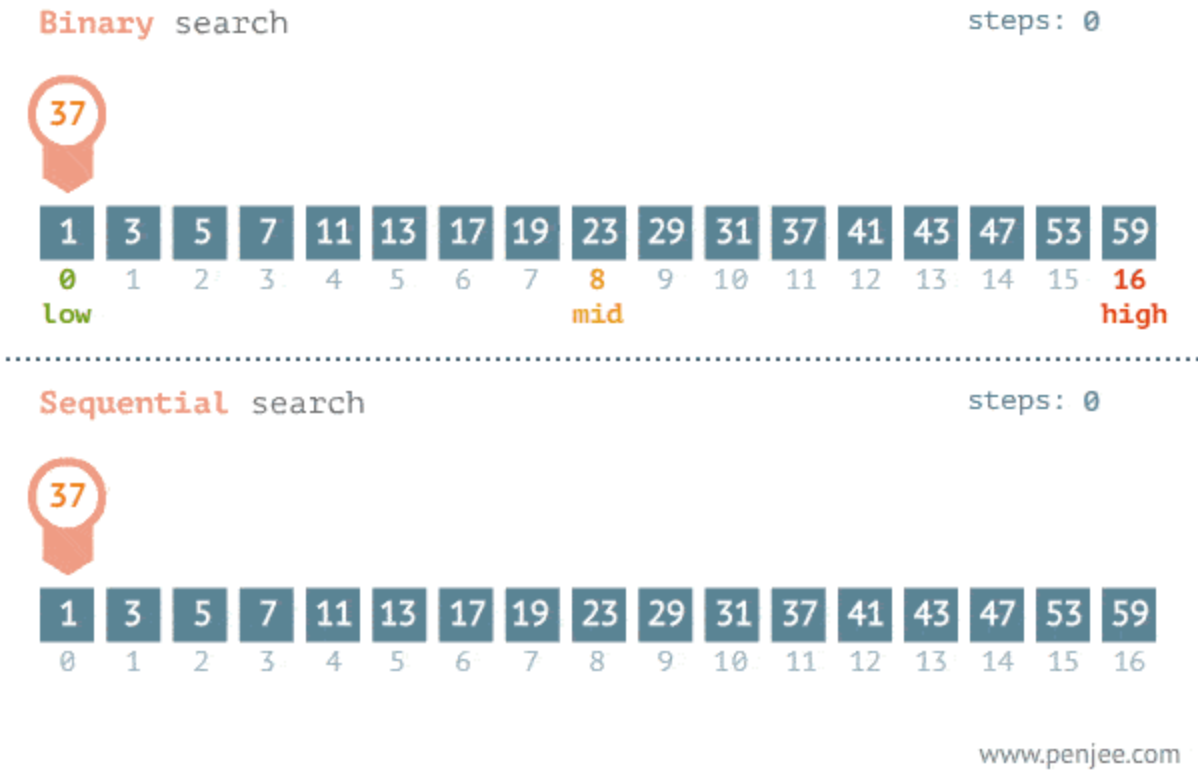
Projeto MétodosBusca pacote BuscaSequencial

Considerações sobre o Método

- Métodos simples de ser implementado, mas podem ser feitas melhorias em um sistema que faz muitas pesquisas (buscas):
  - Cada vez que um registro é procurado ele seria movido para o início do arquivo. Se o registro for novamente procurado então, ele seria mais rapidamente encontrado

Busca Binária

Idéia Geral do Método



Idéia Geral do Método

- É usado para arquivos ORDENADOS
- Compara-se a chave com o elemento central do conjunto.
  - Se forem iguais, o registro foi encontrado;
  - caso contrário, o processo será repetido na metade inferior ou na metade posterior, conforme o resultado da comparação;
  - Volta à comparação até que o registro seja encontrado ou se conclua pela sua inexistência.

Função para Busca Binária

```
public static int buscaBinaria(Registro baseDados[], int chaveproc) {
    System.out.println("Procurando o chave solicitada...");
    int i_baixo = 0;
    int i_medio = 0;
    int i_alto = baseDados.length -1;
    boolean achou = false;
    int posicao = -1;
    while( !achou && i_baixo <= i_alto) {
       i_medio = (i_baixo + i_alto)/2;
       if (chaveproc == baseDados[i_medio].getChave()) {
          posicao = i_medio;
          achou = true;
          }
    else {
        if (chaveproc < baseDados[i_medio].getChave())
            i_alto = i_medio - 1;
        else
            i_baixo = i_medio + 1;
         }
    }
    return posicao;
}
```

Projeto MétodosBusca pacote buscaBinaria

Considerações sobre o Método

- Métodos simples de ser implementado que necessita de ordenação do arquivo
- Tem execução muito mais eficiente que a busca sequencial.
  - Como saber quanto esse método é mais eficiente em relação ao outro?

Vamos fazer uma análise de desempenho.

Como? É o que vamos estudar a seguir...

Eficiência de um Algoritmo

- Como considerar se algoritmo é mais eficiente que um outro?
  - Tempo de execução: porém este deve variar com a máquina onde o programa está sendo executado.
  - Ocupação de espaço na memória: se esta é menor é possível utilizar em dispositivos com quantidade de memória reduzida.

Eficiência de um Algoritmo

- Portanto existes esses 2 aspectos de eficiência considerados: tempo requerido e espaço de armazenamento.
- Geralmente, o programador deverá otimizar um aspecto às custas do outro.

Como analisar a Eficiência de um Algoritmo

FIAP

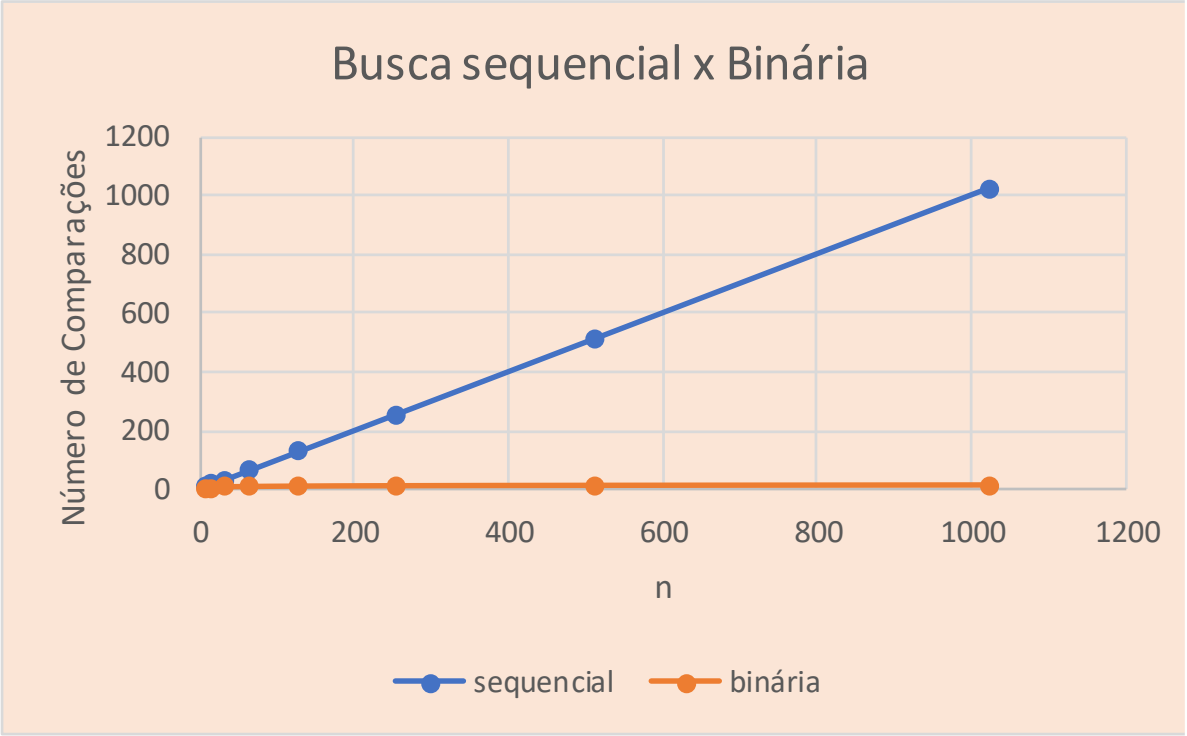
- A medida de tempo seria dado pelo **número de operações críticas** efetuadas durante a execução do algoritmo a fim de resolver um específico problema (por exemplo, a busca de um registro em um arquivo)
- O tamanho do arquivo no qual o algoritmo atua sempre é referenciado em relação à quantidade **n** de registros que o compõe.

Quais são as operações críticas?

- São as operações essenciais que deve ser realizadas para resolução do aplicação
- Exemplos de operações críticas:
  - Para algoritmos de busca seriam apenas as comparações sobre chaves
  - Para algoritmos de soma de vetores seriam apenas as adições sobre os elementos dos vetores
  - Para algoritmos de ordenação seriam as comparações sobre chaves e movimentação de registros (ou trocas)
  - Etc...

Executando o programa de **Comparação entre Métodos**

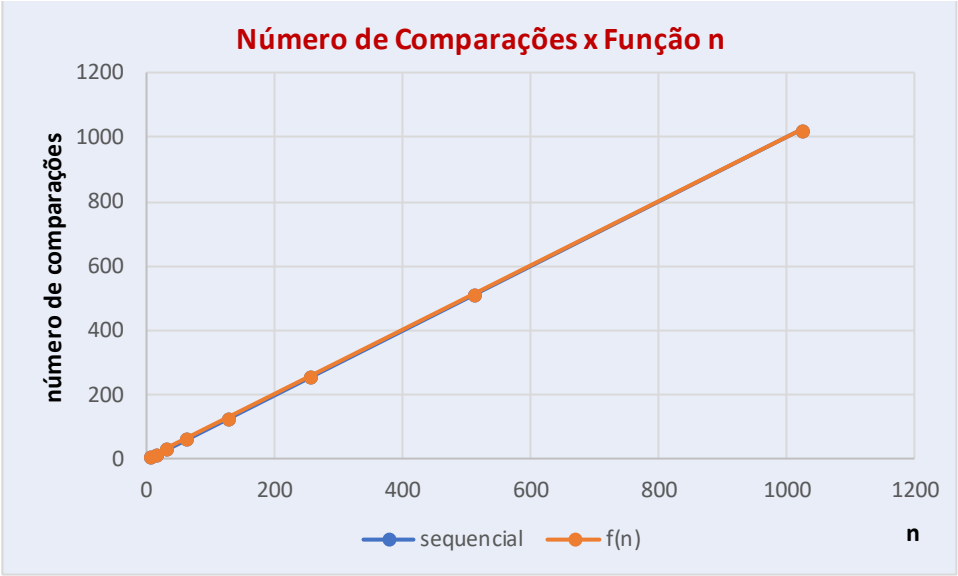
n	sequencial	binária
8	7	4
16	15	5
32	31	6
64	63	7
128	127	8
256	255	9
512	511	10
1024	1023	11



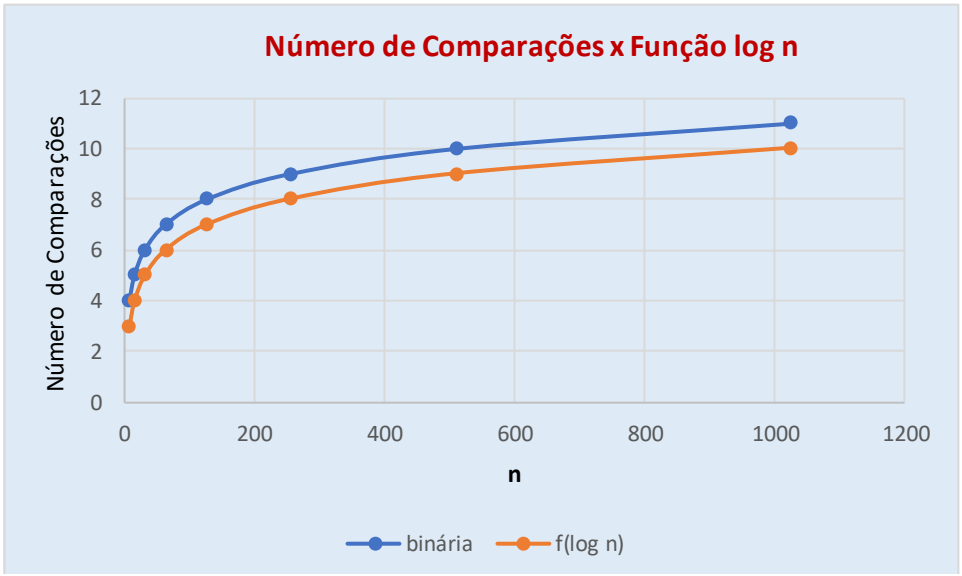
Projeto ComparacaoMetodosBusca

Como descrever o comportamento dos Métodos?

n	sequencial	f(n)
8	7	8
16	15	16
32	31	32
64	63	64
128	127	128
256	255	256
512	511	512
1024	1023	1024



n	binária	f(log n)
8	4	3
16	5	4
32	6	5
64	7	6
128	8	7
256	9	8
512	10	9
1024	11	10



Exercícios

Produto	Código	Fabricante	Preço (Un)
parafuso	623	ABC	R$ 0,75
prego	133	ABC	R$ 0,50
martelo	686	XYZ	R$ 34,75
alicate	461	W2M	R$ 22,50
soldador	201	ABC	R$ 48,99
tesoura	732	W2M	R$ 23,80

Supondo a tabela apresentada, responda:

1. Existe a chave primária? Se sim, qual?
2. Qual método entre os 3 estudados para realizar busca pode ser usado para:
   a) Pesquisar produto de um fabricante selecionado?
   b) Pesquisar produto selecionando um código?
3. Poderia ser usado o método de busca binária no estado em que se encontra a tabela? Explique sua decisão.
4. Crie um projeto JAVA para fazer o que é pedido nos exercícios 2.a, 2.b e 3.

REFERÊNCIAS

- TENENBAUM, A.M. E outros - **Estruturas de Dados usando C**. Makron Books do Brasil Editora Ltda, SP.
- PEREIRA, S. L. **Estrutura de Dados Fundamentais.** São Paulo: Érica.
- FORBELLONE, A.L.V. & EBERSPÄCHER, H.F. – **Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados.** Makron Books, São Paulo, SP
- ASCENCIO,A.F.G e ARAÚJO, G.S. **– Estruturas de Dados: Algoritmos, Análise da Complexidade e Implementação em JAVA e C/C++**