Estrutura de Dados 2

Aula 01 – Buscas: Operações em Vetores

Antonio Angelo de Souza Tartaglia

angelot@ifsp.edu.br

Estrutura de Dados 2

Busca e Ordenação

- No mundo da computação, talvez nenhuma outra tarefa seja mais **fundamental** ou tão **exaustivamente** analisada como estas de "**Busca e Ordenação**".

- Estas rotinas são utilizadas em praticamente todos os programas de Banco de Dados, bem como compiladores, interpretadores e sistemas operacionais

Estrutura de Dados 2

Busca

- Métodos de Busca:
  - Encontrar informações em um vetor desordenado requer uma Busca Linear, começando pelo primeiro elemento, e parando quando o elemento é encontrado ou quando o fim do vetor é alcançado.
  - Este método deve ser usado em dados desordenados mas também pode ser aplicado para dados ordenados.
  - Se os dados estiverem ordenados, então as Buscas “Ordenada” ou “Binária”, podem ser usadas, já que apresentam uma melhor performance em relação a Linear, estando a Busca Binária em um patamar de desempenho superior ao da Busca Ordenada.

Estrutura de Dados 2

Buscas em Vetores

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

- Busca – Definição:
  - Recuperação de dados armazenados em um repositório ou “Base de Dados”;

Vetores, Lista Ligada, Árvore, etc.

- O tipo de Busca depende do tipo de dados armazenados:
  - Dados estão estruturados (Vetor, Lista ou Árvore);
  - Dados ordenados, ou não ordenados;
  - Valores duplicados.

O tipo de Busca a ser realizada sempre depende do tipo de dado.

Uma Busca pode encontrar itens duplicados, então o que fazer? Recuperar o 1º, 2º, ambos?

Estrutura de Dados 2

Busca em vetores

- Tipos de busca abordados:
  - Dados armazenados em um Vetor;
  - Dados Ordenados ou não.
- Métodos:
  - Busca Linear;
  - Busca Ordenada;
  - Busca Binária.

Não necessitam de ordenação.

Trabalham com dados ordenados.

Estrutura de Dados 2

Busca em vetores

- Busca Linear: é a mais fácil de ser codificada. A função a seguir faz uma busca em um vetor de inteiros de comprimento conhecido "**n**", até encontrar o elemento procurado "**elem**":

```
int buscaLinear(int *vetor, int n, int elem){
    int i;
    for(i = 0; i < n; i++){
        if(elem == vetor[i]){
            return i;
        }
    }
    return -1;
}
```

- Esta função retorna o índice do elemento no vetor, caso ele seja encontrado, ou **-1** se o elemento não existir no vetor.
- É fácil notar que uma busca linear testará em média $^1/_2$**n** elementos. No melhor caso testará somente 1 elemento e no pior caso "**n**" elementos.

Estrutura de Dados 2

Busca em vetores

- Busca Linear:

```
int buscaLinear(int *vetor, int n, int elem){
    int i;
    for(i = 0; i < n; i++){
        if(elem == vetor[i]){
            return i;
        }
    }
    return -1;
}
```

Vetor:

120	150	110	130	100	160	140	190	170	180

elem: 100 Elemento à ser procurado

i = 0	120	150	110	130	100	160	140	190	170	180	Diferente: continua a busca...
i = 1	120	150	110	130	100	160	140	190	170	180	Diferente: continua a busca...
i = 2	120	150	110	130	100	160	140	190	170	180	Diferente: continua a busca...
i = 3	120	150	110	130	100	160	140	190	170	180	Diferente: continua a busca...
i = 4	120	150	110	130	100	160	140	190	170	180	Igual: Busca terminada.

Retorna índice 4

Estrutura de Dados 2

Busca em vetores

- Busca Ordenada: igualmente fácil de ser implementada como a Busca Linear, apenas devemos acrescentar uma verificação a cada iteração com a finalidade de detectar, a cada passagem, se o elemento verificado no vetor é maior do que o elemento que se procura "**elem**".

```
int buscaOrdenada(int *vetor, int n, int elem){
    int i;
    for(i = 0; i < n; i++){
        if(elem == vetor[i]){
            return i;
        }else{
            if(elem < vetor[i]){
                return -1;
            }
        }
    }
    return -1;
}
```

- A partir do momento que o elemento no vetor for maior que o elemento que se procura, a Busca é encerrada pois o elemento não existe no vetor, nesse caso a função retorna -1.

Estrutura de Dados 2

Busca em vetores

- Busca Ordenada:

```
int buscaOrdenada(int *vetor, int n, int elem){
    int i;
    for(i = 0; i < n; i++){
        if(elem == vetor[i]){
            return i;
        }else{
            if(elem < vetor[i]){
                return -1;
            }
        }
    }
    return -1;
}
```

A tarefa de busca é otimizada se o vetor estiver ordenado. Ainda assim não é muito eficiente, pois é necessário percorrer todo o vetor em alguns casos.

Vetor:	100	110	120	130	140	150	160	170	180	190

elem: 125 Elemento à ser procurado

i = 0	100	110	120	130	140	150	160	170	180	190	Diferente: continua a busca...
i = 1	100	110	120	130	140	150	160	170	180	190	Diferente: continua a busca...
i = 2	100	110	120	130	140	150	160	170	180	190	Diferente: continua a busca...
i = 3	100	110	120	130	140	150	160	170	180	190	Valor é maior: elem não existe.
i = 4	100	110	120	130	140	150	160				

Retorna -1, não existe o elemento procurado.

Busca encerrada. Daqui p/ frente só maiores.

Estrutura de Dados 2

Busca em vetores

- Busca Binária: Este método é superior aos dois anteriores. Utiliza “Divisão e Conferência”.
  - A estratégia é baseada na ideia de dividir para conquistar. A cada passo, esse algoritmo analisa o valor do meio do vetor. Caso o valor seja igual ao elemento procurado, a busca termina. Do contrário, baseado na comparação anterior (elemento procurado é maior ou menor), a busca continua na metade do vetor em que o elemento pode se encontrar .
  - Este procedimento é repetido até que o elemento procurado seja encontrado ou até que não haja mais elementos a testar, retornando assim **–1**

Estrutura de Dados 2

Busca em vetores

- Busca Binária:

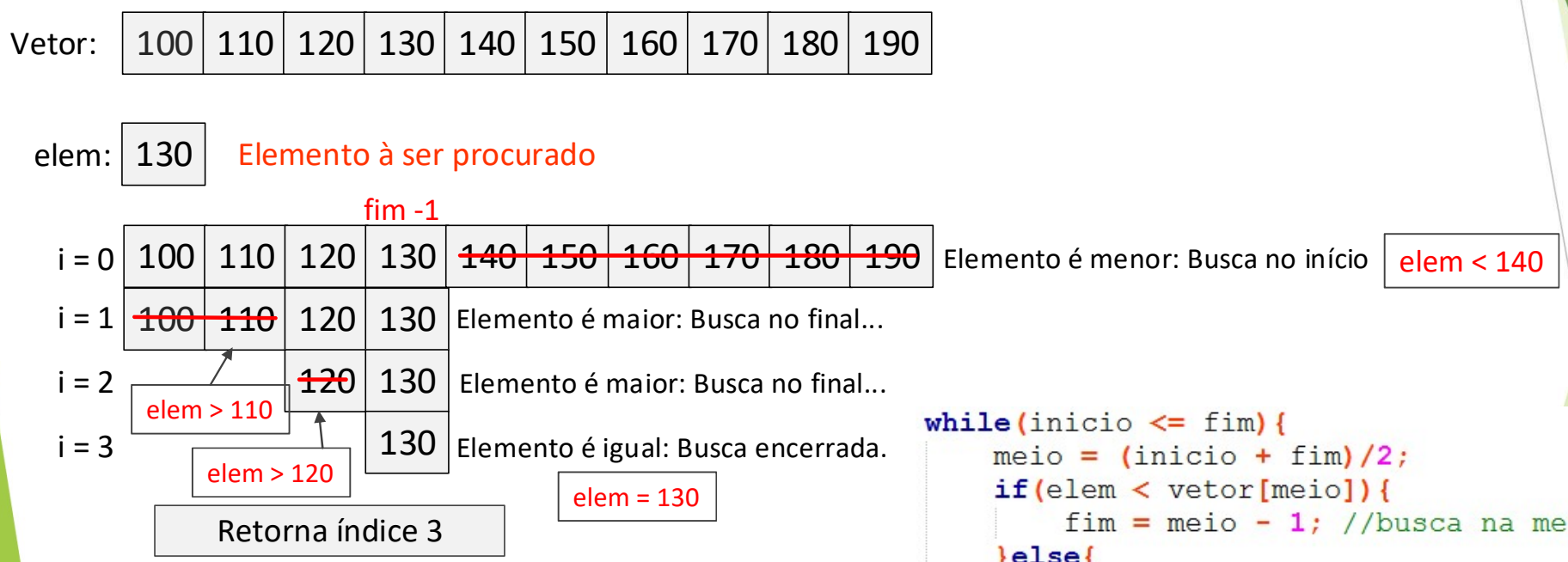
```
int buscaBinaria(int *vetor, int n, int elem){
    int i, inicio, meio, fim;
    inicio = 0;
    fim = n - 1;

    while(inicio <= fim){
        meio = (inicio + fim)/2;
        if(elem < vetor[meio]){
            fim = meio - 1; //busca na metade esquerda
        }else{
            if(elem > vetor[meio]){
                inicio = meio + 1; // busca na metade direita
            }else{
                return meio;
            }
        }
    }
    return -1; // elemento não encontrado
}
```

- Neste tipo de Busca, o número de comparações no pior caso, é $\log_2 n$.

Estrutura de Dados 2

Busca em vetores

- Busca Binária:




```
while(inicio <= fim){
    meio = (inicio + fim)/2;
    if(elem < vetor[meio]){
        fim = meio - 1; //busca na metade esquerda
    }else{
        if(elem > vetor[meio]){
            inicio = meio + 1; // busca na metade direita
        }else{
            return meio;
        }
    }
}
```

Estrutura de Dados 2

Busca em vetores

- É importante lembrar que toda busca é feita utilizando como base uma chave específica. Esta chave é o “**campo**” utilizado para comparação.

- No caso de uma estrutura, a chave é o campo da **struct** utilizado para comparação e deve ser único.

Estrutura de Dados 2

Atividade 1

- Elabore um programa que execute os três tipos de busca:
  - Busca Linear;
  - Busca Ordenada;
  - Busca Binária.
- Para isso, seu programa deverá contar com dois vetores de inteiros, um desordenado para a Busca Linear, e outro ordenado para as Buscas Ordenada e Binária.
- Os resultados devem ser apresentados em tela juntamente com o respectivo vetor para a conferência do índice retornado.
- Entregue no Moodle como Atividade 1 – Buscas em Vetores

Estrutura de Dados 2

Atividade 2

- Utilizando o programa da atividade 1, escreva uma função para a inserção de um novo valor no vetor ordenado em sua posição correta. Desloque os outros números, se necessário. Considere que há espaço vago no vetor.

- Entregue no Moodle como atividade 2.

Estrutura de Dados 2

Busca em Vetor de Estruturas – struct

```c
struct funcionario{
    int codigo;
    char nome[30];
    float salario;
};

struct funcionario func[6];
```

```c
int buscaLinearCodigo(struct funcionario *vetor, int n, int cod){
    int i;
    for(i = 0; i < n; i++){
        if(cod == vetor[i].codigo){
            return i;//elemento encontrado
        }
    }
    return -1;//elemento não encontrado
}
```

Na prática trabalhamos com dados mais complexos, estruturas com mais informações

Codigo; Nome[30]; Salario;	Codigo; Nome[30]; Salario;	Codigo; Nome[30]; Salario;	Codigo; Nome[30]; Salario;	Codigo; Nome[30]; Salario;	Codigo; Nome[30]; Salario;
Func[0]	Func[1]	Func[2]	Func[3]	Func[4]	Func[5]

- Em um vetor de estruturas não basta mais comparar a posição do vetor, é necessário comparar alguns campos da estrutura que está em cada posição dentro do vetor.

Estrutura de Dados 2

Busca em Vetor de Estruturas – struct

- Busca por Código:

```
int buscaLinearCodigo(struct funcionario *vetor, int n, int cod){
    int i;
    for(i = 0; i < n; i++){
        if(cod == vetor[i].codigo){
            return i;//elemento encontrado
        }
    }
    return -1;//elemento não encontrado
}
```

```
struct funcionario{
    int codigo;
    char nome[30];
    float salario;
    char departamento[20];
};
```

- Busca por outro campo qualquer (string):

```
int buscaLinearNome(struct funcionario *vetor, int n, char *nome){
    int i;
    for(i = 0; i < n; i++){
        if(strcmp(nome, vetor[i].nome) == 0){
            return i;//elemento encontrado
        }
    }
    return -1;//elemento não encontrado
}
```

Estrutura de Dados 2

Atividade 3

- Idem aos requisitos da Atividade 1, porém agora utilizando vetores de estruturas apresentadas no exemplo anterior.

- Entregue no Moodle como atividade 3

Estrutura de Dados 2

Atividade 4

- Baseando-se na atividade 1, descarte o vetor ordenado e implemente agora em seu programa uma função para ordenar o vetor desordenado que você criou, tornando assim suas buscas mais eficientes.
- Baseie-se nos conceitos apresentados na sala de aula para esta tarefa.
- Importante!!!

Não usar a internet para buscar o algoritmo pronto!!!

- Entregue no Moodle como atividade 4. Esta tarefa terá peso diferenciado na totalização dos pontos de exercícios, e somente terá validade para o dia e período em sala de aula.