Algoritmos e Estruturas de Dados II

2º Período Engenharia da Computação

Prof. Edwaldo Soares Rodrigues
Email: edwaldo.rodrigues@uemg.br

UNIVERSIDADE
DO ESTADO DE MINAS GERAIS | UEMG
UNIDADE DIVINÓPOLIS

Listas

Listas

- Definição:
  - Listas são estruturas flexíveis que admitem as operações de inserção, remoção e recuperação de itens;
  - É uma sequencia de zero ou mais itens x_1; x_2; : ... ; x_n na qual x_i é de um determinado tipo e n representa o tamanho da lista;
  - Sua principal propriedade estrutural diz respeito as posições relativas dos itens;
    - Se n >= 1, x_1 é o primeiro item e x_n é o último;
    - Em geral, x_i precede x_i+1 para i = 1; 2; ... ; n - 1 e x_i sucede x_i-1 para i = 2; 3; ... ; n;

Listas

- Aplicações:
  - Diversos tipos de aplicações requerem uma lista:
    - Lista telefônica;
    - Lista de tarefas;
    - Gerência de memória;
    - Simulação;
    - Compiladores;
    - Entre outras;

Listas

- TAD Lista:
  - Vamos definir um TAD com as principais operações sobre uma lista:
    - Para simplificar vamos definir apenas as operações principais, posteriormente, outras operações podem ser definidas;

Listas

- Principais operações:
  - Criar lista;
  - Limpar lista;
  - Inserir item (última posição);
  - Inserir item (por posição);
  - Remover item (última posição);
  - Recuperar item (dado uma chave);
  - Recuperar item (por posição);
  - Contar número de itens;
  - Verificar se a lista está vazia;
  - Verificar se a lista está cheia;
  - Imprime lista;

Listas

- Criar Lista:
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** inicia a estrutura de dados;
- Limpar Lista:
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** coloca a estrutura de dados no estado inicial;

Listas

- Inserir item (última posição):
  - **Pré-condição:** a lista não está cheia;
  - **Pós-condição:** insere um item na última posição, retorna true se a operação foi executada com sucesso, false caso contrário;
- Inserir item (por posição):
  - **Pré-condição:** a lista não está cheia;
  - **Pós-condição:** insere um item na posição informada, que deve estar dentro da lista, retorna true se a operação foi executada com sucesso, false caso contrário;

Listas

- Remover item (por posição):
  - **Pré-condição:** uma posição válida da lista é informada;
  - **Pós-condição:** o item na posição fornecida é removido da lista, retorna true se a operação foi executada com sucesso, false caso contrário;
- Recuperar item (dada uma chave):
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** recupera o item dado uma chave, retorna true se a operação foi executada com sucesso, false caso contrário;

Listas

- Recuperar item (por posição):
  - **Pré-condição:** uma posição válida da lista é informada;
  - **Pós-condição:** recupera o item na posição fornecida, retorna true se a operação foi executada com sucesso, false caso contrário;
- Contar número de itens:
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** retorna o número de itens na lista;

Listas

- Verificar se a lista está vazia:
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** retorna true se a lista estiver vazia e false caso-contrário;
- Verificar se a lista está cheia:
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** retorna true se a lista estiver cheia e false caso-contrário;

Listas

- Imprime lista:
  - **Pré-condição:** nenhuma;
  - **Pós-condição:** imprime na tela os itens da lista;

Listas

- Como implementar o TAD Lista?
  - Basicamente existem duas formas:
    - Estática (utilizando vetores);
    - Dinâmica (utilizando listas ligadas);

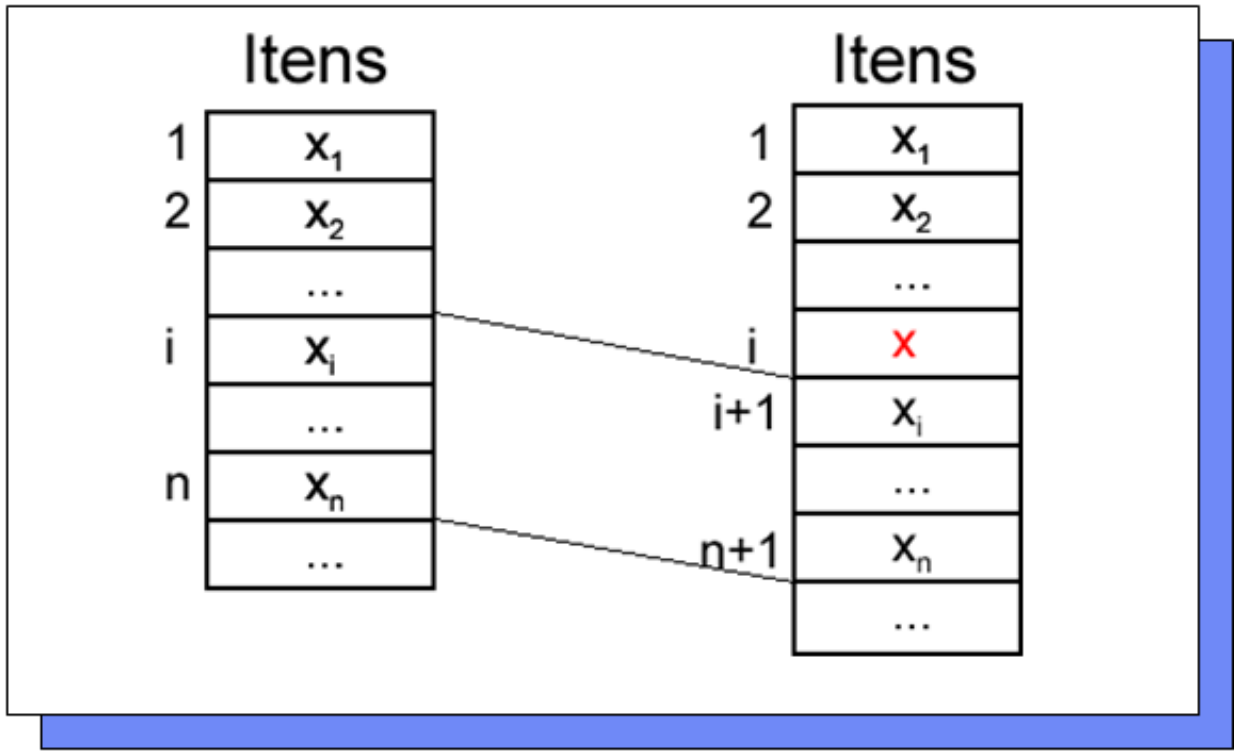
Lista Estática

Lista Estática

- Características:
  - Os itens da lista são armazenados em posições contíguas de memória;
  - A lista pode ser percorrida em qualquer direção;
  - A inserção de um novo item pode ser realizada após o último item com custo constante;

Lista Estática

- Inserção de um item na posição i:
  - Problema do deslocamento de itens;



Lista Estática

- Tad Lista Estática:

```
typedef struct lista{
    int vetor[TAM];
    int fim;
}Lista;
```

Lista Estática Ordenada

Lista Estática Ordenada

- Características:
  - Uma lista pode ser mantida em ordem crescente/decrescente segundo o valor de alguma chave;
  - Essa ordem facilita a pesquisa de itens;
  - Por outro lado, a inserção é mais complexa pois deve manter os itens ordenados;

Lista Estática Ordenada

- TAD Lista Ordenada:
  - O TAD listas ordenadas é o mesmo do TAD listas, apenas difere na implementação;
  - As operações diferentes serão a inserção e a exclusão por posição de itens;

Lista Estática Ordenada

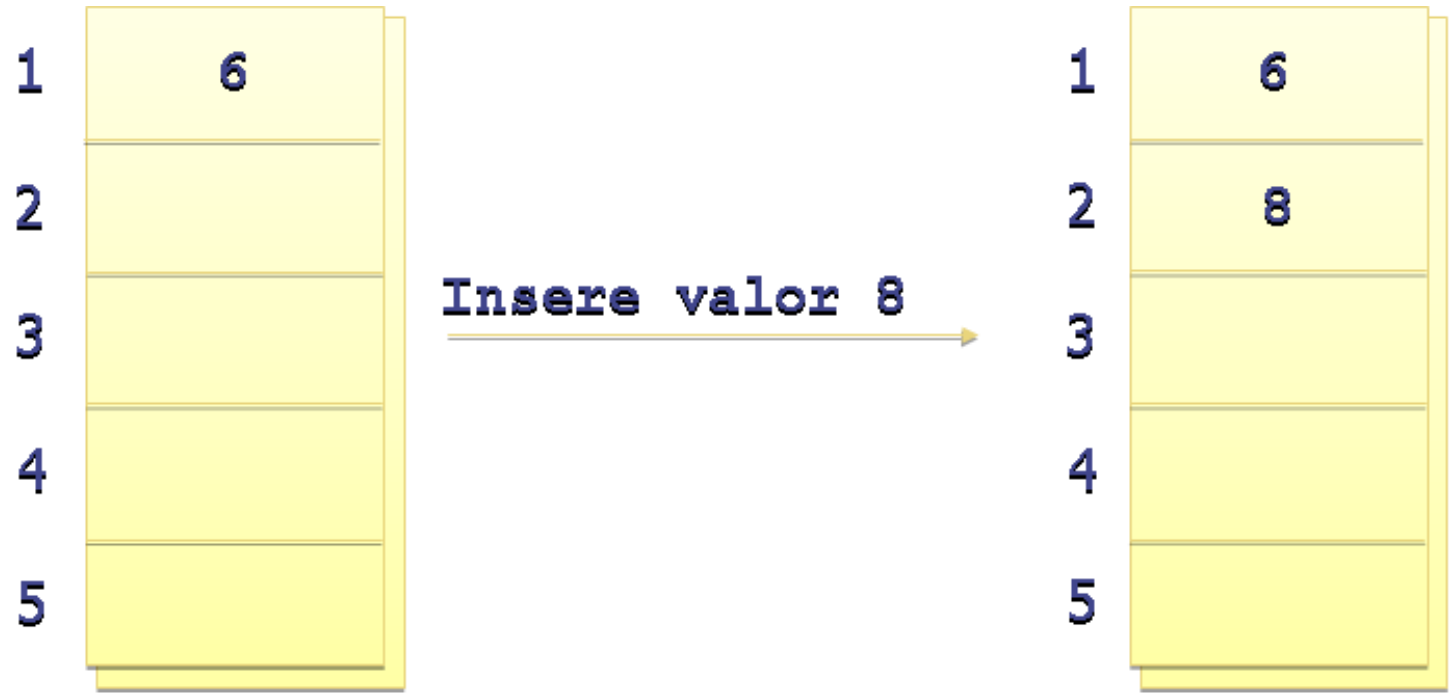
- Inserir item:
  - **Pré-condição:** a lista não está cheia;
  - **Pós-condição:** insere um item em uma posição tal que a lista é mantida ordenada;
- Remover item (por posição):
  - **Pré-condição:** uma posição válida da lista é informada;
  - **Pós-condição:** o item na posição fornecida é removido da lista, a lista é mantida ordenada;

Lista Estática Ordenada

- Exemplo de inserção ordenada:

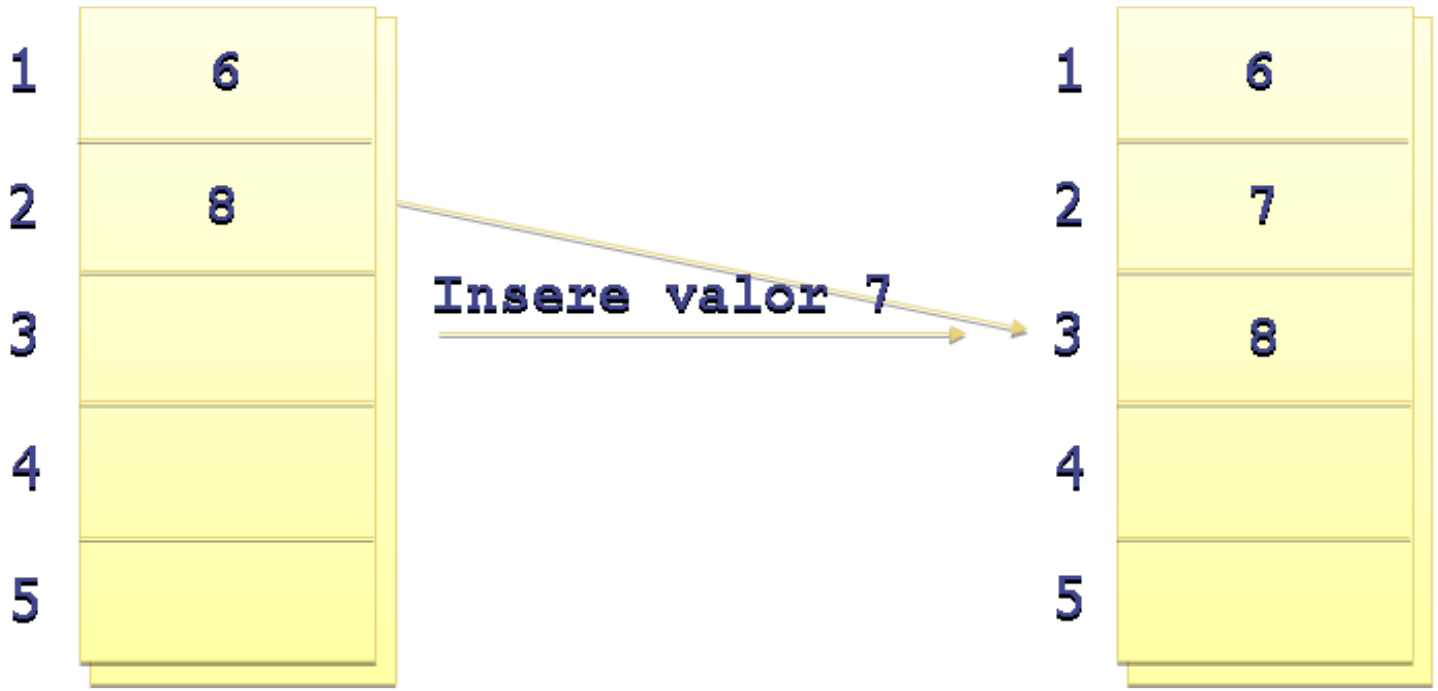
Lista Estática Ordenada

- Exemplo de inserção ordenada:



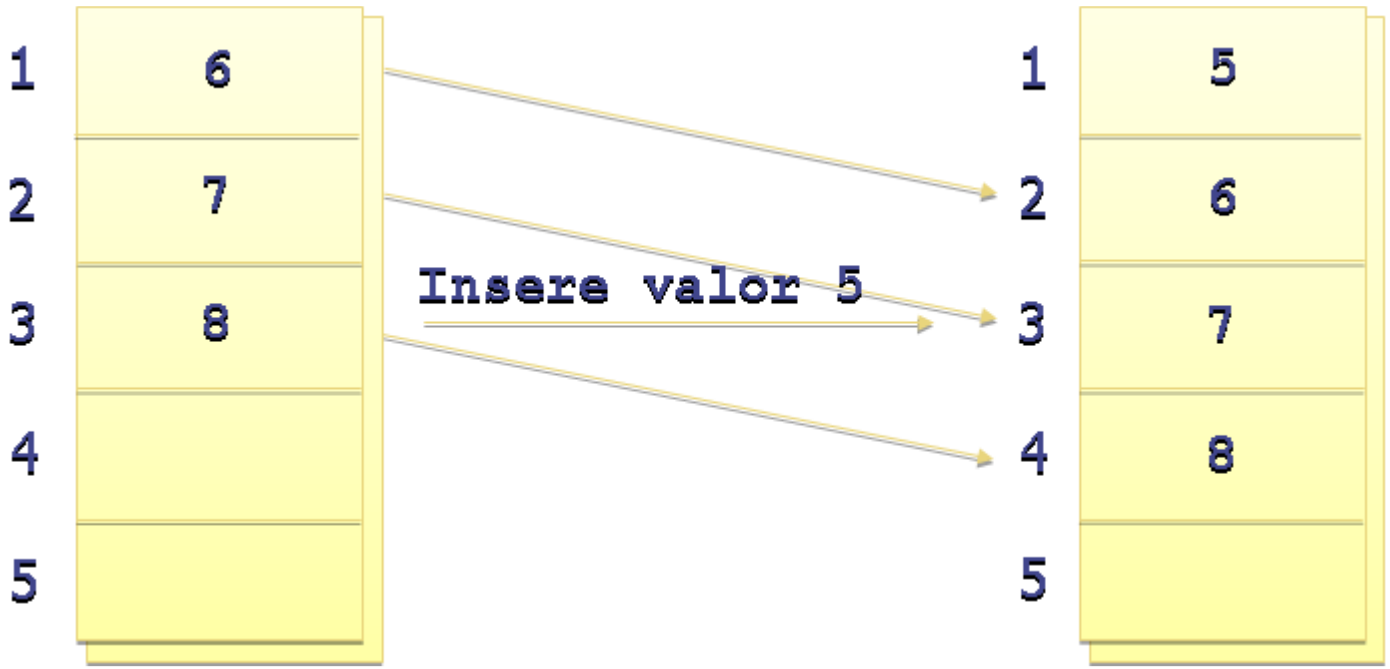
Lista Estática Ordenada

- Exemplo de inserção ordenada:



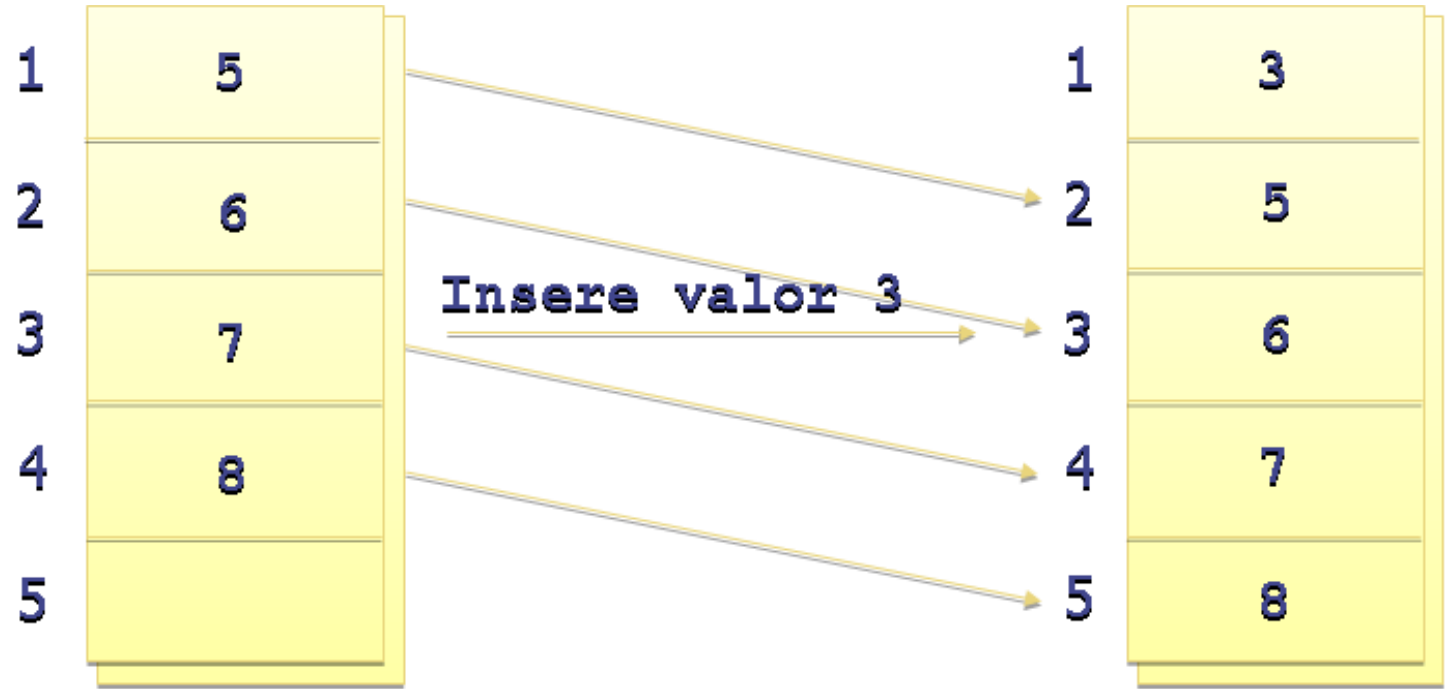
Lista Estática Ordenada

- Exemplo de inserção ordenada:



Lista Estática Ordenada

- Exemplo de inserção ordenada:



Busca em Listas Estáticas

Busca em Listas Estáticas

- Uma tarefa comum a ser executada sobre listas é a busca de itens dada uma chave;
- No caso da lista não ordenada, a busca será sequencial (consultar todos os elementos);
- Porém, com uma lista ordenada, diferentes estratégias podem ser aplicadas que aceleram essa busca:
  - Busca sequencial “otimizada”;
  - Busca binária;

Busca em Listas Estáticas

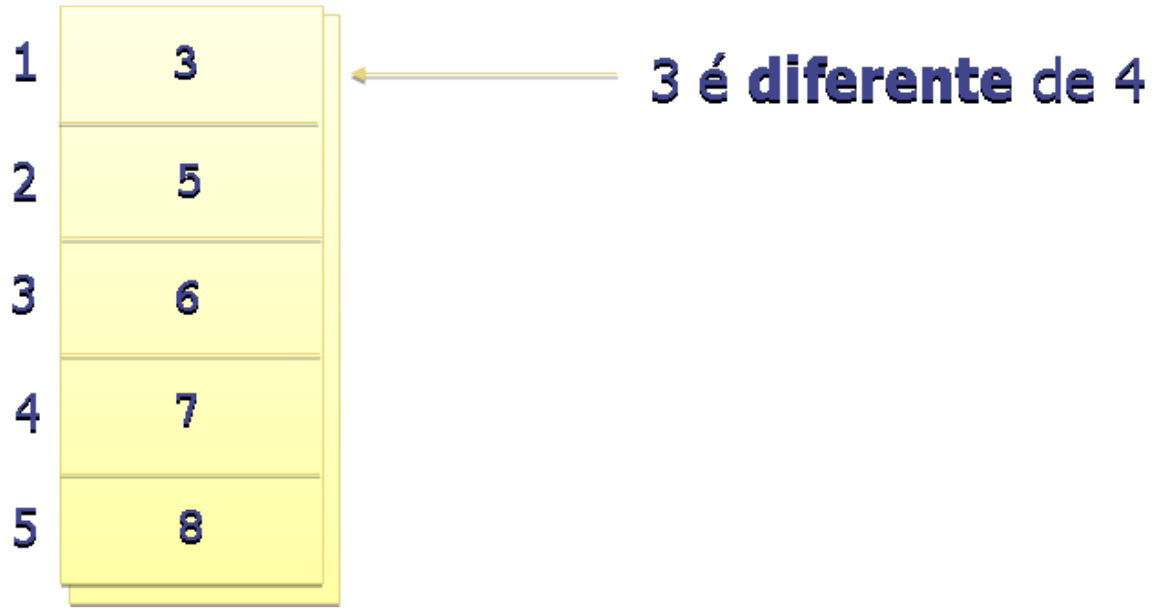
- Busca Sequencial:
  - Na busca sequencial, a ideia é procurar um elemento que tenha uma determinada chave, começando do início da lista, e parar quando a lista terminar ou quando o elemento for encontrado;

Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Quando os elementos estão ordenados, a busca sequencial pode ser "otimizada" (acelerada);
  - Vejamos o seguinte exemplo:

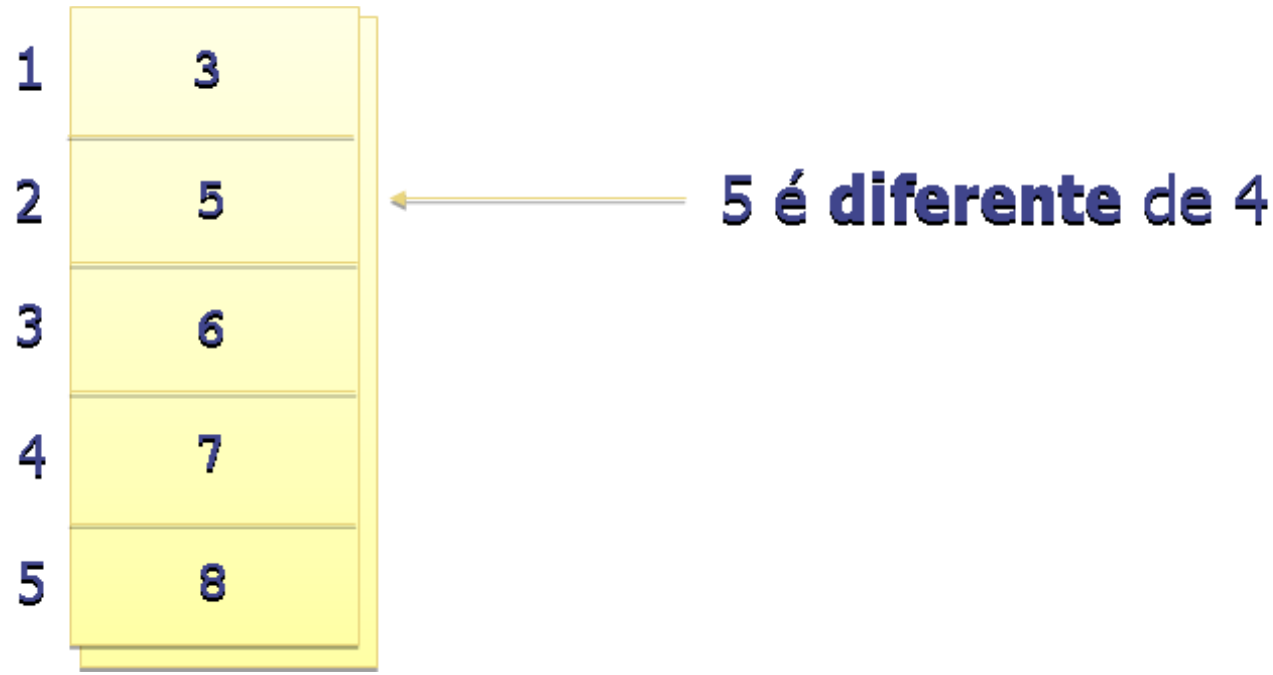
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 1: procurar pelo valor 4:



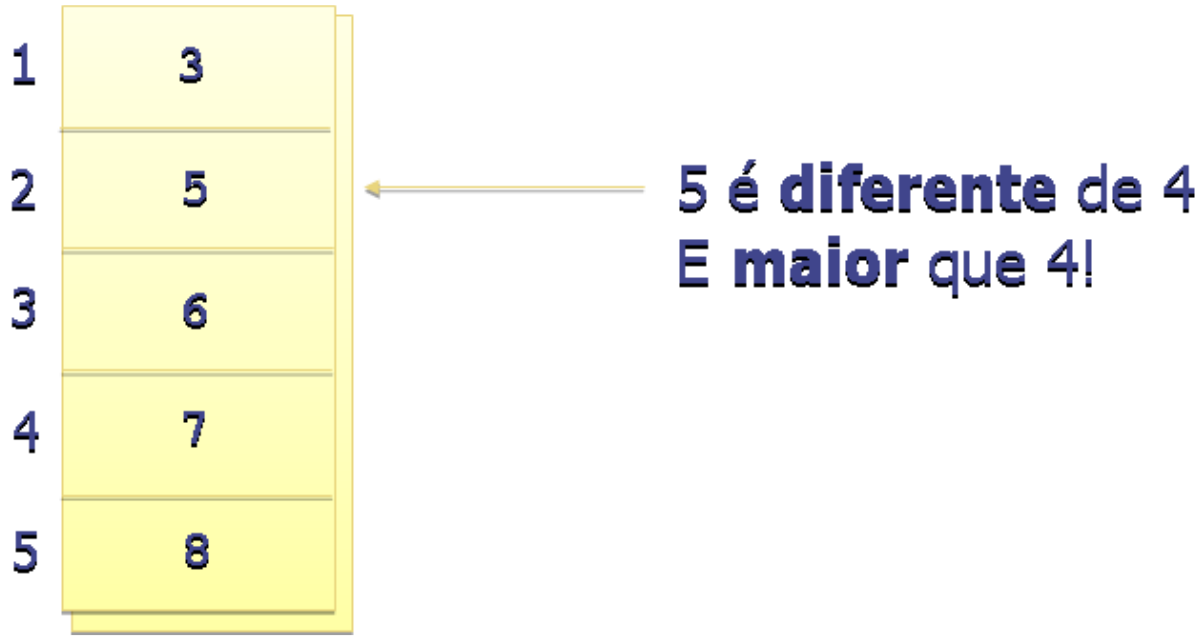
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 1: procurar pelo valor 4:



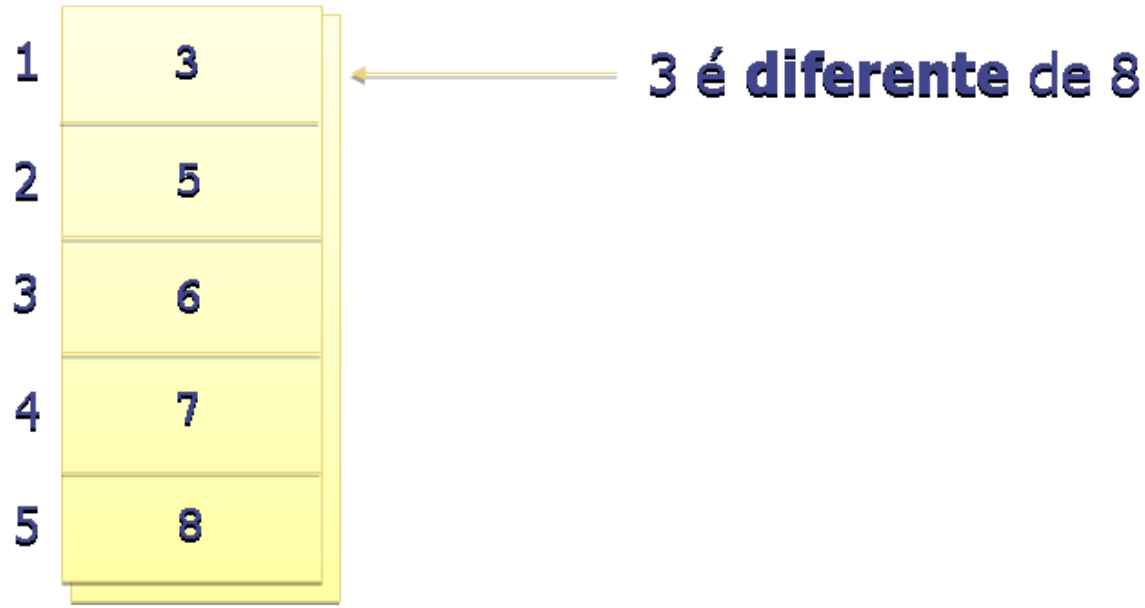
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 1: procurar pelo valor 4:



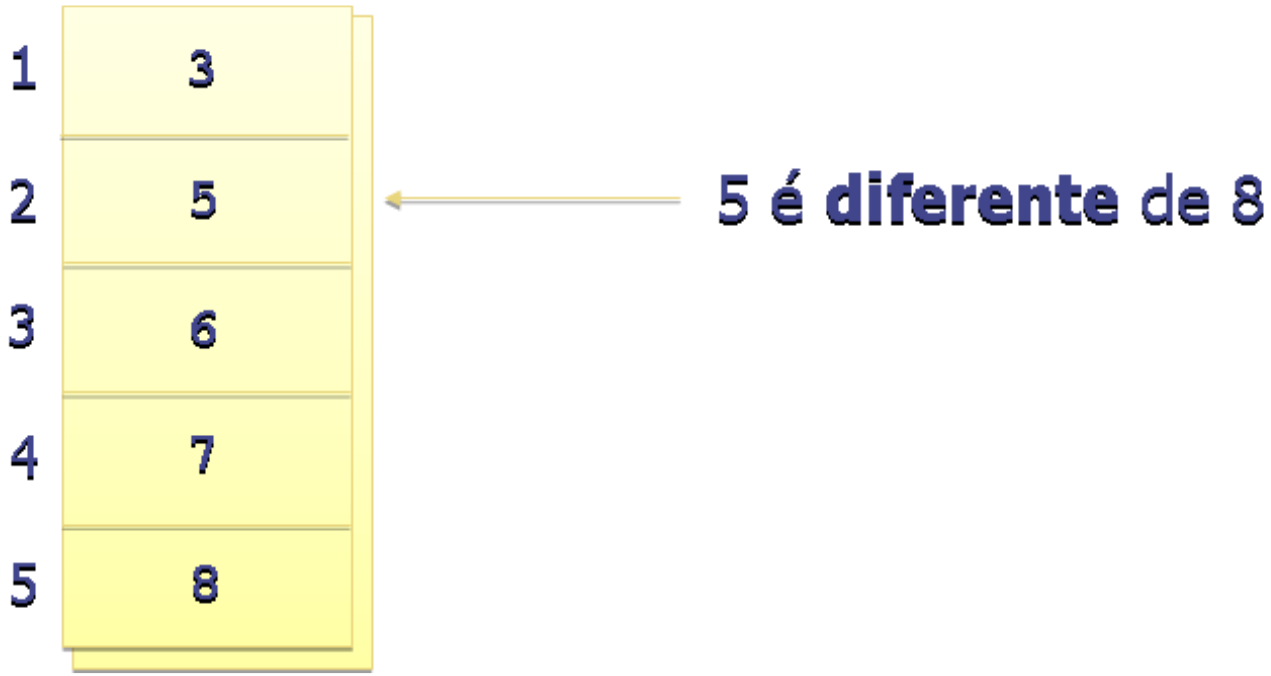
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 2: procurar pelo valor 8:



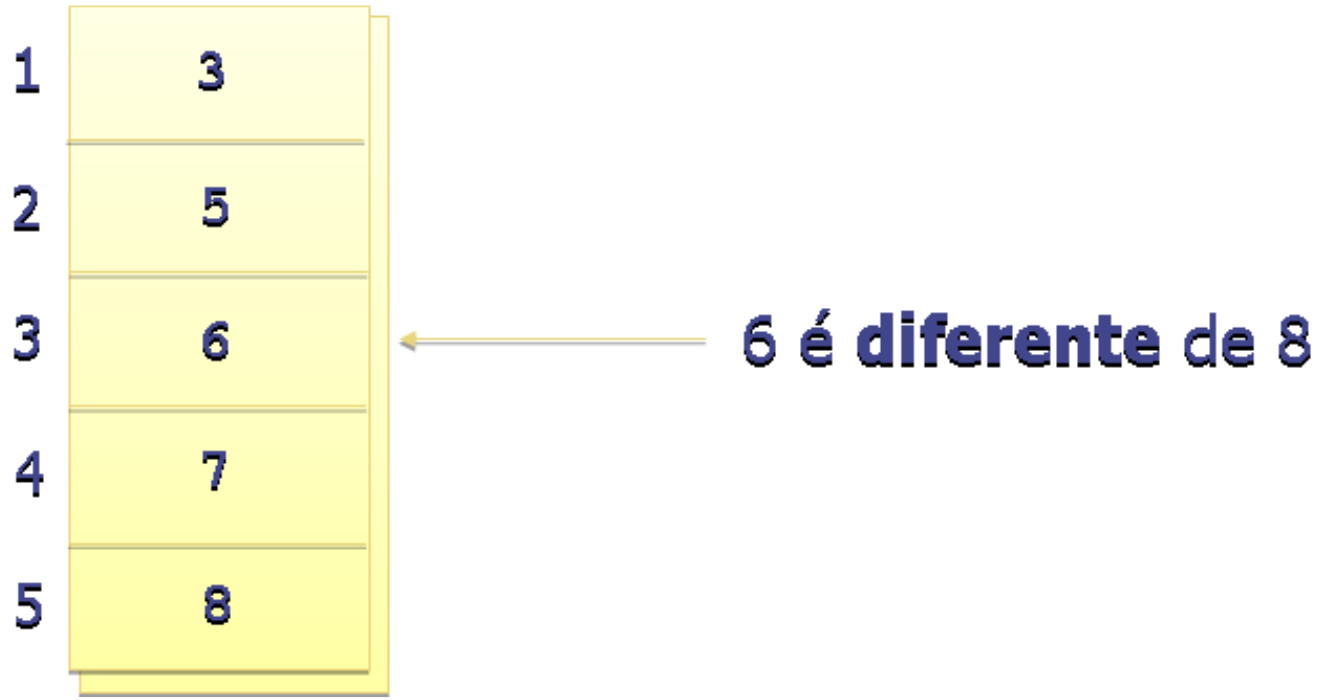
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 2: procurar pelo valor 8:



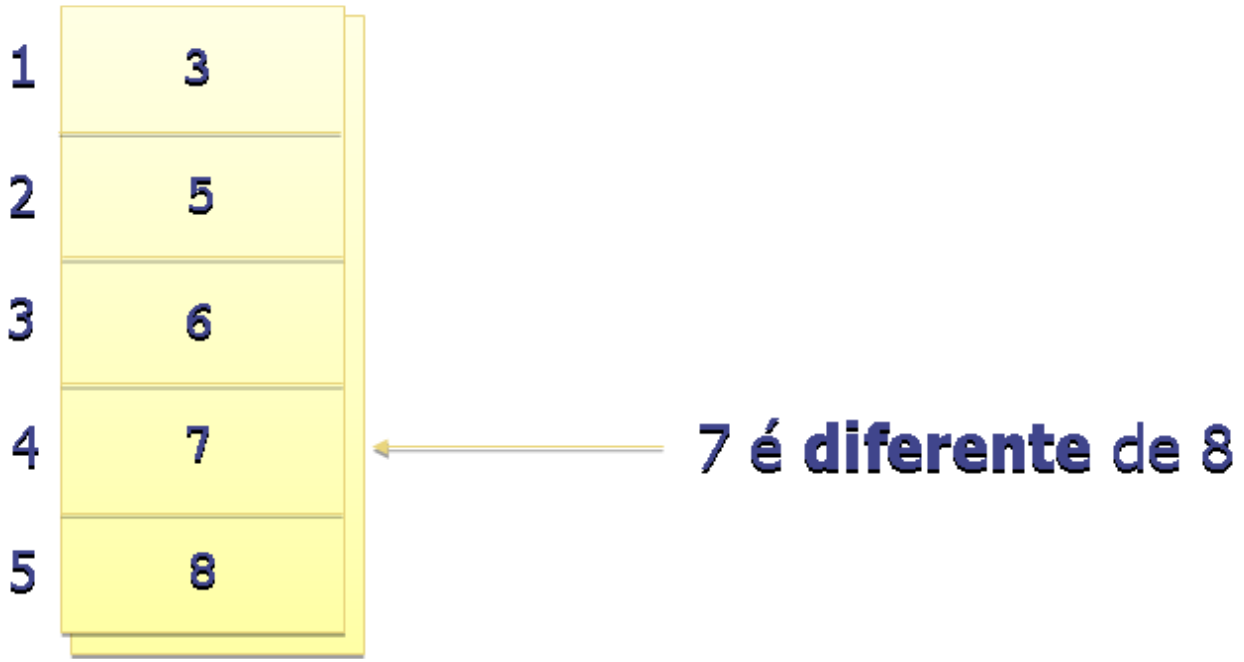
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 2: procurar pelo valor 8:



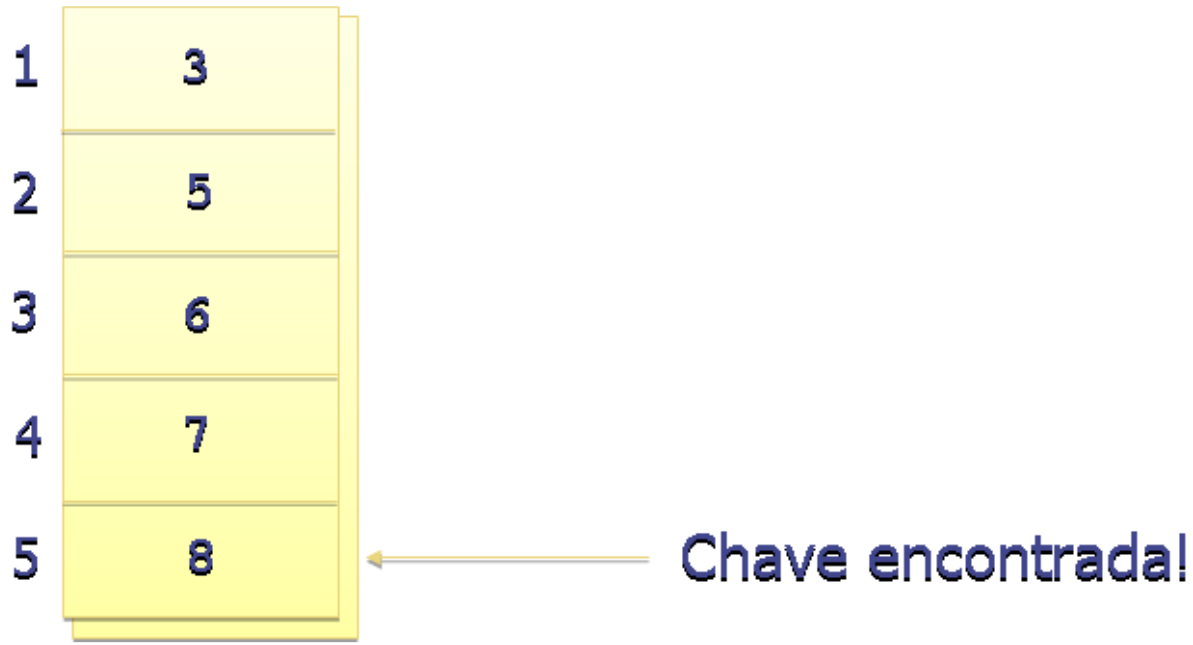
Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 2: procurar pelo valor 8:



Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - Exemplo 2: procurar pelo valor 8:



Busca em Listas Estáticas

- Busca Sequencial (Ordenada):
  - A lista ordenada permite realizar buscas sequenciais mais rápidas uma vez que, caso a chave procurada não exista, pode-se parar a busca tão logo se encontre um elemento com chave maior que a procurada
  - Entretanto, essa melhoria não altera a complexidade da busca sequencial, que ainda é O(n). Porque?

Busca em Listas Estáticas

- Busca Binária:
  - A busca binária é um algoritmo de busca mais sofisticado e bem mais eficiente que a busca sequencial;
  - Entretanto, a busca binária somente pode ser aplicada em estruturas que permitem acessar cada elemento em tempo constante, tais como os vetores;
  - A ideia é, a cada iteração, dividir o vetor ao meio e descartar metade do vetor;

Conclusão

Listas

- Pontos fortes:
  - Tempo constante de acesso aos dados;
- Pontos fracos:
  - Custo para inserir e retirar elementos da lista, dada uma posição fornecida pelo usuário;
  - Tamanho máximo da lista é (dependendo da linguagem) definido em tempo de compilação;

Listas

- Quanto utilizar:
  - Essa implementação simples é mais comumente utilizadas em certas situações:
    - Listas pequenas;
    - Tamanho máximo da lista é conhecido;
    - Poucas ocorrências de utilização dos métodos de inserção e remoção, dada uma posição definida pelo usuário;

Algoritmos e Estruturas de Dados II

- Bibliografia:
  - Básica:
    - CORMEN, Thomas, RIVEST, Ronald, STEIN, Clifford, LEISERSON, Charles. Algoritmos. Rio de Janeiro: Elsevier, 2002.
    - EDELWEISS, Nina, GALANTE, Renata. Estruturas de dados. Porto Alegre: Bookman. 2009. (Série livros didáticos informática UFRGS,18).
    - ZIVIANI, Nívio. Projeto de algoritmos com implementação em Pascal e C. São Paulo: Cengage Learning, 2010.
  - Complementar:
    - ASCENCIO, Ana C. G. Estrutura de dados. São Paulo: Pearson, 2011. ISBN: 9788576058816.
    - PINTO, W.S. Introdução ao desenvolvimento de algoritmos e estrutura de dados. São Paulo: Érica, 1990.
    - PREISS, Bruno. Estruturas de dados e algoritmos. Rio de Janeiro: Campus, 2000.
    - TENEMBAUM. Aaron M. Estruturas de dados usando C. São Paulo: Makron Books. 1995. 884 p. ISBN: 8534603480.
    - VELOSO, Paulo A. S. Complexidade de algoritmos: análise, projeto e métodos. Porto Alegre, RS: Sagra Luzzatto, 2001

Algoritmos e Estruturas de Dados II