SIN 211 - Algoritmos e Estruturas de Dados
(Lista Ordenada)

Profº: Joelson Antônio dos Santos

Universidade Federal de Viçosa
Instituto de Ciências Exatas e Tecnológicas
Campus de Rio Paranaíba - MG

joelsonn.santos@gmail.com
Sala: BBT 233

10 de abril de 2018

Aula de Hoje

1. Lista Encadeada Ordenada
   - Tipos de Ordenação

Créditos

O material desta aula é composto por adaptações e extensões dos originais gentilmente cedidos pelos professores **Moacir Pereira Ponti** e **Rachel Reis**.

Lista Encadeada

- Como é feita a busca na lista encadeada?
- Quais os critérios de parada para a busca na lista encadeada?
- Esse tipo de busca pode ser um problema?

Lista Encadeada

- Formas de organização de listas encadeadas.
  - Lista auto organizada.
  - Lista ordenada.

Motivação

- É possível melhorar a eficiência de listas encadeadas.
- Métodos de auto-organização.
  - Mover para frente.
  - Transposição.
  - Contagem.
- Método de ordenação.

Listas Auto Organizadas

- Ordenam os elementos em função das pesquisas realizadas durante o uso da aplicação.
- Estes métodos buscam colocar os elementos mais prováveis de serem acessados no início da lista.

Listas Auto Organizadas

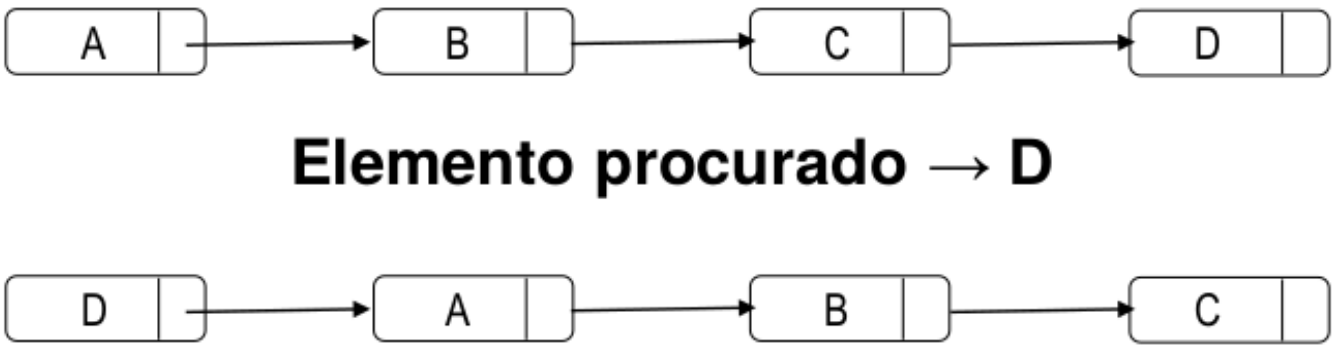
- **Método de mover para frente**

Listas Auto Organizadas

- **Método de mover para frente**
  - O elemento pesquisado vai para o início da lista.

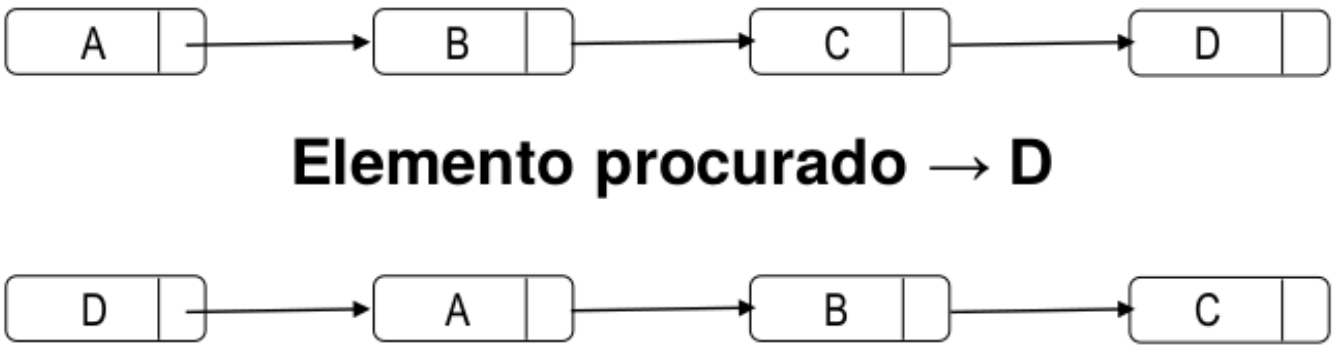
Listas Auto Organizadas

- **Método de mover para frente**
  - O elemento pesquisado vai para o início da lista.



Listas Auto Organizadas

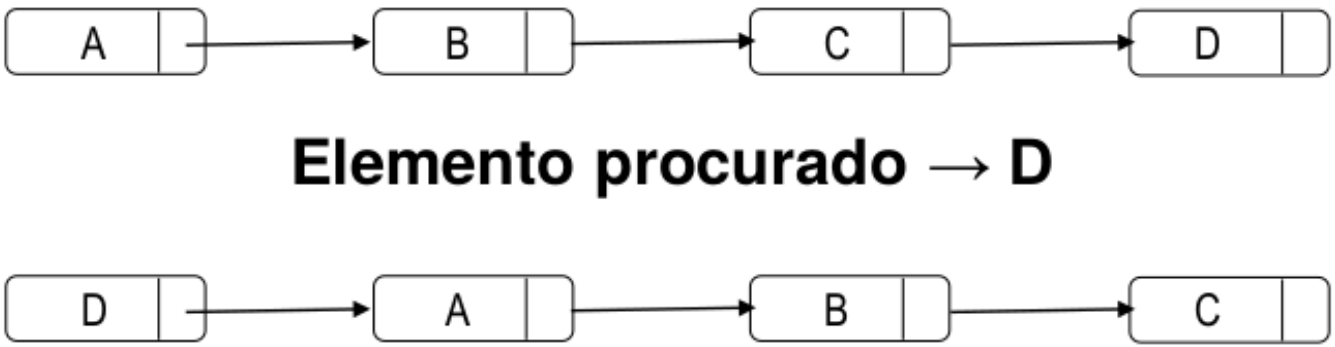
- **Método de mover para frente**
  - O elemento pesquisado vai para o início da lista.



- Desvantagem:

Listas Auto Organizadas

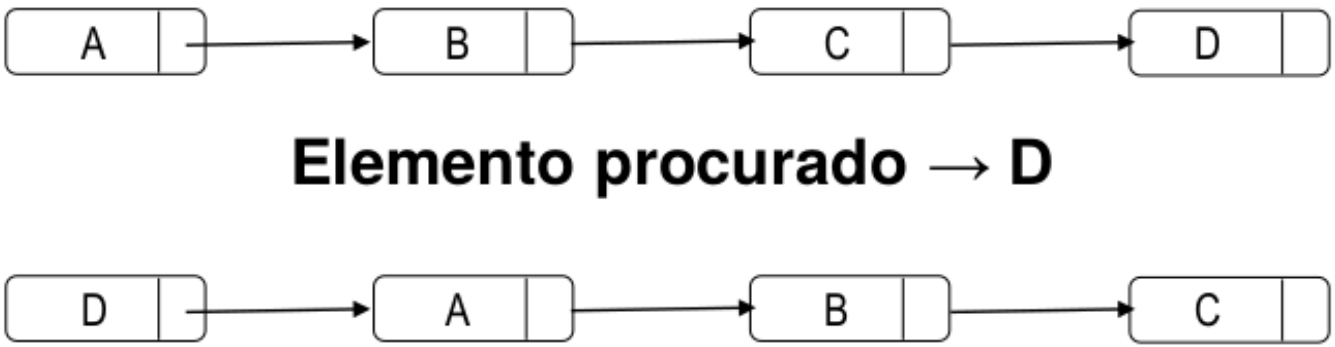
- **Método de mover para frente**
  - O elemento pesquisado vai para o início da lista.



- Desvantagem:
  - Uma única pesquisa pelo elemento não implica que o registro será frequentemente pesquisado.

Listas Auto Organizadas

- **Método de mover para frente**
  - O elemento pesquisado vai para o início da lista.



- Desvantagem:
  - Uma única pesquisa pelo elemento não implica que o registro será frequentemente pesquisado.
- O custo desse método é o mesmo para uma implementação estática?

Listas Auto Organizadas

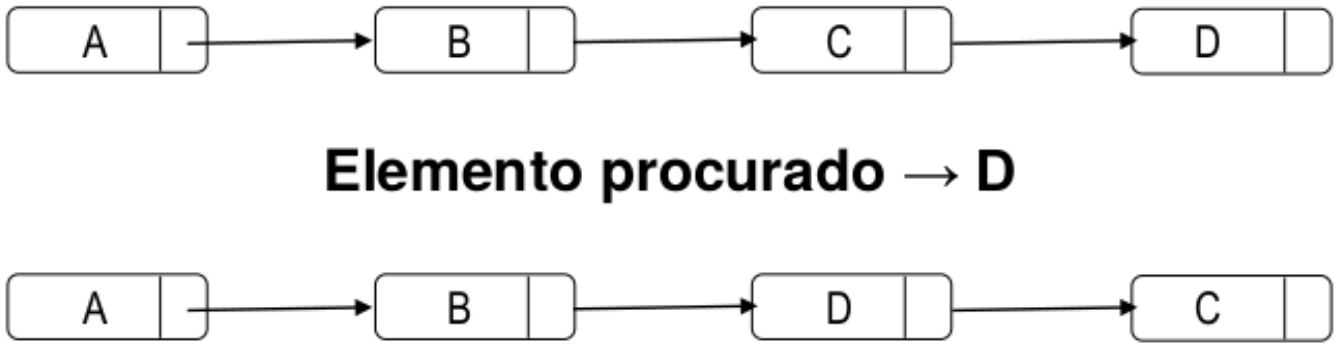
- **Método de transposição**

Listas Auto Organizadas

- **Método de transposição**
  - Sempre que encontrar o elemento, troque-o com seu predecessor, exceto se for o primeiro elemento.

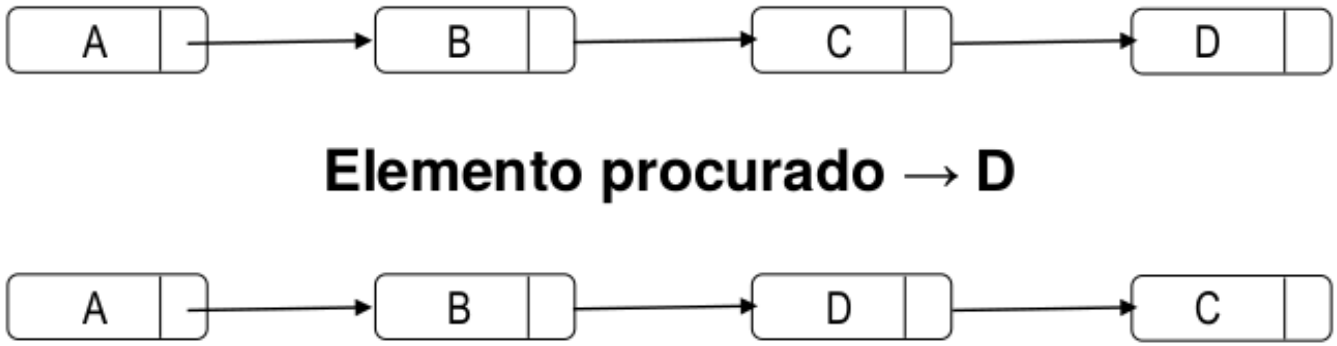
Listas Auto Organizadas

- **Método de transposição**
  - Sempre que encontrar o elemento, troque-o com seu predecessor, exceto se for o primeiro elemento.



Listas Auto Organizadas

- **Método de transposição**
  - Sempre que encontrar o elemento, troque-o com seu predecessor, exceto se for o primeiro elemento.



- Qual a vantagem deste método se comparado ao método mover para frente?

Listas Auto Organizadas

- **Método de contagem**

Listas Auto Organizadas

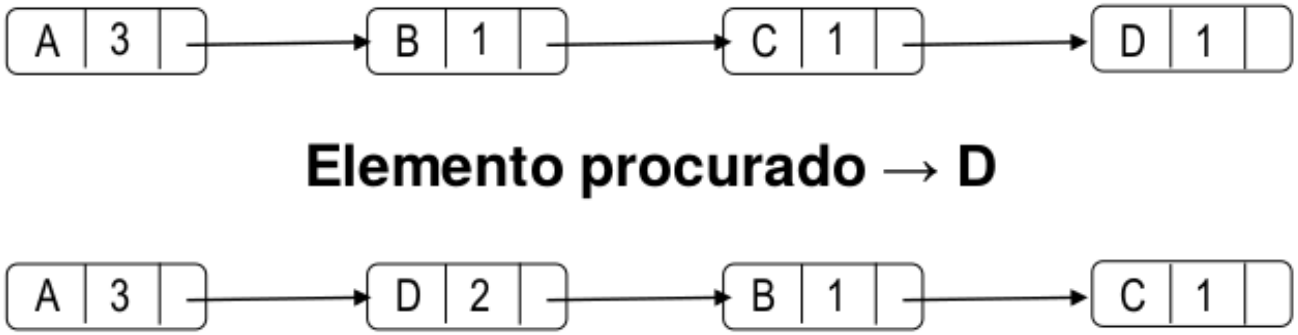
- **Método de contagem**
  - Armazena um campo de contagem, sempre que um elemento é pesquisado esse campo é incrementado.

Listas Auto Organizadas

- **Método de contagem**
  - Armazena um campo de contagem, sempre que um elemento é pesquisado esse campo é incrementado.
  - A lista é ordenada pelo número de vezes que os elementos são acessados.

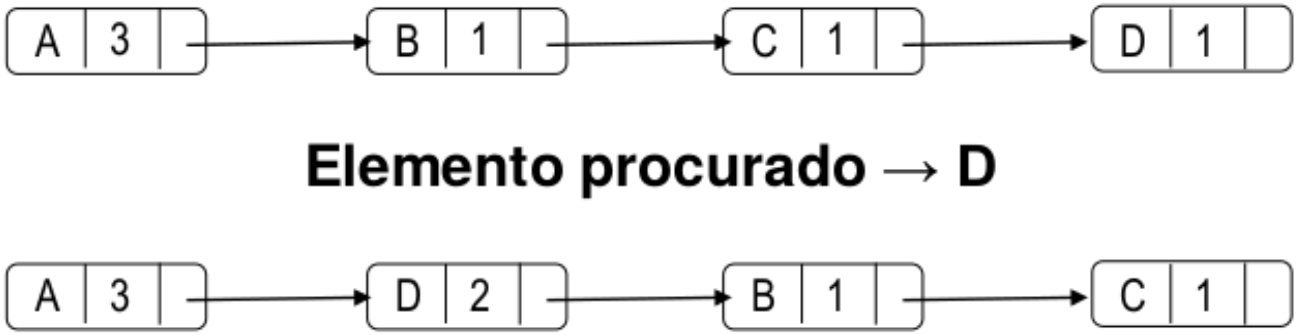
Listas Auto Organizadas

- **Método de contagem**
  - Armazena um campo de contagem, sempre que um elemento é pesquisado esse campo é incrementado.
  - A lista é ordenada pelo número de vezes que os elementos são acessados.



Listas Auto Organizadas

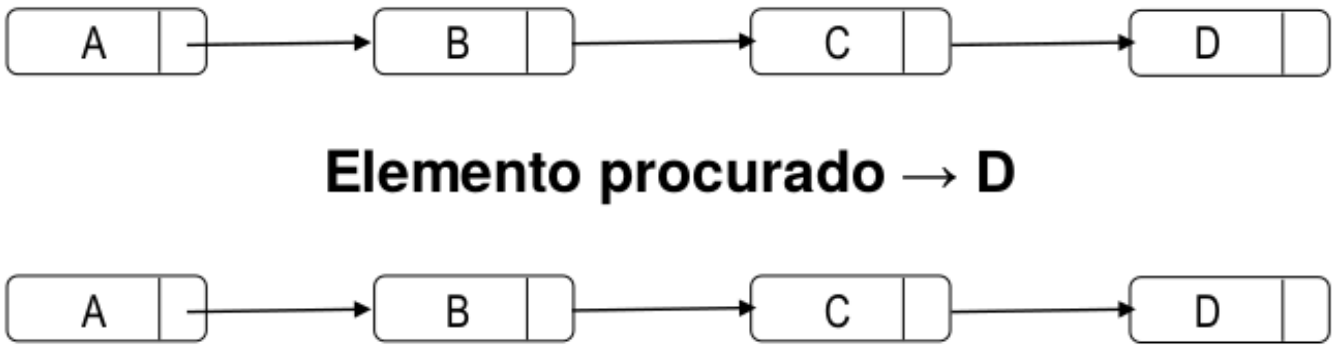
- **Método de contagem**
  - Armazena um campo de contagem, sempre que um elemento é pesquisado esse campo é incrementado.
  - A lista é ordenada pelo número de vezes que os elementos são acessados.



- Caso o campo de contagem seja parte da informação, este método pode ser considerado como **método ordenado.**

Listas Auto Organizadas

- **Método de ordenação**
  - A ordenação da lista depende do campo de informação. Ao contrário do método de contagem, neste tipo de ordenação qualquer campo pode ser escolhido, por exemplo: menor nota, maior salário, ordem alfabética, etc.



Lista Ordenada

- Ordena os valores em função do campo de conteúdo de seus nós (nas aulas, o campo **info**).

- Vantagem na busca pela informação, porém se esta não se encontra na lista, a pesquisa percorre até encontrar o seu fim (campo **prox** igual a **NULL**).

Lista Ordenada

- Insere um nó mantendo a lista ordenada.

Lista Ordenada

- Insere um nó mantendo a lista ordenada.
- Se a lista estiver vazia:

Lista Ordenada

- Insere um nó mantendo a lista ordenada.

- Se a lista estiver vazia:
  - O ponteiro externo deve apontar para o nó a ser inserido.

Lista Ordenada

- Insere um nó mantendo a lista ordenada.
- Se a lista estiver vazia:
  - O ponteiro externo deve apontar para o nó a ser inserido.
- Senão:

Lista Ordenada

- Insere um nó mantendo a lista ordenada.
- Se a lista estiver vazia:
  - O ponteiro externo deve apontar para o nó a ser inserido.
- Senão:
  - Encontra o nó maior ou igual ao nó atual.

Lista Ordenada

- Insere um nó mantendo a lista ordenada.
- Se a lista estiver vazia:
  - O ponteiro externo deve apontar para o nó a ser inserido.
- Senão:
  - Encontra o nó maior ou igual ao nó atual.
  - O novo nó aponta para o nó atual.

Lista Ordenada

- Insere um nó mantendo a lista ordenada.
- Se a lista estiver vazia:
  - O ponteiro externo deve apontar para o nó a ser inserido.
- Senão:
  - Encontra o nó maior ou igual ao nó atual.
  - O novo nó aponta para o nó atual.
  - O nó anterior aponta para o novo nó.

Lista Ordenada pelo Preço

```c
int inserirOrdenada(CELULA** lista, PRODUTO elemento){
      CELULA *aux, *anterior;
      CELULA* p =(CELULA*)malloc(sizeof(CELULA));
      if(p == NULL){
        printf("\n Erro de alocacao");
        return 1;
      }
      p->info = elemento;
      aux = (*lista);
      while(aux != NULL){
        if(elemento.preco <= aux->info.preco){
            break;
        }
        anterior = aux;
        aux = aux->prox;
      }
      p->prox = aux;
      if(aux == (*lista)){
        (*lista) = p;
      }else{
         anterior->prox = p;
      }
   return 0;
}
```

```c
typedef struct sProduto{
    int id;
    char nome[50];
    float preco;
}PRODUTO;
```

Exercícios

1. Considere uma lista encadeada com os seguintes elementos inteiros:
   - 6, 2, 3, 13, 56,13, 77, 88, 2, 78, 90, 1, 5, 133;
2. Realize as seguintes operações nesta lista. Desenhe a lista após cada operação.

1. PesqMoveParaFrente(90);
2. PesqTransposicao(56);
3. PesqTransposicao(5);
4. PesqTransposicao(56);
5. PesqMoveParaFrente(133);
6. PesqContagem(5);
7. PesqTransposicao(5);
8. PesqTransposicao(90);
9. PesqContagem(78);
10. PesqContagem(78);
11. PesqTransposicao(78);
12. PesqMoverParaFrente(2);

Bibliografia Básica

- DROZDEK, Adam. Estrutura de Dados e Algoritmos em C++. Editora Pioneira *Thomson Learning*, 2005. Pág 91, seção 3.5 (Listas Auto-Organizadas)- até pág. 94