Listas Encadeadas

Prof. Alberto Costa Neto

Recapitulando: O problema

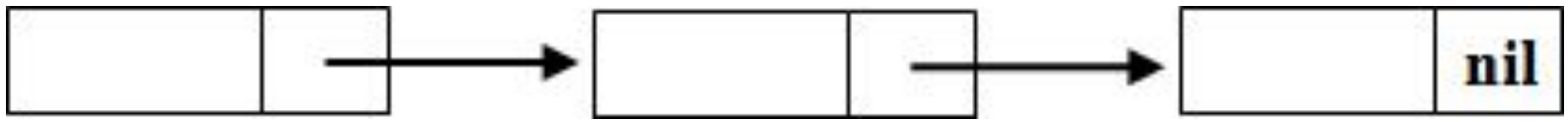
- **Dificuldade em definir quantidade exata de dados armazenados:**
  - Como saber quantos alunos teremos em uma Universidade?
  - Ou quantos veículos teremos em um país?
  - Ou quantos computadores estão ligados à Internet?
- Além disso, **parte dos dados pode ser necessária apenas em certas etapas do processamento.**
- Escolher o tamanho das EDs estaticamente normalmente incorre em **desperdício** ou **limita o tamanho da entrada de dados.**

Alocação Estática de Memória

- Toda a memória que pode vir a ser necessária é **alocada toda de uma vez**
- **Sem considerar a quantidade realmente necessária** em cada execução do programa
- O **máximo** de alocação possível **é ditado pelo hardware**, ou seja, pelo tamanho da memória “endereçável”

Lista Encadeada (ou ligada): Definição

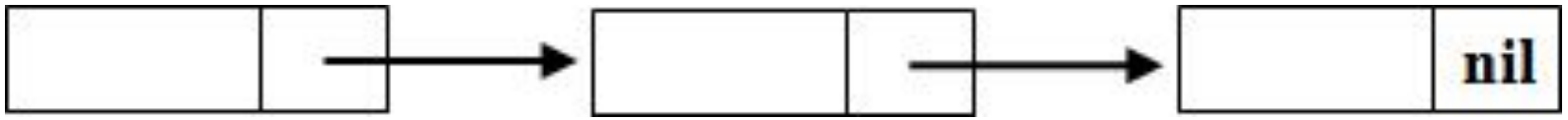
- É uma **coleção de registros** em que **cada registro tem um campo que indica a localização do próximo registro** na lista.
- Assim, a "**ordenação**" é fornecida explicitamente por esse campo.
- Esquematicamente podemos representar uma lista ligada (encadeada) da seguinte forma:



- **Nil** em inglês significa **nada**. Em C, seria o correspondente a NULL.

Lista Encadeada

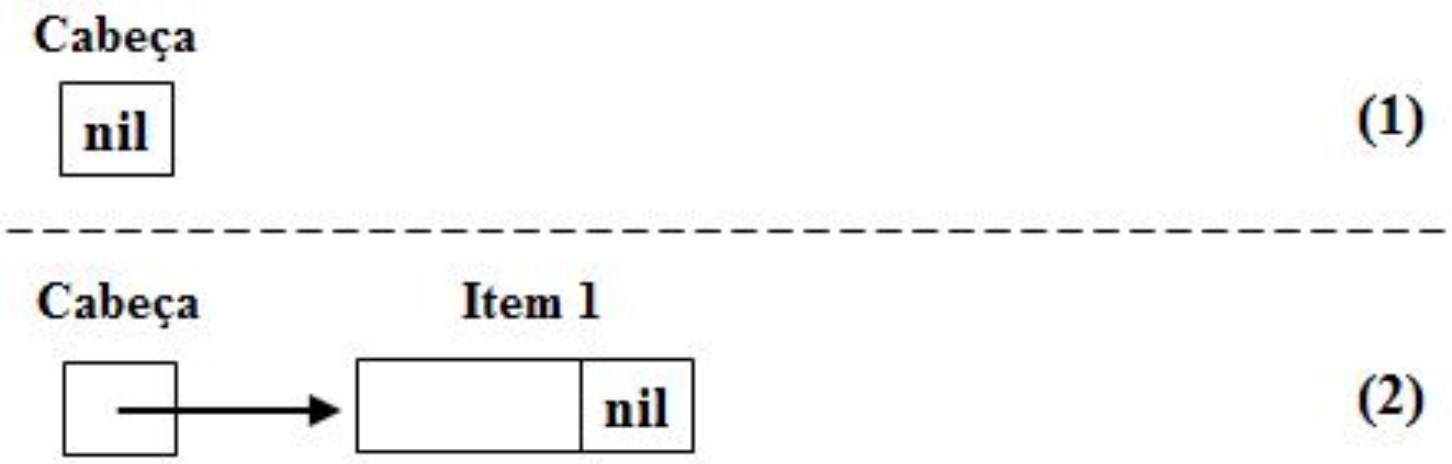
- O registro que representa cada componente (Nó) da lista possui campos com:
  - **Item** contendo o valor efetivamente armazenado;
  - **Apontador** para o próximo registro, podendo ou não ter uma certa sequência de ordenação.
- A Lista em si, pode ter vários campos, mas há dois bastante comuns:
  - **Cabeça**: Ponteiro para o primeiro Nó da lista.
  - **Cauda**: Ponteiro para o último elemento da lista. Para indicar que não há elementos após a cauda, atribui-se NULL (nil) ao apontador para o próximo.



Inserção

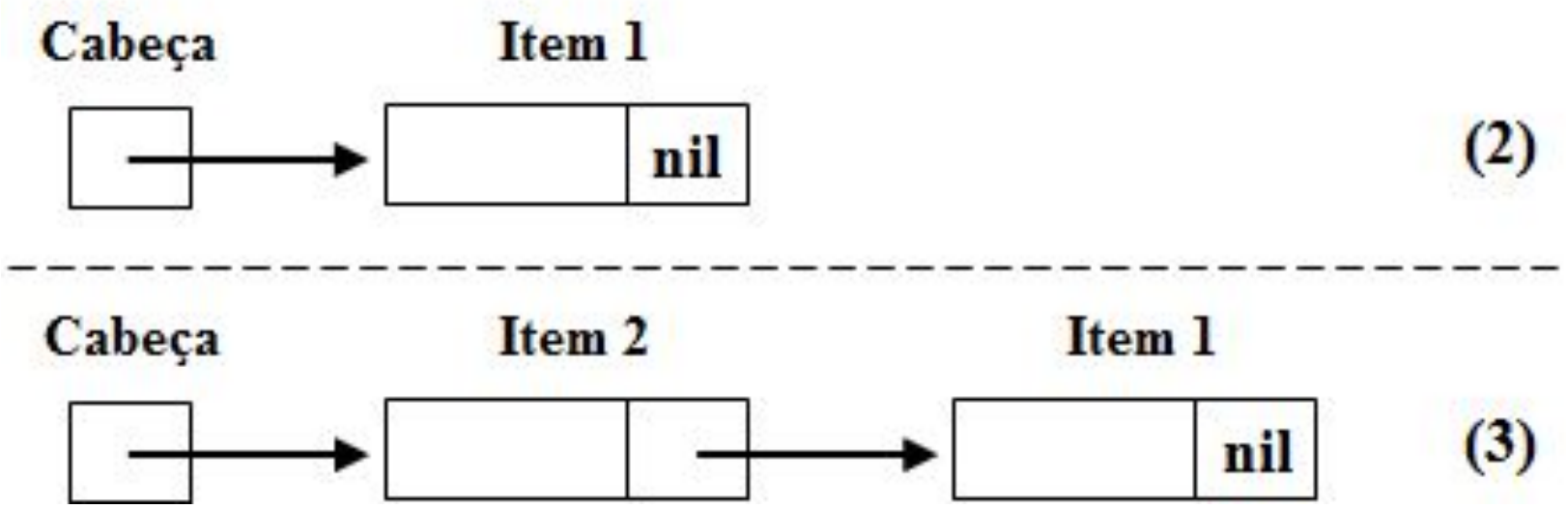
Inserção

- A situação mais simples (1) é quando a lista está vazia:
  - **Criar um novo nó**;
  - Fazer a **cabeça apontar para ele**;
  - Este **novo nó tem como próximo NULL** (nada).



Inserção

- Outra situação (2) é quando a lista já contém algum nó:
  - Neste caso, a inserção mais rápida, eficiente e fácil é na cabeça.
  - O novo nó criado aponta para antiga cabeça da lista



Outras opções de Inserção

- Poderia **escolher** uma **posição específica entre 0 e o tamanho** da lista!
- Também poderia inserir seguindo a ordem da chave. Neste caso, teríamos uma **lista Ordenada.**
- Isso ficará como exercício para vocês!!!

Busca e Alteração

Busca

- A busca consiste em partir da cabeça da lista procurando pelo item e retornar sua posição.
- Caso não ache o item na cabeça, segue o apontador para o próximo nó da lista até:
  - Encontrar; ou
  - Chegar ao final sem encontrar.
- Quando acha o item, retorna sua posição
  - Poderia retornar o apontador para o Nó

Alteração

- Dado um novo item e a posição, parte-se da cabeça até chegar na posição e substituir o valor atual pelo novo item.
  - Quando não acha, retorna false

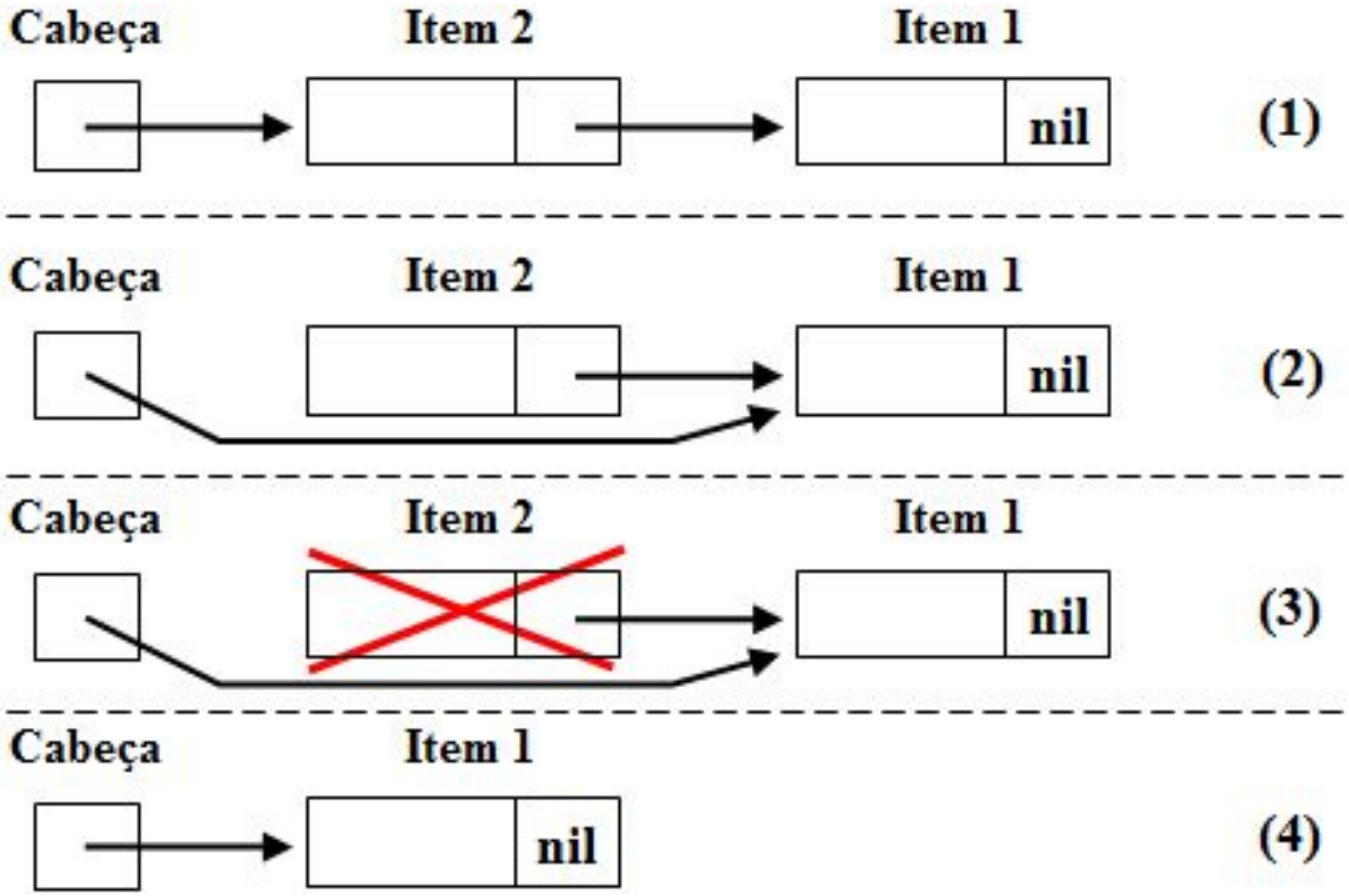
Remoção

Remoção na cabeça

Quando o **item a ser removido encontra-se na cabeça** da lista, executam-se os seguintes passos:

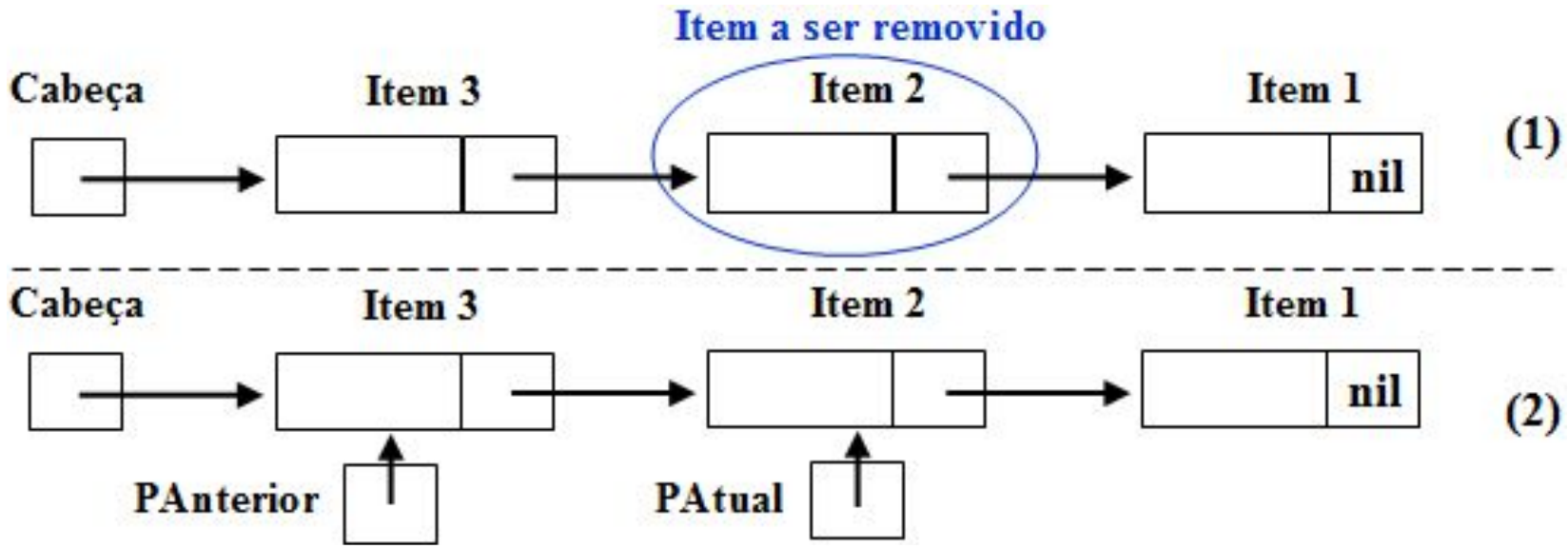
1. Acessar o item a ser removido e **guardar o apontador para o próximo** item (segundo) da lista.
2. Fazer com que o apontador para a **cabeça passe a apontar para o item seguinte** à cabeça da lista, através do apontador obtido no passo 1.
3. Já que a cabeça já foi ajustada, **liberar a memória do nó removido**.

Remoção na cabeça



Remoção no meio ou na cauda

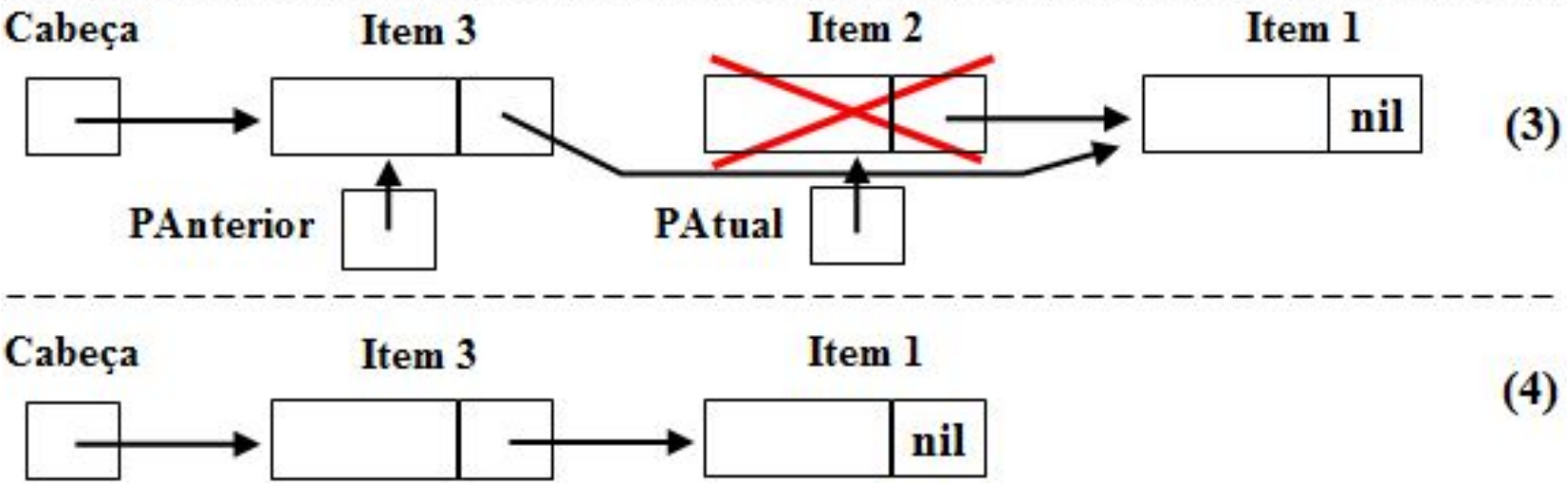
Remoção no meio ou na cauda



Quando o item a ser removido não está na cabeça da lista, é necessário atualizar o apontador prox do item anterior:

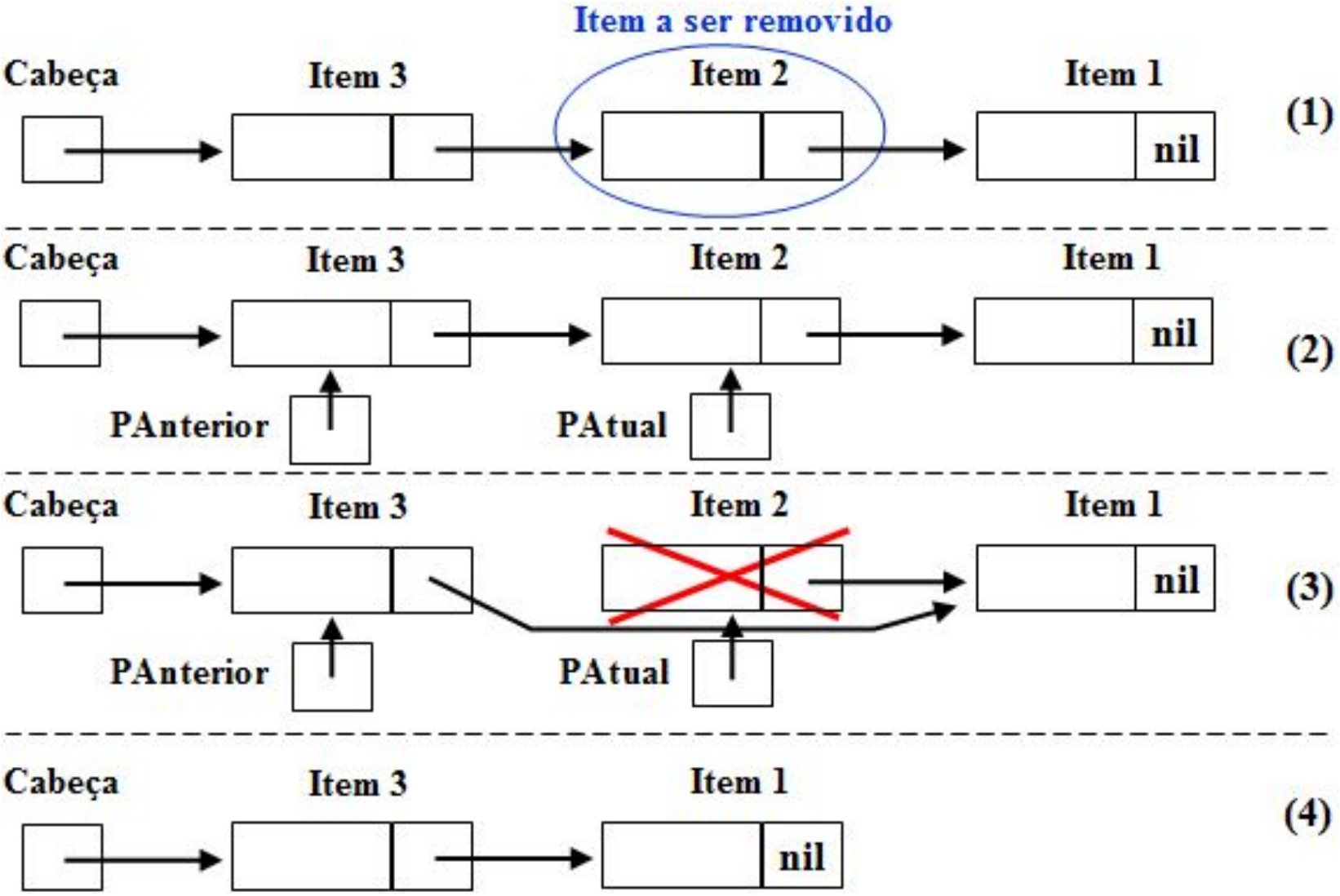
- Usamos um apontador (PAtual) para procurar o Item a ser removido e outro (PAnterior) para guardar o Item anterior na lista encadeada.

Remoção no meio ou na cauda



- De posse de PAtual e PAnterior, podemos ajustar o apontador prox de PAnterior para o próximo item apontado por PAtual.
- Liberar a variável dinâmica que guarda o item removido (3).
- Ao final, temos uma lista encadeada com um item a menos e com todos os apontadores ajustados.

Remoção no meio ou na cauda (todos os passos)



Vantagens e Desvantagens de Listas Encadeadas

Vantagens e Desvantagens

- **Vantagem:** Diferentemente das listas sequenciais baseadas em array, as encadeadas **podem aumentar e reduzir de tamanho dinamicamente**, podendo assim ser usadas em diversas situações **sem comprometer o uso de memória**.
- **Desvantagens:**
  - Necessidade de **armazenamento adicional** para guardar os apontadores.
  - **Suporta apenas a pesquisa sequencial**.
  - **Acesso por posição é mais lento** porque demanda passar por todos os nós anteriores.

Sugestões de Estudo

Estruturas de Dados (Nina Edelweiss)

- Seção 3.4

Projeto de Algoritmos com implementações em Java e C++ (Nivio Ziviani)

- Seção 3.1.2

Estruturas de dados (Paulo Veloso)

- Seção 5.4.2 e 5.5