Aula 12: Alocação Encadeada

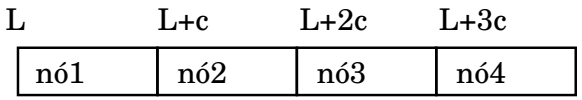
- Conceito e Modelo de Alocação Encadeada
- Lista de espaço disponível
- Algoritmos de manipulação da lista de espaço disponível

Características da Alocação Encadeada de Memória

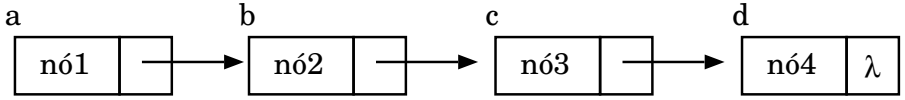
- Posições de memória são alocadas (ou desalocadas) na medida em que são necessárias (ou dispensadas)

- Os nós de uma lista encontram-se aleatoriamente dispostos na memória e são interligados por ponteiros, que indicam a posição do próximo elemento da tabela.

  - A figura a seguir apresenta uma lista linear em suas representações sequencial e encadeada.



Animação



Vantagens e Desvantagens da Alocação Encadeada

Vantagens:

- Facilidade para inserir elementos na lista
- Facilidade para remover elementos na lista
- Facilidade para unir elementos na lista
- Facilidade para separar elementos na lista
  - Estas facilidades devem-se à manipulação de ponteiros, como veremos adiante.

Desvantagens:

- Gasto maior de memória em virtude da necessidade de manter variáveis-ponteiro;
- Necessidade de se percorrer a lista para acesso a um elemento individual (na alocação sequencial o acesso é imediato).

Lista de Espaços Disponível (LED)

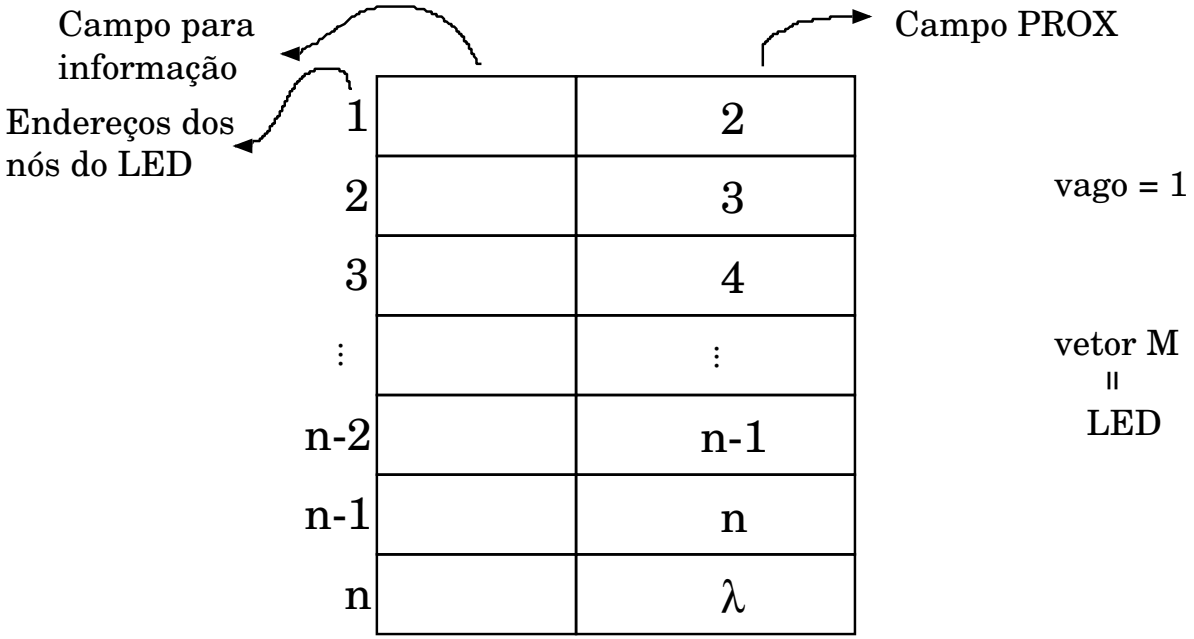
- Necessidade: com as sucessivas inserções e remoções nas listas encadeadas, há necessidade de um gerenciamento.
  - Encontar novas posições de memória para armazenamento
  - Liberar posições de memória para posterior reutilização
- LED: Lista que contém posições de memória ainda não utilizadas ou dispensadas após sua utilização, como um "armazém de posições de memória".
  - Observação: a LED e as estruturas definidas pelos programas compartilham a mesma memória disponível

Primeira Implementação da LED

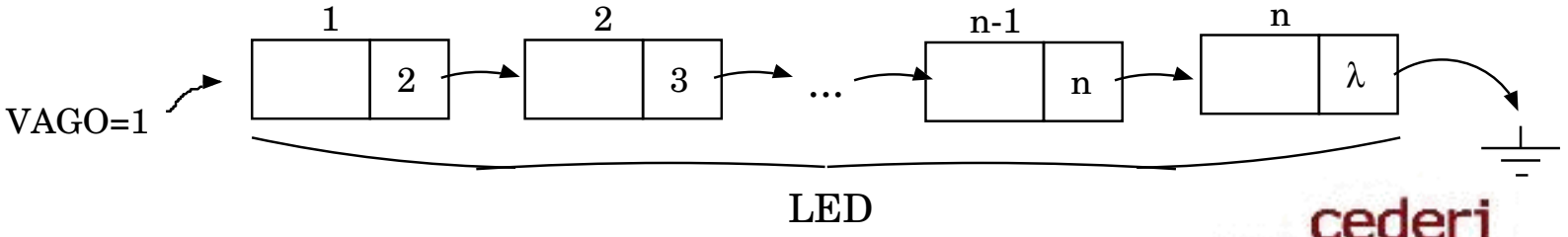
- Idéia: Dimensionamento de um único vetor dos nós M simulando a memória total disponível.
- Inicialização: No princípio, como a memória se acha totalmente disponível, todos os seus nós são encadeados na LED.
- Passos da inicialização:
  - Criar uma variável VAGO para se referir ao primeiro nó disponível da LED.
  - Os campos ponteiros dos nós são encadeados sequencialmente.
  - O ponteiro VAGO é inicializado com o endereço do primeiro nó da lista encadeada
  - O campo ponteiro do último nó recebe o valor λ, indicando fim de lista.

Inicialização da LED

Primeira Implementação



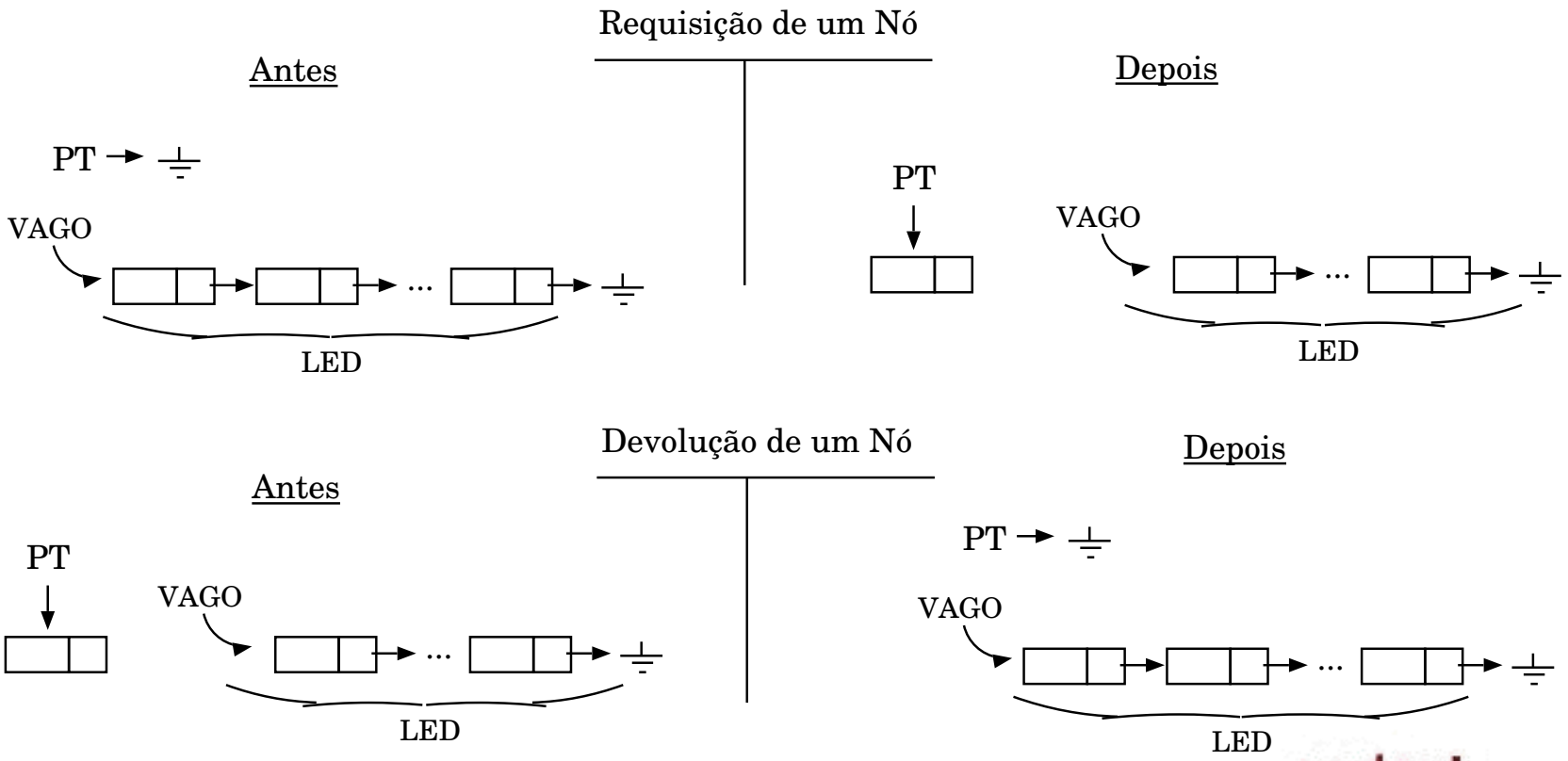
Representando com ponteiros:



Algoritmos de Manipulação da LED

Primeira Implementação

- Variável PT: Indica o índice do nó disponível



Algoritmos de Manipulação da LED

Primeira Implementação

- Variável PT: Indica o índice do nó disponível

Algoritmo: requisição de um novo nó da LED

```
procedimento OCUPAR(PT)
    se    VAGO ≠ λ
    então    PT   : =  VAGO
             VAGO   : = M[VAGO].PROX
    senão   "OVERFLOW"
```

Algoritmo: devolução de um novo nó da LED

```
procedimento DESOCUPAR(PT)
    M[PT].PROX   : =  VAGO
    VAGO         : =  PT
```

Outras Implementações da LED

- Os programadores em geral optam por utilizar a LED de acordo com facilidades já embutidas na linguagem de programação utilizada.

- Cada linguagem possui um módulo específico de gerência de memória disponível.

- O programador apenas se refere às rotinas internas de ocupação e devolução de nós da LED.

Outras Implementações da LED

Exemplo: PASCAL

- Requisição de um nó da LED

  new(pt) - devolve o endereço de um nó na variável ponteiro pt

- Devolução de um nó à LED

  dispose(pt) - devolve à LED o nó cujo endereço é dado pela variável ponteiro pt

- Obs: cada ponteiro utilizado é associado a um único nó. Assim, por exemplo,

  pt↑.info

  representa o campo info do nó apontado por pt