

Etapa 2: Estruturas de Dados Lineares - Listas encadeadas

Listas Encadeadas - Guia Didático

O que são Listas Encadeadas?

Imagine uma caça ao tesouro onde cada pista leva à próxima localização. É assim que funcionam as listas encadeadas:

- **Não são armazenadas em blocos contínuos** de memória (como arrays)
- Cada elemento sabe **onde está o próximo**
- São conectadas por **"ponteiros"** (endereços de memória)

Estrutura Básica

Cada "elo" da corrente (chamado de **nó**) contém:

...

[Dado | Ponteiro] → [Dado | Ponteiro] → [Dado | NULL]

...

Cada nó possui:

- **Dado**: a informação armazenada
- **Ponteiro**: o endereço do próximo nó na sequência

Tipos de Listas Encadeadas

1. Lista Simplesmente Encadeada

- **Funcionamento**: Só anda para frente
- **Navegação**: Unidirecional (→)
- **Exemplo**: `1 → 2 → 3 → NULL`
- **Característica**: Como uma rua de mão única

2. Lista Duplamente Encadeada

- **Funcionamento**: Anda para frente e para trás
- **Navegação**: Bidirecional (↔)
- **Exemplo**: `NULL ↔ 1 ↔ 2 ↔ 3 ↔ NULL`
- **Característica**: Como uma rua de mão dupla - cada nó sabe quem é o anterior e o próximo

3. Lista Circular

- **Funcionamento**: O último volta para o primeiro
- **Navegação**: Forma um ciclo infinito
- **Exemplo**: `1 → 2 → 3 → 1` (volta para o 1)
- **Característica**: Como uma pista circular de corrida

Vantagens e Desvantagens

✅ Vantagens

- **Inserção/remoção rápidas**: $O(1)$ em posições conhecidas
- **Tamanho flexível**: Cresce e diminui conforme necessário
- **Sem realocação**: Não precisa realocar memória como os arrays
- **Eficiente para operações sequenciais

❌ Desvantagens

- **Acesso lento**: $O(n)$ para buscar elementos (tem que percorrer a lista)
- **Memória extra**: Gasta espaço armazenando ponteiros
- **Cache ineficiente**: Dados espalhados na memória
- **Não suporta acesso direto por índice

Aplicações Práticas no Mundo Real

Sistemas de Navegação

- **Histórico do navegador**: Páginas visitadas em sequência
- **Voltar/Avançar**: Navegação entre páginas

Sistemas Operacionais

- **Gerenciamento de memória**: Blocos de memória livre/alocada
- **Lista de processos**: Tarefas em execução

Aplicações do Dia a Dia

- **Lista de contatos**: Nomes e telefones encadeados
- **Playlist de músicas**: Músicas em sequência
- **Lista de tarefas**: Afazeres pendentes

Estruturas de Dados

- **Pilhas e filas**: Implementadas sobre listas encadeadas
- **Tabelas hash**: Resolução de colisões por encadeamento

Como Funciona na Prática

Inserção no meio:

1. Encontre a posição desejada
2. Crie um novo nó
3. Ajuste os ponteiros:
 - Novo nó aponta para o próximo
 - Anterior aponta para o novo nó

Remoção:

1. Encontre o nó a ser removido
2. Ajuste o ponteiro do anterior para o próximo
3. Libere a memória do nó removido

Resumo Visual

****Simples****: Início → Nó1 → Nó2 → Nó3 → Fim

****Dupla****: Início ↔ Nó1 ↔ Nó2 ↔ Nó3 ↔ Fim

****Circular****: Nó1 → Nó2 → Nó3 → Nó1 (ciclo)

As listas encadeadas são ideais quando precisamos de flexibilidade no tamanho e fazemos muitas inserções/remoções, mas não são boas para buscas frequentes por posições específicas.

Perguntas de Múltipla Escolha - Listas Encadeadas

Pergunta 1: Inserção no Início

Em uma lista simplesmente encadeada, qual é a complexidade de tempo para inserir um novo nó no início da lista?

- a) $O(n)$ - Linear
- b) $O(1)$ - Constante
- c) $O(\log n)$ - Logarítmica
- d) $O(n^2)$ - Quadrática

Resposta: b) $O(1)$ - Constante

Pergunta 2: Diferença Estrutural

Qual é a principal diferença entre uma lista simplesmente encadeada e uma lista duplamente encadeada?

- a) A duplamente encadeada usa arrays internamente
- b) A simplesmente encadeada é sempre ordenada
- c) A duplamente encadeada permite navegação nos dois sentidos
- d) A simplesmente encadeada ocupa mais memória

Resposta: c) A duplamente encadeada permite navegação nos dois sentidos