

Sistema de Gerenciamento de Pedidos – Tia Lu Delivery

Foco: Implementação da Árvore AVL para Indexação Eficiente

Discentes: Aloisio Caldas da Silva Junior
Eduardo Sousa da Silva
Eveny Castro de Almeida
Iran Pablo Santos Martins
Thiago Sanches Hohlenwerger

Resumo

O objetivo dessa apresentação é :

Demonstrar como a Árvore AVL
otimizou as buscas no sistema de
pedidos.

1.Introdução: Contexto do projeto e importância da Árvore AVL.

2.Fundamentação teórica: Conceitos e características da Árvore AVL.

3.Metodologia: Implementação e integração no sistema.

4.Resultados e discussões: Testes e desempenho obtidos.

5. Considerações finais: Conclusões e melhorias futuras

Introdução



Contexto do Projeto:

- Sistema de gerenciamento de pedidos para restaurante
- Necessidade de buscas rápidas por código de pedidos e itens
- Grande volume de operações de consulta

Problema:

- Estruturas lineares (listas) tornam-se ineficientes com crescimento de dados
- Buscas sequenciais com complexidade $O(n)$

Solução:

- Implementação de Árvore AVL para indexação eficiente
- Buscas com complexidade $O(\log n)$
- Estrutura auto-balanceada

```
----- SISTEMA DE PEDIDOS - LU DELIVERY -----
1 - Registrar Item
2 - Atualizar Item
3 - Listar Items
4 - Criar Pedido
5 - Listar Pedidos (ordenados por código)
6 - Processar Pedido Pendente
7 - Atualizar Status de Pedido
8 - Cancelar Pedido
9 - Relatório
0 - Sair

Escolha uma opção: 4
Nome do cliente: Aloisio

Itens disponíveis para o pedido:
Código: 1, Nome: Smash Burger Classico, Preço: R$ 10.00, Estoque: 20
Código: 2, Nome: Cheddar Bacon, Preço: R$ 25.00, Estoque: 20
Código: 3, Nome: X-Egg Salada, Preço: R$ 20.00, Estoque: 19
Código: 4, Nome: Duplo Monster, Preço: R$ 32.00, Estoque: 14
Código: 5, Nome: Hot Dog Tradicional, Preço: R$ 12.00, Estoque: 30
Código: 6, Nome: Dog Texano, Preço: R$ 10.00, Estoque: 20
Código: 7, Nome: Dog Especial da Casa, Preço: R$ 16.00, Estoque: 25
Código: 8, Nome: Frango com Creme de Ricota, Preço: R$ 14.00, Estoque: 18
Código: 9, Nome: Atum Fit, Preço: R$ 13.00, Estoque: 18
Código: 10, Nome: Veggie Grão-de-Bico, Preço: R$ 17.00, Estoque: 15
Código: 11, Nome: Batata Palito, Preço: R$ 10.00, Estoque: 40
Código: 12, Nome: Batata Cheddar e Bacon, Preço: R$ 10.00, Estoque: 25
Código: 13, Nome: Onion Rings, Preço: R$ 14.00, Estoque: 19
Código: 14, Nome: Refrigerante Lata, Preço: R$ 6.00, Estoque: 49
Código: 15, Nome: Suco Natural, Preço: R$ 6.00, Estoque: 29
Código: 16, Nome: Milkshake, Preço: R$ 15.00, Estoque: 20
Código: 17, Nome: Brownie Caseiro, Preço: R$ 8.00, Estoque: 24
Código: 18, Nome: Pudim no Pote, Preço: R$ 7.00, Estoque: 25
Código: 19, Nome: Acai no Copo, Preço: R$ 12.00, Estoque: 30
Código: 20, Nome: , Preço: R$ 2.50, Estoque: 250

Informe o código do item para adicionar (ou ENTER para finalizar): 20
Quantidade de "": 1
Item "" adicionado ao pedido.
Informe o código do item para adicionar (ou ENTER para finalizar):
Pedido registrado com sucesso!
```

Fundamentação Teórica

Árvore AVL - Conceitos:

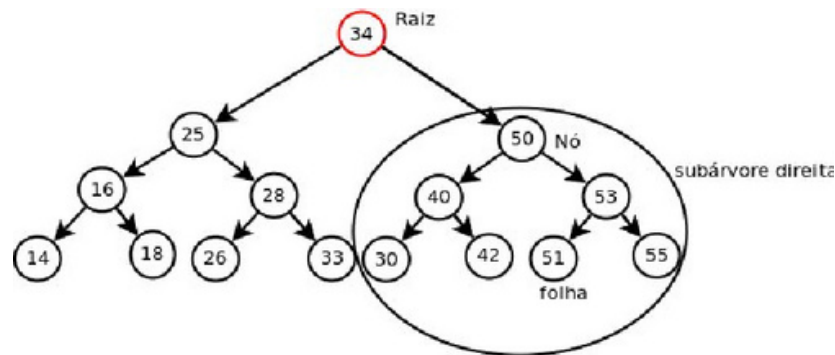
- Árvore de busca binária balanceada
- Fator de balanceamento: diferença entre alturas das subárvores
- Balanceamento automático através de rotações

Características:

- Complexidade $O(\log n)$ para buscas, inserções e remoções
- Estável: mantém ordem dos elementos
- In-place: uso eficiente de memória

Operações de Balanceamento:

- Rotações simples (direita/esquerda)
- Rotações duplas (esquerda-direita/direita-esquerda)



Implementação

```
class AVLNode:
    def __init__(self, key, value):
        self.key = key
        self.value = value
        self.left = None
        self.right = None
        self.height = 1
```

Integração no Sistema:

- Módulo `arvore_avl.py` isolado e especializado
- Função `build_from_list()` para carregamento inicial
- Método `search()` para buscas rápidas por código
- Indexação automática de itens e pedidos

Fluxo de Dados:

- JSON → Árvore AVL → Buscas Rápidas → Relatórios

Resultados e discussões

A Árvore AVL trouxe ganhos significativos de performance ao sistema. As buscas por código tornaram-se instantâneas, com complexidade $O(\log n)$ comprovada na prática. O tempo de resposta é imperceptível mesmo com o crescimento dos dados. O auto-balanceamento manteve a eficiência durante todas as operações, garantindo altura logarítmica constante. A integração com o sistema foi harmoniosa, onde a árvore atua como índice eficiente e o Bubble Sort cuida da ordenação final para relatórios.

Considerações finais

A Árvore AVL mostrou-se ideal para o sistema, atendendo todos os requisitos de performance.

A estrutura balanceada garantiu velocidade constante nas buscas, essencial para a experiência do usuário.

A complexidade de implementação foi compensada pelos benefícios em escalabilidade e organização do código. Para o futuro, sugere-se adicionar cache para consultas frequentes e índices secundários para outros campos de busca.

Referências

1. **Cormen, T. et al. Introduction to Algorithms. MIT Press, 2009.**
2. **Sedgewick, R.; Wayne, K. Algorithms. Addison-Wesley, 2011.**
3. **Documentação do Python - Estruturas de Dados**
4. **Código fonte do projeto**