Relatório Técnico-Científico: Sistema de Gerenciamento de Pedidos

Projeto Elaborado Por

Jamile Martins Coutrim

Docente: Lucas Almeida Silva

Sistema de Informação - Estrutura de Dados

Itabuna- Ba 2025

Introdução

Os sistemas de gerenciamento de pedidos são fundamentais para o funcionamento de restaurantes e lanchonetes, pois permitem organizar itens do cardápio, registrar pedidos e acessar informações de forma rápida e estruturada. No entanto, a eficiência desse processo depende diretamente da forma como os dados são armazenados, ordenados e recuperados pelo sistema. Para isso, técnicas de ordenação e estruturas de dados avançadas desempenham um papel essencial.

Neste relatório, apresentamos a implementação de um sistema de pedidos em console utilizando a linguagem Python. O sistema utiliza:

(a) Mapas (dict) para representar itens e pedidos;
(b) Algoritmo de ordenação Insertion Sort, aplicado aos pedidos;
(c) Árvore AVL como indexador dos dados;
(d) Armazenamento persistente em JSON;
(e) Carregamento automático dos dados na inicialização.

O objetivo deste trabalho foi compreender e aplicar na prática os conceitos de ordenação e árvores balanceadas, explorando sua estrutura, desempenho e aplicabilidade no desenvolvimento de sistemas reais.

Fundamentação Teórica

2.1 Algoritmos de Ordenação

A ordenação consiste no processo de organizar elementos de uma lista de acordo com um critério específico, como número ou nome. A escolha do algoritmo impacta diretamente a performance do sistema.

Insertion Sort

O algoritmo implementado pela equipe foi o Insertion Sort. Ele funciona de maneira semelhante à forma como organizamos cartas na mão: para cada elemento, o algoritmo o compara com os anteriores até encontrar sua posição correta.

Pseudocódigo do Insertion Sort

```
para i de 1 até n:
   atual = lista[i]
   j = i - 1
   enquanto j >= 0 e lista[j] > atual:
      lista[j + 1] = lista[j]
      j = j - 1
   lista[j + 1] = atual
```

Características do algoritmo:

Complexidade média e pior caso: $O(n^2)$
Melhor caso: O(n) (quando a lista já está ordenada)
Estável (não troca elementos iguais de posição)
Simples de implementar
Eficiente para listas pequenas

2.2 Árvores Binárias de Busca (AVL)

Uma árvore AVL é um tipo de árvore binária de busca autobalanceada. Ela mantém o fator de balanceamento entre -1 e 1, evitando degeneração da árvore e garantindo eficiência nas operações.

Características

Altura garantida: O(log n)
Operações de busca, inserção e remoção: O(log n)
Uso de rotações para manter o balanceamento
Excelente para sistemas que precisam de busca rápida

Funcionamento no Projeto

No sistema, a árvore AVL é utilizada como indexador para itens e pedidos. Isso significa que, após carregar o JSON, cada elemento é inserido na árvore para permitir consultas mais eficientes.

Metodologia

A implementação do sistema foi dividida em etapas:
3.1 Representação dos dados

Cada item e pedido foi modelado usando mapas (dict):
Item:
{ "id": "...", "nome": "...", "preco": ... }
Pedido:
{ "id": "...", "cliente": "...", "itens": [...] }
3.2 Armazenamento em JSON
Os dados de itens e pedidos são salvos no arquivo:
dados.json
Esse arquivo é carregado automaticamente ao início do sistema.

3.3 Funções implementadas
registrar_item()
mostrar_cardapio()
realizar_pedido()
relatorio_pedidos()
carregar_em_avl()
Cada função cuida de uma parte isolada do sistema, facilitando organização e manutenção.

3.4 Ordenação com Insertion Sort
Foi aplicado para ordenar os pedidos pelo número do ID antes da exibição do relatório.

3.5 Indexação usando AVL
Ao inicializar o sistema, todos os itens e pedidos são carregados em duas árvores AVL distintas:
Árvore de Itens
Árvore de Pedidos
Isso permite buscas mais rápidas e demonstra o uso prático dessa estrutura.

Resultados e Discussões

4.1 Prints e Funcionamento
A) Cardápio carregado
===== CARDÁPIO =====
1 - X-Burger - R$ 15.0
2 - Refrigerante - R$ 5.0
B) Realização do pedido
Nome do cliente: Maria
Digite o ID do item: 1
Item 'X-Burger' adicionado!
Digite o ID do item: sair

Pedido realizado com sucesso!
ID do pedido: 1
C) Relatório ordenado com Insertion Sort
===== RELATÓRIO DE PEDIDOS =====
Pedido 1 | Cliente: Maria | Itens: 1

4.2 Discussão
O uso de Insertion Sort mostrou ser suficiente para a quantidade pequena de pedidos esperada no sistema. Em ambientes maiores, algoritmos mais sofisticados seriam recomendados.
A árvore AVL demonstrou clara vantagem em cenários de indexação, pois garante buscas eficientes e mantém os dados balanceados.
A integração entre JSON + dicts + AVL formou um sistema consistente, organizado e funcional.

Considerações Finais

Este trabalho permitiu compreender, de forma prática, a aplicação de estruturas fundamentais da Ciência da Computação. A implementação do algoritmo de ordenação e da árvore AVL no contexto de um sistema real de pedidos reforça conceitos como:
Estruturação de dados

- Busca eficiente
- Ordenação
- Persistência em arquivos
- Divisão modular do código

Como desafios, destacamos:

- Garantir o balanceamento correto da AVL
- Criar uma ordenação funcional e estável
- Evitar inconsistências no JSON ao manipular dados

Se houvesse mais tempo, seria possível:

- Implementar busca direta na árvore AVL pelo ID
- Criar relatório financeiro
- Adicionar remoção e edição de itens/pedidos
- Criar interface gráfica opcional
-

Referências

CORMEN, Thomas et al. Algorithms. MIT Press.
SEDGEWICK, Robert. Algorithms in Python. Addison-Wesley.
GOODRICH, Michael. Estruturas de Dados e Algoritmos em Python.
Material da disciplina – Árvores AVL e Ordenação.