Centro Universitário de Excelência
Sistemas de Informação

Indexador AVL para Otimização de Buscas em um sistema de restaurante

Autor: Danilo Motta Rodrigues
Gabriel da Silva Carvalho
Guilherme Viana Oliveira
Pedro Santos Simoes da Silva
Vitor Hugo Lima Silva Bittencourt

Vitória da Conquista, 2025

Agenda

O objetivo dessa apresentação é: Demonstrar a substituição de estruturas lineares por um indexador baseado em Árvore AVL para garantir buscas de complexidade logarítmica.

1. Contextualização:
O problema de escalabilidade das buscas lineares no sistema atual.

2. Fundamentação Teórica:
Conceitos de Árvore Binária de Busca (BST) e a necessidade de balanceamento.

3. Árvore AVL:
Definição, Fator de Balanceamento e Rotações.

4. Metodologia:
Estratégia de implementação do indexador e integração com JSON.

5. Considerações Finais:
Comparativo de desempenho (O(N) vs O(\log N)) e benefícios finais.

Contextualização

- Limitação atual.
- Problema de escala.
- Solução proposta.

Fundamentação Teórica

- Definição.
- Fator de Balanceamento.
- Garantia de performance.

```python
import json
import os
import arvore_avl

menu_de_itens = []
todos_pedidos = []
id = 1

raiz_itens = None
raiz_pedidos = None

if os.path.exists("restaurante.json") and os.path.getsize("restaurante.json") > 0:
    with open("restaurante.json", "r", encoding="utf-8") as arq:
        dados_json = json.load(arq)

    menu_de_itens = dados_json.get("itens", [])
    todos_pedidos = dados_json.get("pedidos", [])

    if menu_de_itens:
        id = max(item["id"] for item in menu_de_itens) + 1

else:
    menu_de_itens = []
    dados_json = {
        "itens": [],
        "pedidos": []
    }
    with open("restaurante.json", "w", encoding="utf-8") as arq:
        json.dump(dados_json, arq, ensure_ascii=False, indent=4)

    menu_de_itens = dados_json["itens"]
    todos_pedidos = dados_json["pedidos"]

for item in menu_de_itens:
    raiz_itens = arvore_avl.inserirNode(raiz_itens, item["id"], item)
for pedido in todos_pedidos:
    raiz_pedidos = arvore_avl.inserir_pedido(raiz_pedidos, pedido)

def registrar_item():
```

Árvore AVL

- O Gatilho:
  - Toda vez que inserimos um pedido, o algoritmo calcula o Fator de Balanceamento (Altura Esquerda - Altura Direita).:
- A regra:
  - Se o fator for maior que 1 ou menor que -1, a árvore está desequilibrada.
- A solução:
  - O algoritmo executa uma Rotação (Simples ou Dupla) para reorganizar os nós e manter a altura logarítmica.

Metodologia

- Integração com o sistema.
- Rebalanceamento Dinâmico:
  - Rotação simples.
  - Rotação dupla.

```
fila_pedidos_pendentes.append(pedido)
todos_pedidos.append(pedido)

global raiz_pedidos
raiz_pedidos = arvore_avl.inserir_pedido(raiz_pedidos, pedido)

dados_json["pedidos"] = todos_pedidos
with open("restaurante.json", "w", encoding="utf-8") as arq:
    json.dump(dados_json, arq, ensure_ascii=False, indent=4)
```

```
valor_total = 0

buscar_item = arvore_avl.buscarNode(raiz_itens, itens)
if buscar_item is None:
    print("Item não encontrado!")
    return
valor_total = buscar_item["preco"]

valido = 0
while valido == 0:
    desconto = input("Deseja adicionar algum cupom de desconto? (S
```

Considerações finais

- Eficiência comprovada.
- Escalabilidade.

Referências

- geeksforgeeks.org/avl-tree-set-1-insertion/
- https://www.datacamp.com/tutorial/avl-tree