RELATÓRIO: HASH TABLE

Curso: Tópicos em Ciência de Dados A (MAE013)

Professor: Milton R Ramirez

Integrantes

Leonardo Domingos (DRE: 120168324);

Hash Table

1. Introdução

Este projeto teve como objetivo desenvolver uma estrutura de dados eficiente para resolver o problema de deduplicação em datasets utilizando tabelas hash. A abordagem escolhida implementa uma tabela hash genérica que pode ser configurada com diferentes funções de dispersão e métodos de tratamento de colisões, permitindo a remoção de registros duplicados em complexidade O(n), superior à abordagem tradicional de ordenação com complexidade (O(n log n)).

O sistema desenvolvido permite:

- Inserção e busca de registros com chaves personalizáveis
- Configuração de diferentes algoritmos de hashing
- Escolha entre métodos de resolução de colisões
- Processamento eficiente de arquivos CSV
- Deduplicação baseada em campos-chave
- Operação via interface de programação e linha de comando

A solução foi implementada em Python.

2. Metodologia

O projeto foi estruturado em apenas uma classe principal:

2.1 Classe TabelaHash

Classe principal que implementa toda a funcionalidade da tabela de dispersão.
Funcionalidades:

- Armazenamento dos dados em estrutura linear (listas ou arrays)
- Configuração de diferentes funções de hash (divisão, multiplicação, dobra, meio-quadrado, extração)
- Implementação de métodos de resolução de colisões (encadeamento exterior e endereçamento aberto)
- Operações básicas (inserção, busca e remoção por chave)
- Interface para acesso via operadores ([] e in)
- Métodos para carregamento e processamento de arquivos CSV
- Funcionalidade toda de deduplicação

3. Comentários sobre o Desenvolvimento

Um dos primeiros obstáculos foi implementar as diferentes funções de hash de forma que funcionasse consistentemente com diversos tipos de chave (como números e strings). Além disso, algumas funções como a dobra e meio-quadrado não lidavam bem com strings longas, exigindo ajustes na conversão para valores numéricos.

A implementação dos métodos de resolução de colisões também trouxe dificuldades. O encadeamento exterior foi fácil, mas o endereçamento aberto exigiu cuidado especial para evitar loops infinitos quando a tabela ficava cheia.

Um desafio particular foi fazer a integração com arquivos CSV de forma flexível. A solução precisava:

1. Identificar automaticamente a coluna-chave
2. Lidar com cabeçalhos variáveis
3. Manter toda a linha do CSV como valor na tabela hash
4. Processar arquivos grandes de forma eficiente

Depois de várias tentativas, cheguei a um design onde a classe pode ser configurada com o nome da coluna-chave ou usar a primeira coluna por padrão, fazendo funcionar para diferentes formatos de arquivo.

Por fim, a otimização do tamanho da tabela hash em relação ao dataset processado foi uma consideração importante. Implementei uma lógica onde o tamanho inicial pode ser configurado conforme a necessidade, mas em uma versão futura poderia ser interessante adicionar redimensionamento automático para melhorar a eficiência com datasets muito grandes.