UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA UFRJ

GABRIEL SANTA BARBARA RODRIGUES SOUZA - 123356007
JULIA FERNANDA TERRA SOUZA - 123460458
SAMUEL DO NASCIMENTO CAROLA - 123537899

Tabela Hash e Deduplicação

Rio de Janeiro
2025

Comentários sobre o Desenvolvimento do Projeto HASH

Estruturas de Dados Utilizadas

O projeto utiliza como estrutura central uma tabela hash, implementada por meio de uma lista de listas (buckets), onde cada bucket armazena clientes cujo CPF gerou colisão para o mesmo índice hash. O tamanho da tabela é mantido sempre como um número primo para melhorar a distribuição dos dados. A tabela pode ser redimensionada dinamicamente para manter um bom fator de carga.

Divisão de Módulos

O código está dividido em dois módulos principais:

- **Hashing.py**: Contém toda a lógica da tabela hash, funções para gerar dados de teste, processar deduplicação e exportar resultados.
- **Hashing_interface.py**: Implementa uma interface gráfica com Tkinter, permitindo ao usuário gerar, processar, buscar e remover clientes sem necessidade de usar o terminal. (opcional apenas para uma melhor visualização).

Descrição das Rotinas e Funções

Classe TabelaHashCPF: Responsável por inserir, buscar, remover e deduplicar clientes pelo CPF, além de fornecer estatísticas como colisões, fator de carga e média por bucket.
Função _hash_cpf: Calcula o índice hash do CPF baseado no método da multiplicação, utilizando uma constante irracional.
inserir_cliente: Insere clientes na tabela apenas se o CPF for único.
buscar_cliente/remover_cliente: Permitem busca e remoção eficiente de clientes pelo CPF.
gerar_cpf_aleatorio e **criar_dataset_teste**: Auxiliam na criação de dados realistas para testes, incluindo duplicatas.
processar_dataset: Realiza a deduplicação efetiva e coleta estatísticas.
salvar_csv: Exporta o resultado final para um arquivo CSV.

Complexidades de Tempo e Espaço

A tabela hash, em média, proporciona inserção, busca e remoção em tempo O(1), desde que o fator de carga esteja controlado e a função de hash distribua bem os dados. O pior caso ocorre em caso de muitas colisões, tornando o acesso O(n), porém a escolha do tamanho primo da tabela e o redimensionamento dinâmico minimizam esses cenários. O espaço utilizado é proporcional à quantidade de clientes únicos processados.

Problemas e Observações

Durante o desenvolvimento, foi necessário ajustar o fator de carga para evitar excesso de colisões, e garantir que o redimensionamento preservasse todos os dados corretamente. A interface gráfica foi útil para facilitar testes, mas pode ser melhorada em usabilidade e

feedback. O tratamento de duplicatas foi simplificado pela estrutura da tabela hash, evitando pós-processamento.

Conclusão

O projeto alcançou seus objetivos de demonstrar a eficiência das tabelas hash para deduplicação de dados. As operações de inserção, busca e remoção mostraram-se rápidas mesmo para grandes conjuntos de dados simulados. As estatísticas geradas permitiram visualizar o comportamento interno da estrutura e a eficiência do método de hashing. O sistema é adequado para fins didáticos e pode ser utilizado como base para aplicações práticas envolvendo deduplicação por chaves únicas, como CPF.

OBS FINAL

Para futuras modificações talvez seja interessante colocar o projeto ABB mesclado com esse, aqui gera CPFs aleatórios, com o banco de dados da ABB maior utilizá-lo seja mais eficicaz para ter um controle.