Relatório — Projeto hash_duplicacao

Nome: Bruno Videira Pinho
DRE: 119.161.539

Objetivo

Este projeto implementa uma estrutura de dados **Tabela Hash** genérica em Python, com o objetivo de armazenar registros de forma eficiente, detectar e remover duplicatas. O projeto demonstra o uso de funções de hash customizáveis.

Estrutura de Diretórios

```
hash_duplicacao/
├── hash_map.py      # HashMap genérica
├── test.py          # Script de teste
└── test.csv         # Arquivo CSV de entrada com duplicatas
```

Principais Módulos e Funções

- **hash_map.py**:
  Implementa a classe `HashMap`, incluindo funções de hash, inserção única, remoção, busca, iteração e contagem de colisões/duplicatas.

- **test.py**:
  Script de exemplo que lê um arquivo CSV, insere os registros no HashMap, remove duplicatas e exibe estatísticas.

Funções e Métodos do HashMap

- `__init__(size=10, hash_function=None)`: Inicializa o HashMap com tamanho e função de hash opcionais.
- Funções de hash:
  - `division_hash(key)`: Função de hash por divisão (padrão).
  - `multiplication_hash(key)`: Função de hash por multiplicação.
  - `fold_hash(key)`: Função de hash por soma de partes da chave.
  - `redefine_hash_function(new_hash_function)`: Permite trocar a função de hash em tempo de execução.
- Funções de operação:
  - `insert(key, value)`: Insere ou atualiza um par chave-valor.
  - `find(key)`: Busca o valor associado à chave.
  - `remove(key)`: Remove um par chave-valor.
- Funções especiais para o problema pedido
  - `unique_insert(key, value)`: Insere apenas se a chave não existir (detecta duplicatas e colisões).

    - `__iter__()`: Itera sobre todos os valores armazenados.
    - `__len__()`: Retorna o número total de elementos.
    - `num_collisions`: Contador de colisões de hash.
    - `num_duplicates`: Contador de duplicatas detectadas.

Fluxo do Algoritmo

1. **Leitura do CSV**
   O script lê o arquivo CSV linha a linha, separando a chave (ex: 'id') dos demais dados.

2. **Inserção Única no HashMap**
   Cada registro é inserido usando `unique_insert`. Se a chave já existir, é contabilizada como duplicata ou colisão.

3. **Remoção de Duplicatas**
   Apenas o primeiro registro de cada chave é mantido no HashMap.

4. **Resultados**
   O script exibe o DataFrame original, o DataFrame sem duplicatas, e estatísticas de colisões e duplicatas.

Parâmetros

- **size**: Tamanho da tabela hash (número de buckets).
- **hash_function**: Função de hash customizada (opcional).
- **key**: Chave usada para indexação (ex: 'id', 'cpf', etc).

Complexidade de Tempo e Espaço

- **Tempo**:
    - Inserção, busca e remoção: O(1) em média, O(n) no pior caso (todas as chaves colidem).
- **Espaço**:
    - Proporcional ao número de elementos e ao tamanho da tabela hash.

Observações e Limitações

- O tratamento de colisão é feito por **encadeamento externo** (listas em cada bucket).
- O método `unique_insert` não atualiza valores existentes, apenas ignora duplicatas.

Conclusão

O projeto `hash_duplicacao` mostra, de forma prática, como funciona uma tabela hash genérica em Python, incluindo o tratamento de duplicatas e colisões. Permite experimentar diferentes funções de hash e observar na prática como elas afetam o desempenho e a ocorrência de colisões.