Este trabalho apresenta um código que utiliza uma classe de Tabela Hash personalizada, na qual cada posição da tabela contém uma lista de pares (chave, valor). Essa estrutura permite lidar com colisões por meio de encadeamento, evitando sobrescrições e possibilitando armazenar múltiplos elementos na mesma posição, mantendo a eficiência de acesso. Além disso, utilizam-se listas (arrays) para armazenar as posições da tabela e dicionários para facilitar a manipulação dos dados por nome, respectivamente.

O código é organizado nos seguintes módulos e funções:

1. Classe TabelaHash com as principais funções: inserir(chave), remover(chave) e buscar(chave).
2. Operadores especiais: __getitem__, __setitem__ e __delitem__.
3. Método elementos(): iterador sobre os dados únicos armazenados.
4. Função deduplicar_csv: recebe o caminho de um arquivo CSV e uma função chave_func que extrai a chave para deduplicação; insere os elementos na Tabela Hash e retorna apenas os registros únicos.
5. Função menu(): interface interativa via terminal com opções para gerar o dataset, deduplicar, visualizar, buscar e remover dados.

Quanto às complexidades, as operações de inserção, busca e remoção na Tabela Hash têm complexidade média O(1), sendo que no pior caso podem atingir O(n) devido a colisões na mesma posição. O processo de deduplicação usando a Tabela Hash apresenta complexidade temporal O(n), pois o conjunto de dados é percorrido uma única vez, com inserções em tempo constante no melhor cenário. A complexidade espacial é também O(n), para armazenar os registros únicos.

Durante o desenvolvimento, houve um problema ao selecionar a opção 4 no menu e inserir o CPF, onde a aplicação retornava “opção inválida”. Isso ocorreu porque a função input() capturava espaços extras ou quebras de linha, impedindo a comparação correta das opções do menu e das chaves. A aplicação do método .strip() solucionou esse problema.

Em conclusão, o uso da Tabela Hash personalizada para deduplicação mostrou-se uma solução eficiente, com complexidade O(n), superando abordagens tradicionais

baseadas em ordenação. A estrutura permite acesso, inserção e remoção rápidos de registros únicos com facilidade. Além disso, a interface em terminal facilitou testes simples, intuitivos e eficazes para a limpeza de entradas redundantes.