Cuckoo Hashing

1. Objetivo do Sistema

O sistema desenvolvido implementa uma estrutura de **tabela hash com tratamento de colisões baseado em *Cuckoo Hashing***, utilizando duas tabelas distintas (Tabela1 e Tabela2). Ele permite **inserção**, **remoção**, **busca** e **visualização** de chaves inteiras. Além disso, oferece uma saída ordenada das chaves inseridas, indicando a tabela e a posição de cada uma.

---

2. Estrutura Geral

A estrutura central é a **HashTable**, que contém duas tabelas (Tabela1 e Tabela2) com tamanho fixo M (constante definida em hash.h). Cada tabela armazena valores inteiros ou marcadores especiais:

- **VAZIO**: posição nunca usada.
- **REMOVIDO**: posição que teve chave excluída.

---

3. Funções Implementadas

inicializa_hash

- Inicializa ambas as tabelas com o valor VAZIO, preparando a estrutura para uso.

Funções Hash (h1 e h2)

- h1(k): retorna k mod M
- h2(k): usa multiplicação por constante para gerar valor pseudoaleatório, garantindo boa distribuição. h2(k) = ⌊m * (k * 0.9 − ⌊k * 0.9⌋)⌋

busca

- Procura a chave k nas duas tabelas. Retorna a posição onde a chave foi encontrada ou -1 se não existir.

inserir

- Insere a chave k:
- Se Tabela1[h1(k)] estiver livre, insere ali.
- Se houver colisão, move a chave existente para Tabela2[h2(k)] e insere k em Tabela1.
- Não permite duplicatas.
- **Particularidade**: Assume que Tabela2[h2(ki)] estará livre para a chave deslocada ki – **não há tratamento para múltiplas colisões sucessivas**.

remover

- Remove uma chave k, marcando a posição como REMOVIDO. A busca futura ignorará essa posição, mas ela poderá ser reutilizada em futuras inserções.

imprimeHash

- Gera uma lista ordenada de todas as chaves válidas (nem VAZIO nem REMOVIDO) de ambas as tabelas. A ordenação segue a chave crescente, com prioridade para a Tabela 1 em caso de empate.
- Utiliza **Bubble Sort** para ordenação interna dos dados antes da exibição no formato: chave, tabela, posição.

---

4. Execução do Programa

O programa principal lê comandos do tipo:

- i <valor>: insere a chave.
- r <valor>: remove a chave.
- -h: imprime informações do aluno e encerra.
- As operações são lidas da entrada padrão até o fim do arquivo (EOF). Ao final, as chaves válidas são impressas ordenadamente.

---

5. Limitações e Considerações

- O sistema não trata colisões múltiplas consecutivas, o que pode causar sobrescrita em Tabela2 se h2(ki) já estiver ocupada.
- O uso de apenas duas tabelas torna o sistema eficiente, mas menos flexível em casos de grande número de chaves.
- A ordenação via Bubble Sort é simples, mas ineficiente para grandes volumes de dados.
- As funções busca, inserir e remover são eficientes e de fácil entendimento, respeitando a filosofia do Cuckoo Hash com simplicidade.

---

6. Conclusão

O sistema apresenta uma implementação de Cuckoo Hashing. A estrutura em duas tabelas, aliada a funções hash distintas, permite o gerenciamento eficiente de colisões em situações pequenas e controladas.