RELATÓRIO SOBRE A IMPLEMENTAÇÃO DA TABELA HASH DE ENDEREÇAMENTO ABERTO

UNIVERSIDADE FEDERAL DO PARANÁ

Natã Abraão Serafini da Rocha GRR20200183

Victor Ribeiro Garcia GRR20203954

Resumo.

Esse breve relatório descreve o processo criativo e as ideias e procedimentos usados para desenvolver um algoritmo em linguagem C que manipula dados numa estrutura de tabela hash, com a intenção testar e avaliar as habilidades e conhecimentos dos alunos da matéria CI1057 – Algoritmos e estrutura de dados III.

Objetivo do trabalho.

O trabalho final deve apresentar um algoritmo que recebe uma entrada com a operação que deve ser realizada e o valor a ser usado na operação, que ocorrerá em uma das duas tabelas hashs implementadas com funções diferentes, após a leitura de todas as entradas o programa deve entregar uma saída com os valores presentes nas tabelas, também chamados de chaves, em ordem crescente e junto a tabela na qual está inserido e sua posição.

Processo de criação e resultado.

Uma tabela hash é uma estrutura de dados que consiste numa tabela que define onde guardar seus dados através de uma função definida. O trabalho implementa o algoritmo do Cuckoo Hash, o objetivo desse modelo é evitar colisões, que é quando dois valores ao passarem pela função caem no mesmo espaço, logo existem duas tabelas, cada uma com uma função. Assim ao ser notada uma colisão a chave que estava na primeira tabela é incluída na segunda tabela, e a chave que se deseja incluir é colocada na primeira tabela.

Para melhor compreensão e organização do código, criou-se uma biblioteca hash.h com suas funções no arquivo hash.c. Os elementos da tabela tiveram sua implementação por uma struct de nome slot, que contém dois elementos inteiros, um chamado chave, que guarda o valor armazenado, e outro chamado excluído, que será útil na primeira tabela para informar no algoritmo de busca que uma exclusão foi realizada naquele slot. A tabela através de um typedef foi definida como um vetor de structs de tamanho 11, e nome da estrutura Hash.

A primeira função “inicializaHash” preenche a tabela igualando as chaves a -1 e os excluídos a 0. A funções “h1” e “h2” calculam o qual o slot será preenchido na primeira e na segunda tabela respectivamente. A função hash h1 é "h1(k) = k mod m", a segunda h2 possui função hash "h2(k) = ⌊m * (k * 0.9 − ⌊k * 0.9⌋)⌋".

A função “busca” procura o valor em ambas as tabelas e retorna a posição ao encontrar e -1 caso o elemento não seja encontrado em nenhuma das duas tabelas. A função “insere” tenta colocar o valor na primeira tabela, ao encontrar uma colisão insere na segunda tabela o valor que estava na primeira e, assim, insere o valor novo no slot que acabou de ficar vazio na primeira tabela.

A função “exclui” usa a função “busca” que caso só seja encontrado na segunda tabela apenas exclui o valor e caso ache o valor na primeira tabela ele exclui o valor e iguala a variável excluído a 1.

Para impressão algumas etapas foram necessárias, pois é solicitado a impressão em ordem crescente. A solução que se encontrou foi incluir todas as chaves de ambas as tabelas num vetor auxiliar que possui o dobro de tamanho de uma tabela, conseguindo comportar até duas tabelas completas. Após criar esse vetor sua ordenação é realizada através do algoritmo de ordenação QuickSort. Com os valores das chaves do vetor ordenados a função “imprime” chama a função busca, que ao encontrar a chave numa das tabelas, imprime três informações do slot, o valor na chave, a tabela que está e a sua posição, nessa ordem. A biblioteca contendo as funções explicadas anteriormente é incluída no arquivo myht.c que faz a leitura da entrada e chama as funções na ordem correta. O programa final recebe uma entrada no formato especificado e entrega a saída no formato esperado e seus dados corretos o que atende as especificações pedidas.