Relatório Trabalho de Hashing

Bruno Aziz Spring Machado (GRR20211279
Natael Pontarolo Gomes (GRR20211786)
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil

***Resumo*—Trabalho do desenvolvimento na linguagem C de Hashing que foi apresentada em sala.**

I. Introdução

Neste trabalho foi implementado um algotirmo hashing, que é uma técnica de armazenamento e recuperação de dados utilizada em estruturas de dados como tabelas hash, mapas e sets. O objetivo é associar chaves a valores de maneira rápida e eficiente. A ideia é usar uma função hash para mapear as chaves para posições únicas na tabela, de modo que as operações de inserção, busca e remoção possam ser realizadas com tempo de complexidade O(1), em média.

II. Função Main

Nos arquivos entregues, está a função main que inicializa três tabelas hash, t1, t2 e aux. Ela então lê caracteres da entrada padrão até o final do arquivo e, dependendo do caractere lido, realiza uma inserção ou uma remoção na tabela hash. Depois de todas as operações de inserção e remoção terem sido realizadas, a função hash merge é chamada para mesclar as tabelas t1 e t2 em uma tabela auxiliar. Em seguida, a função sort é chamada para classificar a tabela auxiliar e, finalmente, a função imprime hash é chamada para imprimir o conteúdo da tabela auxiliar. Ao final, as tabelas t1 e t2 são destruídas com a função destroi hash e a memória alocada para a tabela auxiliar é liberada.

III. Principais Funções

A. hash_init

A função ”hash_init”é chamada duas vezes para inicializar duas tabelas hash, t1 e t2. Em seguida, uma terceira tabela, aux, é inicializada com o dobro da capacidade das duas primeiras

B. busca_hash

A função ”busca_hash”busca uma chave na tabela hash dupla, retornando seu índice na tabela se encontrada, ou -1 caso contrário. A função ”insere_hash”insere uma chave na tabela hash, utilizando a primeira função de hash (chave%m) para calcular o índice. Se o espaço na tabela estiver ocupado, a segunda função de hash é usada para inserir o elemento na segunda tabela.

C. remove_hash

A função ”remove_hash”remove uma chave da tabela hash dupla, definindo a flag ”removido”como verdadeira para o elemento encontrado. A função ”imprime_hash”imprime os elementos não removidos da tabela hash. A função ”destroi_hash”desaloca a memória alocada para as duas tabelas.

D. hash_merge

A função ”hash_merge”une as duas tabelas em uma única tabela auxiliar, copiando apenas os elementos não removidos.

IV. Arquivo ”sort.c”

Este arquivo contém uma implementação da função de ordenação ”sort”em C. A função sort() realiza a ordenação de um vetor ”tabela_t”a partir do método de ordenação por inserção (Insertion Sort). O método de ordenação por inserção é aplicado através das funções insere() e busca(). A função busca() tem como objetivo encontrar a posição correta onde a estrutura deve ser inserida. A função insere() coloca a estrutura na posição correta, com isso a função sort() realiza a chamada recursiva da função inserção para cada elemento do vetor.

A função troca() tem como objetivo trocar dois elementos do vetor de estruturas. A função sort() retorna 0 caso seja bem sucedida na sua execução.

V. Arquivo makefile

A seção ”all”define o que será construído quando o comando ”make”é executado sem argumentos. Neste caso, o executável ”myht”é construído, já na seção ”$(name): $(objects)”é definido como o executável ”myht”será construído a partir dos arquivos objeto listados na variável ”objects”,as seções ”*.o: *.c”descrevem como cada arquivo objeto será compilado a partir dos arquivos fontes correspondentes, na seção ”clean”é usada para limpar os arquivos objeto gerados durante a compilação, e por fim, a seção ”purge”remove tanto os arquivos objeto quanto o executável

VI. Conclusão

Os testes foram realizados conforme disponibilizados pelo professor na especificação do projeto, com isso este algoritmo implementa uma tabela hash dupla, permitindo a inserção, busca e remoção de elementos em tempo constante, além de resolver colisões com a segunda função de hash. É importante destacar a necessidade de se definir o tamanho m da tabela de maneira adequada para evitar colisões excessivas.