Tipo Abstrato de Dados Hash

Samuel de Souza Gomes

[1]Instituto de Ciências Exatas e Aplicadas – Universidade Federal de Ouro Preto (UFOP)
Rua 36, 115 - Loanda, João Monlevade - MG, 35931-008

samuel.gomes2@aluno.ufop.edu.br

Abstract. *This report aims to use basic programming concepts as well as Abstract Data Types, more specifically, Hash table and linear probing.*

Resumo. *Este relatório tem como objetivo o uso de conceitos básicos de programação bem como os Tipos Abstratos de Dados, mais especificamente, tabelas Hash e linear probing (sonda linear).*

1. Introdução

O programa visa resolver problemas relacionados à tabela hash. Com isso, o código tem funções que permitem a utilização correta da estrutura de dados, que incluem inicialização, inserção, condição de inserção e deleção de elementos.

2. Implementação

Uma das principais diferenças da estrutura de dados Hash Table, é a forma como os elementos são inseridos. Para realizar esse procedimento são utilizadas chaves que serão as posições para os valores inseridos. Porém, os valores não serão apenas inseridos numa chave qualquer, é preciso efetuar um cálculo. O local inserido será: key mod m, onde key = a chave passada e m = tamanho da tabela. Tal operação é feita na função put.

O código em questão ainda utiliza o recurso de sonda linear (linear probing), que, ao verificar que a chave passada já possui algum valor busca pela próxima vazia e insere o elemento. Isso é feito para evitar colisões e poder aproveitar toda a tabela.

Como sabemos, para deletar um elemento de uma tabela hash usando linear probing não basta apenas inserir NULL à chave passada. Portanto, a função del é responsável por deletar o elemento corretamente. Com isso, para deletar é preciso setar um valor que não seja uma possível chave para a tabela (valor negativo caso os valores na tabela sejam todos positivos, por exemplo), com objetivo de ter a possibilidade de inserir um outro elemento naquela posição depois sem perder a chave.

Além dessas funções, temos uma função contadora que serve basicamente para contar quantas chaves o código precisou percorrer até encontrar uma chave vazia. À partir da chave passada por parâmetro, a função começa verificar qual a próxima sonda vazia para poder inserir um valor.

3. Estudo de Complexidade

A implementação típica de uma hash table tem complexidade O(1), pois basta inserir/buscar/deletar um valor através da sua chave, o que evita percorrer toda a tabela. Porém, a estrutura utilizada no trabalho é acrescida da funcionalidade linear probing, ou

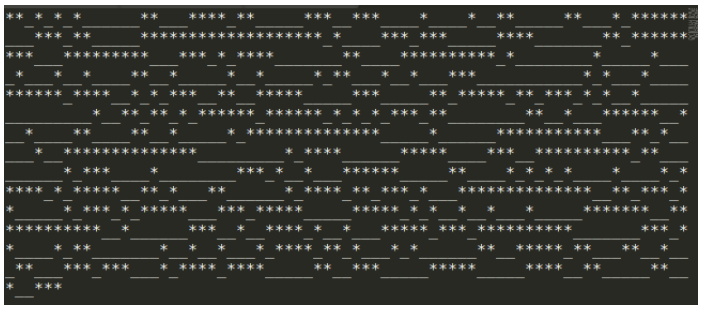
seja, a complexidade vai depender do número de posições disponíveis e do número de chaves inseridas, M e N, respectivamente. A fórmula para calcular tal complexidade é: 1/2*(1 + 1 / 1 - (1 - alfa)), sendo alfa = N/M (fator de carga). Portanto, a complexidade deste método é O(1/2*(1 + 1 / 1 - (1 - alfa))) e pior caso O(n), onde teria que percorrer toda a tabela.

4. Listagem de Testes Executados

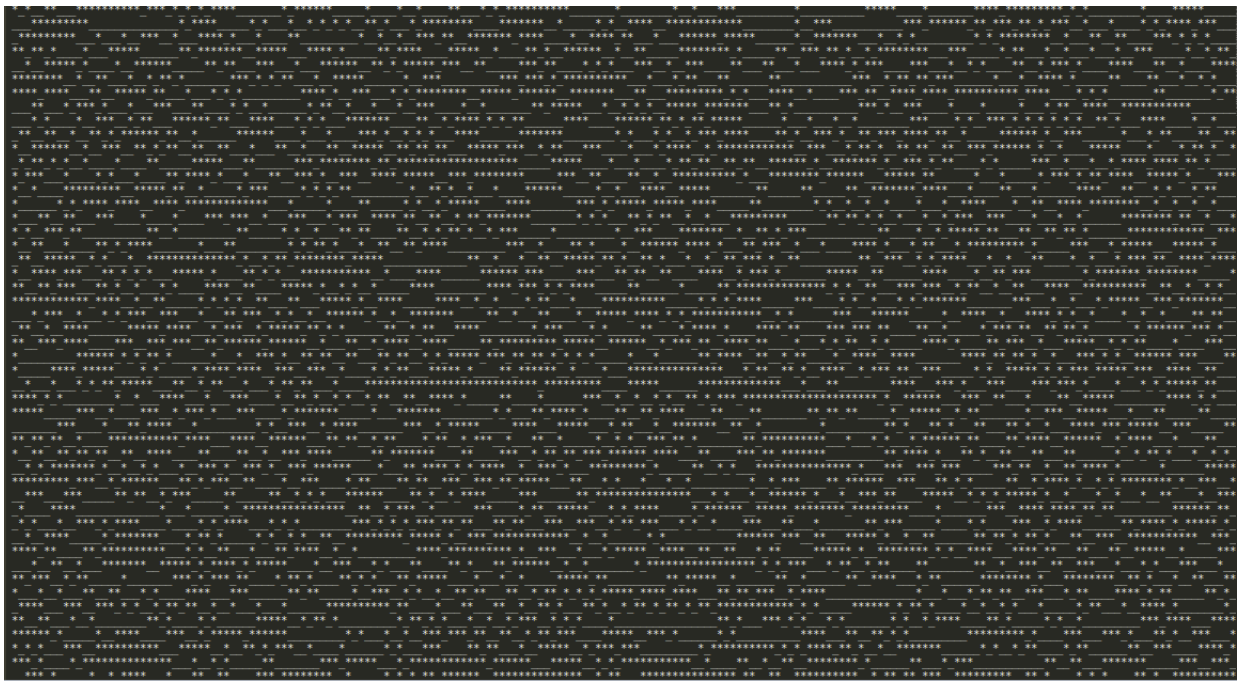
Foram realizados testes apenas com tamanhos 100, 1000 e 10000, pois acima disso o programa já demora executar.

4.1. Tamanho 100

4.2. Tamanho 1000



4.3. Tamanho 10000



5. Conclusão

Tabela hash é uma estrutura de dados especial, que diferente das outras utiliza suas chaves como posições no vetor. Com isso, nota-se uma alta eficiência no custo de pesquisa, porém, o custo é alto quando é preciso recuperar os registros ordenados pela chave, sendo necessário ordenar toda a tabela.

Ao utilizar linear probing, é possível aproveitar melhor a estrutura. Isso devido ao fato de que quando as chaves de elementos distintos são a mesma, o próximo elemento é ”setado” para a próxima posição/chave vazia na tabela. Assim, os valores ficam melhor distribuídos, principalmente, quando o tamanho da tabela é suficientemente grande.

6. Referências

WIKIPÉDIA. **Algoritmos e Estruturas de Dados/Tabela de Hash**. Disponível em: https://pt.wikibooks.org/wiki/Algoritmos-e-Estruturas-de-Dados/Tabela-de-Hash. Acesso em: 5 jul. 2019.

GEEKS FOR GEEKS. **Implementing own Hash Table with Open Addressing Linear Probing in C++**. Disponível em: https://www.geeksforgeeks.org/implementing-hash-table-open-addressing-linear-probing-cpp/. Acesso em 5 jul. 2019.