Análise experimental da tabela hash com endereçamento fechado e árvore AVL

Allan Cordeiro Rocha de Araújo.

[1]Centro de ciência e tecnologia– Universidade Federal de Roraima (UFRR)
Boa Vista– RR – Brasil

allanps32008@gmail.com

Resumo. *Será feito uma análise da comparação do tempo de execução dos algoritmos implementados com tabela hash com endereçamento fechado e com árvore AVL.*

1. Introdução

O objetivo aqui é, usando várias entradas de vários tamanhos, vê qual tipo de tabela hash é mais eficiente com a operação de inserção, a tabela hash com lista encadeada ou com árvore AVL.

2. Tabela hash

Também conhecida como tabela de dispersão, é uma estrutura de dados que atribui chaves aos respectivos valores, usando essas chaves é calculado sua posição na lista, com o objetivo de fazer buscas rápidas (em tese com custo O(1)) ao custo de mais memória usada para alocação de espaço.

Ela é usada bastante em bancos de dados, onde a quantidade de dados é enorme e a busca rápida pela informação é algo tido como prioridade.

A função de espalhamento é a principal característica da tabela hash, ela quem irá calcular a posição daquele elemento na tabela. O desafio aqui é o tratamento de colisões, que acontece quando dois elementos são calculados para a mesma posição da lista, existem diversas técnicas para esse tratamento, seja por endereçamento aberto ou encadeamento.

3. Árvore AVL

Árvores AVL são árvores balanceadas de busca, nela a diferença de altura entre um nó folha e outro é no máximo 1. Sua principal característica é seu alto balanceamento, onde a cada inserção é feita rotações em seus nós para balanceá-la novamente, o custo dessa operação de balanceamento é O(n).

Uma árvore binária T é dita AVL quando, para qualquer nó v de T, a diferença de altura das subárvores esquerda He(v) e direita Hd(v) é no máximo igual a 1.

Fator de balanceamento: FB(v) = He(v) - Hd(v). Onde FB = -1 ou FB = 1 ou FB = 0.

As operações de inserção, busca e remoção são O(n).

5. Análise do tempo de execução

Os testes foram realizados em uma distro do Linux. Intel core i7, geforce GTX 960m, 8gb de RAM, Ubuntu versão 16.04. Todos os testes foram executados 7 vezes e foi retirado a média entre o tempo de execução.

Foi usado a linguagem C++, porém não foi usado qualquer biblioteca que auxiliasse na criação das estruturas de dados. Para o cálculo de tempo foi usado variáveis do tipo clock_t e para coleta de tempo a função clock(), que usa os ciclos do processador para a obtenção de tal.

Tabela hash com endereçamento fechado. A primeira linha é a quantidade de elementos inseridos na tabela e segunda linha o tempo gasto, em segundos, para inserir todos estes elementos.

Observação: o tamanho da tabela, que também é usado para o cálculo hashing, foi de 36131, que por sinal é um número primo.

100	1000	10000	100000	1000000	5000000	8000000	10000000
0,0000006	0,0000055	0,00045	0,006522	0,138113	5,859	17,232	27,52

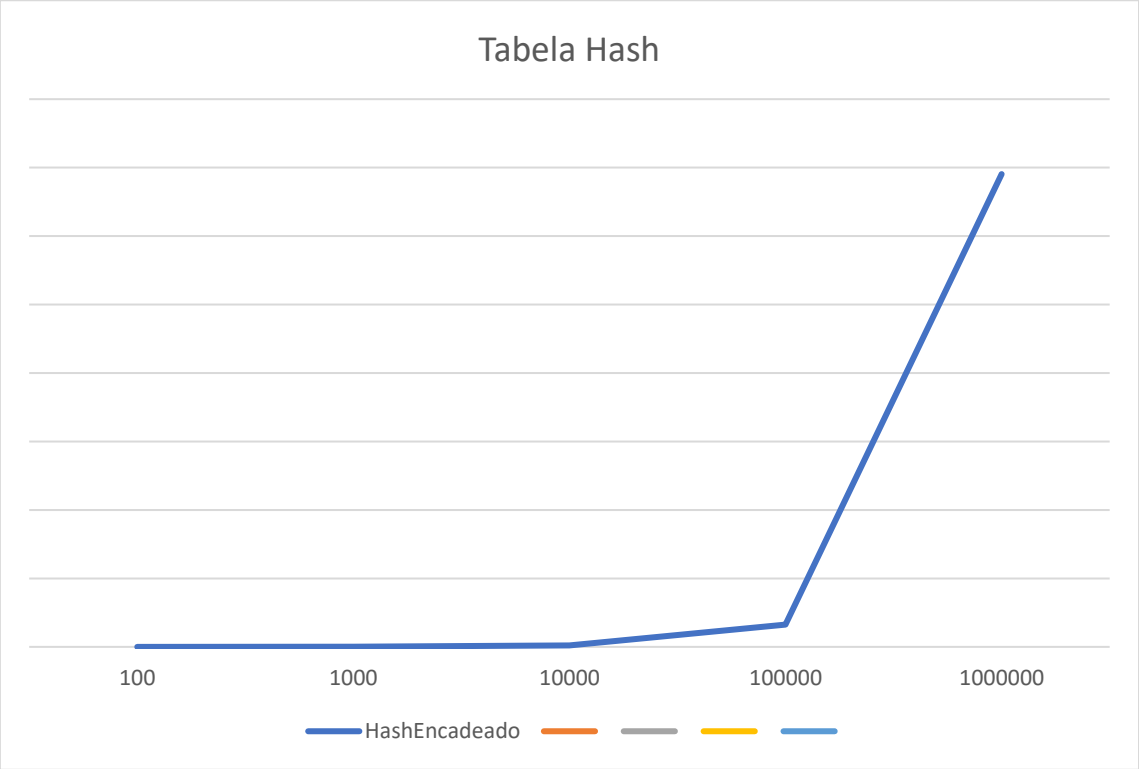

Gráfico com a quantidade de valores inseridos e o tempo gasto.

Percebe-se que até mesmo uma tabela hash leva um certo tempo para inserir uma quantidade absurda de elementos, por exemplo, leva em torno de 27 segundos para inserir 10 milhões de elementos.

Referências

Hashing”; IME; Disponível em <https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/st-hash.html>. Acesso em 12 de julho de 2018.

Árvores AVL”; Prof. Robinson Alves; Disponível em <https://docente.ifrn.edu.br/robinsonalves/disciplinas/estruturas-de-dados-nao-lineares/arvoreAVL.pdf>. Acesso em 12 de julho de 2018.