Análise de Desempenho de Tabelas Hash com Diferentes Funções de Hash

Matheus Moreira - Ricardo Moreira

November 1, 2024

1 Introdução

As tabelas hash são estruturas de dados amplamente utilizadas devido à sua capacidade de fornecer acesso rápido a elementos. Este relatório apresenta uma análise do desempenho de uma tabela hash implementada em Java, utilizando três diferentes funções de hash: resto da divisão, multiplicação e dobramento. O objetivo é comparar o desempenho em termos de tempo de inserção, tempo de busca, número de colisões e total de comparações realizadas durante as operações.

2 Metodologia

O código foi desenvolvido em Java, utilizando uma tabela hash com listas encadeadas para gerenciar colisões. As funções de hash implementadas incluem:

- **Resto da Divisão:** Usa o resto da divisão do código pelo tamanho da tabela.
- **Multiplicação:** Baseia-se na constante de Knuth e calcula o índice como uma função da parte fracionária do produto do código pela constante.
- **Dobramento:** Soma os dígitos do código e usa o resultado para calcular o índice.

Foram gerados dados aleatórios representando códigos de registro, e o desempenho foi avaliado para tabelas hash de tamanhos 1000, 2000 e 5000, com quantidades de registros variando de 10.000 a 50.000.

Os resultados foram registrados em um arquivo CSV para posterior análise e construção gráfica.

3 Resultados

Os resultados obtidos foram compilados em tabelas, apresentando os tempos de inserção, número de colisões, comparações em buscas e tempo médio de busca.

3.1 Tabela 1: Resultados para Tabela Hash de Tamanho 1000

Table 1: Resultados para Tabela Hash de Tamanho 1000

Quantidade	Função de Hash	Tempo de Inserção (ns)	Colisões	Comparações	Tempo Médio de Busca
10.000	Resto	13.387.000	9.000	300.280	4.658.880
10.000	Multiplicação	12.610.700	9.000	298.905	9.522.640
10.000	Dobramento	37.418.600	9.941	8.165.175	30.608.220
50.000	Resto	24.094.900	49.000	6.492.145	22.930.600
50.000	Multiplicação	29.126.900	49.000	6.503.575	23.015.860
50.000	Dobramento	445.513.000	49.935	203.263.455	417.031.220
20.000	Resto	3.263.800	19.000	1.100.290	2.711.720
20.000	Multiplicação	3.851.800	19.000	1.097.555	4.008.120
20.000	Dobramento	57.276.000	19.938	32.432.145	60.252.200

3.2 Tabela 2: Resultados para Tabela Hash de Tamanho 2000

Table 2: Resultados para Tabela Hash de Tamanho 2000

Quantidade	Função de Hash	Tempo de Inserção (ns)	Colisões	Comparações	Tempo Médio de Busca
10.000	Resto	984.100	8.010	174.575	672.900
10.000	Multiplicação	1.505.400	8.013	174.360	1.061.460
10.000	Dobramento	11.787.800	9.941	8.165.175	11.532.240
50.000	Resto	11.761.500	48.000	3.375.070	10.060.600
50.000	Multiplicação	16.676.100	48.000	3.379.840	14.852.340
50.000	Dobramento	443.214.700	49.935	203.263.455	445.554.380
20.000	Resto	2.695.900	18.000	601.150	1.938.440
20.000	Multiplicação	4.536.100	18.001	599.260	3.397.480
20.000	Dobramento	58.253.200	19.938	32.432.145	59.997.100

3.3 Tabela 3: Resultados para Tabela Hash de Tamanho 5000

Table 3: Resultados para Tabela Hash de Tamanho 5000

Quantidade	Função de Hash	Tempo de Inserção (ns)	Colisões	Comparações	Tempo Médio de Busca
10.000	Resto	3.902.200	5.673	99.560	595.220
10.000	Multiplicação	1.308.700	5.677	99.775	869.180
10.000	Dobramento	10.276.600	9.941	8.165.175	10.128.320
50.000	Resto	6.009.500	45.001	1.496.205	5.845.080
50.000	Multiplicação	10.248.900	45.000	1.501.640	8.137.940
50.000	Dobramento	439.944.700	49.935	203.263.455	443.108.200
20.000	Resto	1.915.600	15.094	299.105	1.394.740
20.000	Multiplicação	2.958.500	15.080	299.065	2.203.180
20.000	Dobramento	56.687.400	19.938	32.432.145	57.184.440

4 Análise

Os resultados mostram variações significativas no desempenho das funções de hash utilizadas. Em geral, a função de hash por resto da divisão apresenta o melhor desempenho em termos de tempo de inserção e busca, enquanto a função de hash por dobramento resulta em tempos muito maiores, especialmente com grandes quantidades de registros.

A tabela hash de tamanho 1000 teve o maior número de colisões em quase todos os casos, sugerindo que um tamanho maior da tabela pode melhorar o desempenho, reduzindo o número de colisões.

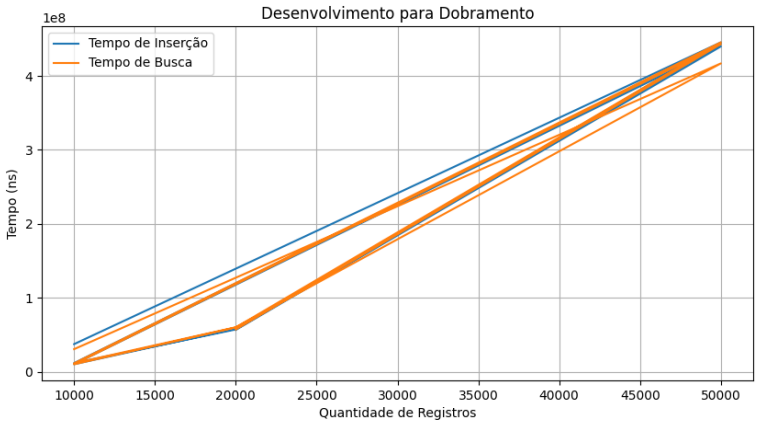

Figure 1: Gráfico Desenvolvimento para Dobramento

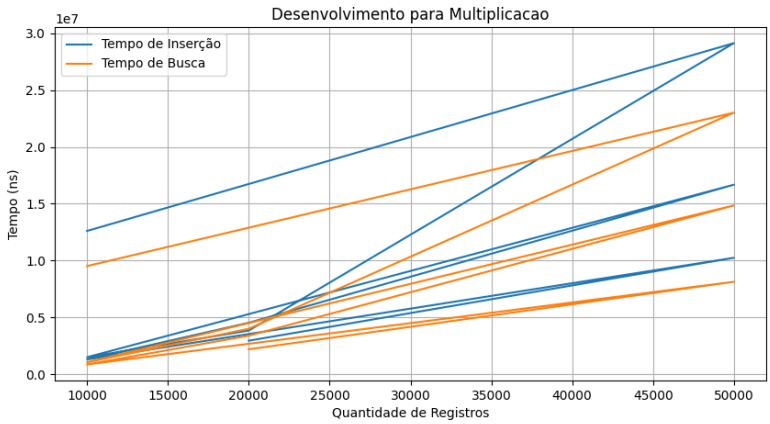

Figure 2: Gráfico Desenvolvimento para Multiplicação

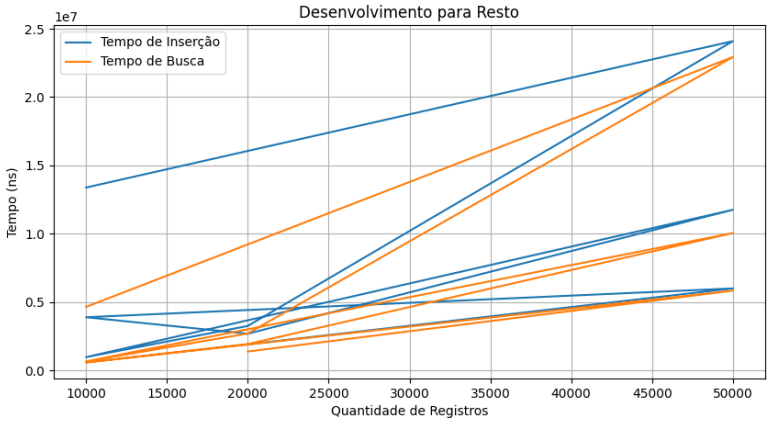

Figure 3: Gráfico Desenvolvimento para Resto