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 200.000.

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	6.887.000	9.000	300.280	3.248.200
10.000	Multiplicação	11.785.100	9.000	298.905	5.724.360
10.000	Dobramento	136.180.100	9.941	8.165.175	15.005.900
50.000	Resto	27.671.800	49.000	6.492.145	17.354.800
50.000	Multiplicação	28.926.500	49.000	6.503.575	28.265.440
50.000	Dobramento	409.774.500	49.935	203.263.455	402.254.720
200.000	Resto	339.382.700	198.982	100.998.075	316.625.420
200.000	Multiplicação	608.806.300	198.982	100.968.590	673.591.340
200.000	Dobramento	8.481.320.900	199.912	3.247.650.770	8.171.041.660

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	507.800	8.010	174.575	549.820
10.000	Multiplicação	884.600	8.013	174.360	953.040
10.000	Dobramento	10.179.300	9.941	8.165.175	10.007.620
50.000	Resto	8.823.500	48.000	3.375.070	8.965.720
50.000	Multiplicação	11.228.800	48.000	3.379.840	12.139.740
50.000	Dobramento	401.346.600	49.935	203.263.455	407.773.720
200.000	Resto	146.295.000	197.982	51.010.200	149.534.200
200.000	Multiplicação	274.297.600	197.982	51.000.430	273.522.700
200.000	Dobramento	7.797.390.900	199.912	3.247.650.770	7.932.543.820

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	395.700	5.673	99.560	426.260
10.000	Multiplicação	715.500	5.677	99.775	788.080
10.000	Dobramento	9.611.900	9.941	8.165.175	9.832.060
50.000	Resto	4.657.800	45.001	1.496.205	4.816.800
50.000	Multiplicação	6.580.500	45.000	1.501.640	7.282.840
50.000	Dobramento	396.987.600	49.935	203.263.455	404.628.660
200.000	Resto	63.430.900	194.982	21.000.700	65.898.680
200.000	Multiplicação	120.750.500	194.982	20.990.770	130.887.860
200.000	Dobramento	7.788.623.800	199.912	3.247.650.770	7.865.838.940

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.

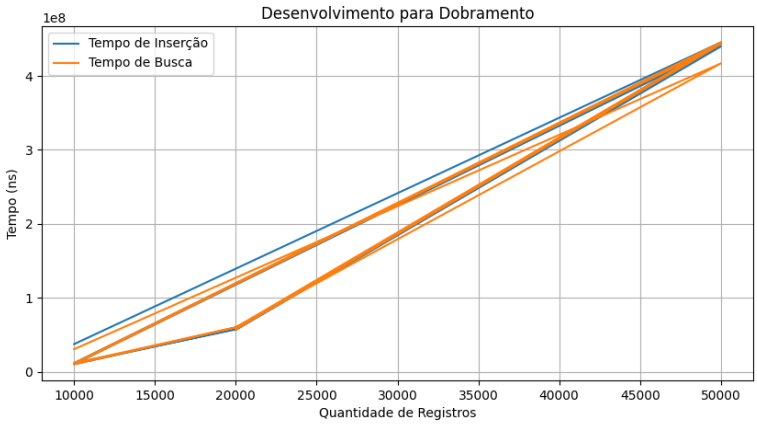

Figure 1: Gráfico Desenvolvimento para Dobramento

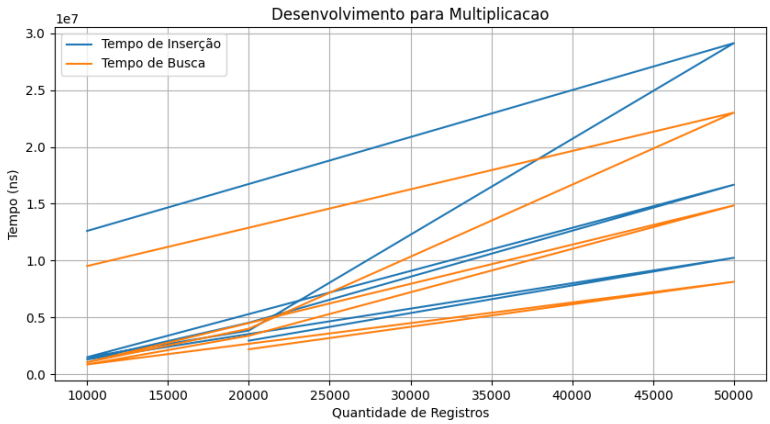

Figure 2: Gráfico Desenvolvimento para Multiplicação

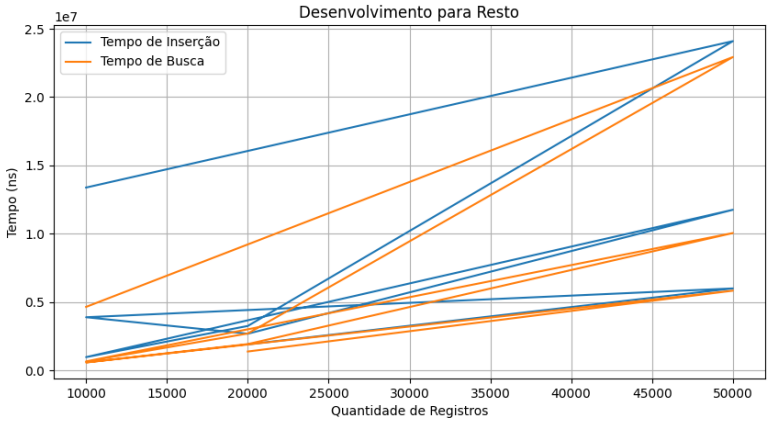

Figure 3: Gráfico Desenvolvimento para Resto