Tabela Hash
Avaliação RA3

Alexandre Cícero Araujo Beiruth, João Vitor Gabardo da Cunha, Leandro Cardoso Vieira, Luana Tiemann Halicki Cordeiro.

[1]Escola Politécnica - Pontifícia Universidade Católica do Paraná (PUCPR).

1. Introdução

Este relatório apresenta uma análise de desempenho para métodos distintos de funções de hash. O objetivo é implementar e avaliar a eficácia de cada função de hash com base em sua capacidade de distribuir dados uniformemente sem colisões, bem como sua eficiência em termos de tempo de inserção e busca.

2. Métodos

- Foi escolhido usar rehashing devido a ser uma implementação menor em relação a lista encadeada, além disso, já tínhamos um esboço de código de busca fornecido pelo professor.
- Os 5 tamanhos de vetores da tabela hash foram de 5.500.000, 10.000.000, 20.000.000, 30.000.000, 40.000.000, sendo eles maiores do que os dados como exemplo, para evitar problemas com tabelas cheias.
- As 3 variações da função hash escolhidas foram resto da divisão e multiplicação seguindo a sugestão do professor, e XOR no lugar da sugestão de dobramento, visando diminuir o tempo de execução.
- Geração de dados foi feita aleatoriamente usando Seeds com tamanhos de 20 mil, 100 mil, 500 mil, 1 milhão e 5 milhões elementos cada, e cada elemento sendo um objeto da classe registro contendo o código do registro com 9 dígitos.
- O código busca e insere os elementos de cada conjunto de dados em cada tabela hash, usando cada função hash, medindo seus tempos e números de colisões para cada combinação, sendo realizadas ao menos 5 buscas e inserções para avaliar os tempos.

3. Gráficos e Tabelas

Os resultados foram convertidos para um arquivo Excel para a criação de tabelas e gráficos para auxiliar na apresentação da análise do desempenho de diferentes tabelas hash e funções hash. Abaixo segue os gráficos e tabelas criados:

3.1. Colisões

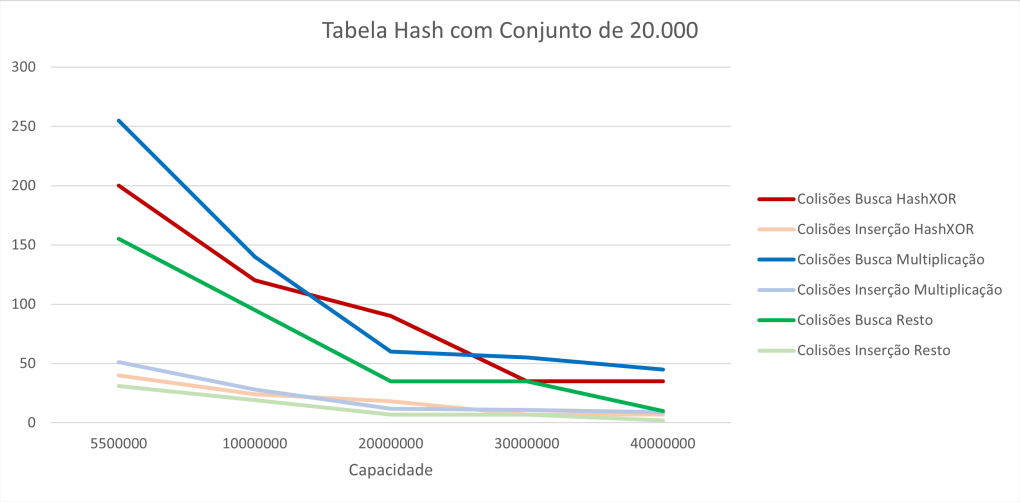

Figure 1. Gráfico mostrando o desempenho da tabela hash em termos de colisões na inserção e busca com 20 mil elementos para todas as funções hash utilizadas

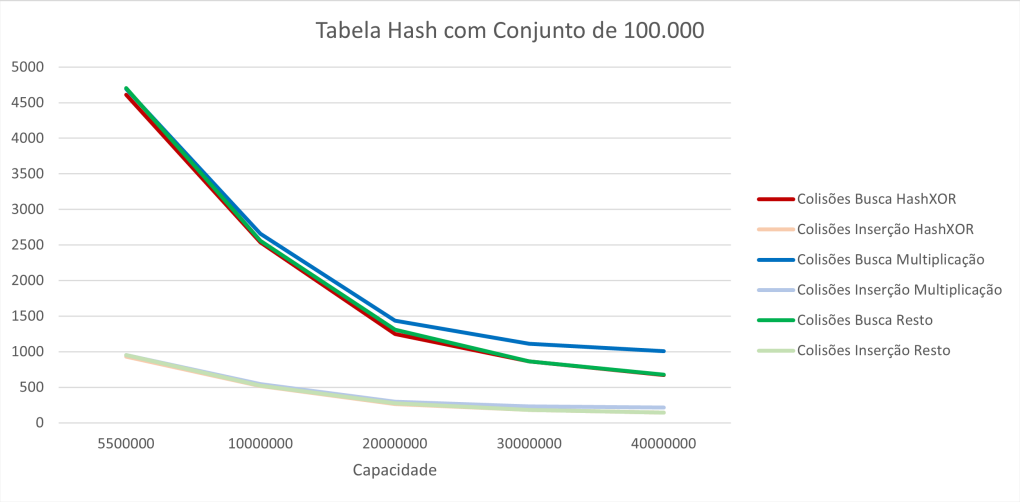

Figure 2. Gráfico mostrando o desempenho da tabela hash em termos de colisões na inserção e busca com 100 mil elementos para todas as funções hash utilizadas

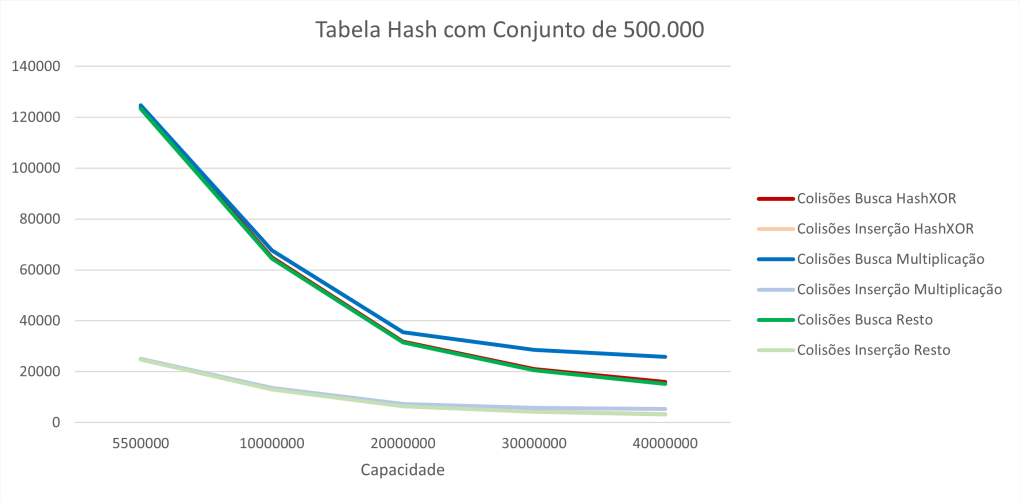

Figure 3. Gráfico mostrando o desempenho da tabela hash em termos de colisões na inserção e busca com 500 mil elementos para todas as funções hash utilizadas

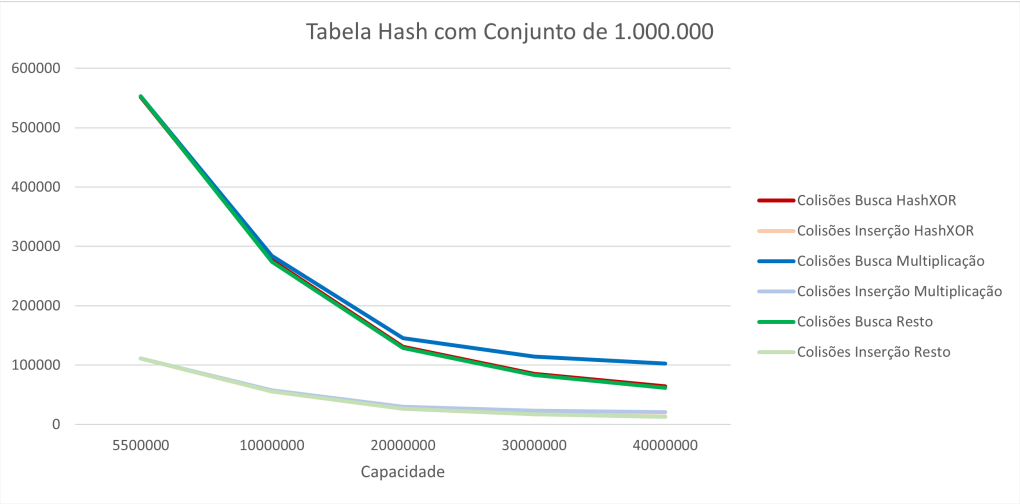

Figure 4. Gráfico mostrando o desempenho da tabela hash em termos de colisões na inserção e busca com 1 milhão de elementos para todas as funções hash utilizadas

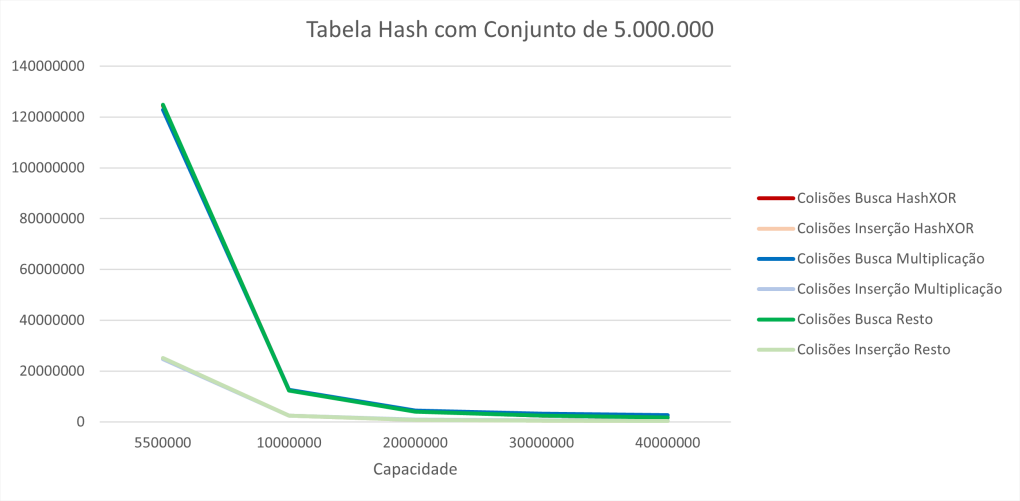

Figure 5. Gráfico mostrando o desempenho da tabela hash em termos de colisões na inserção e busca com 5 milhões de elementos para todas as funções hash utilizadas

3.2. Tempo

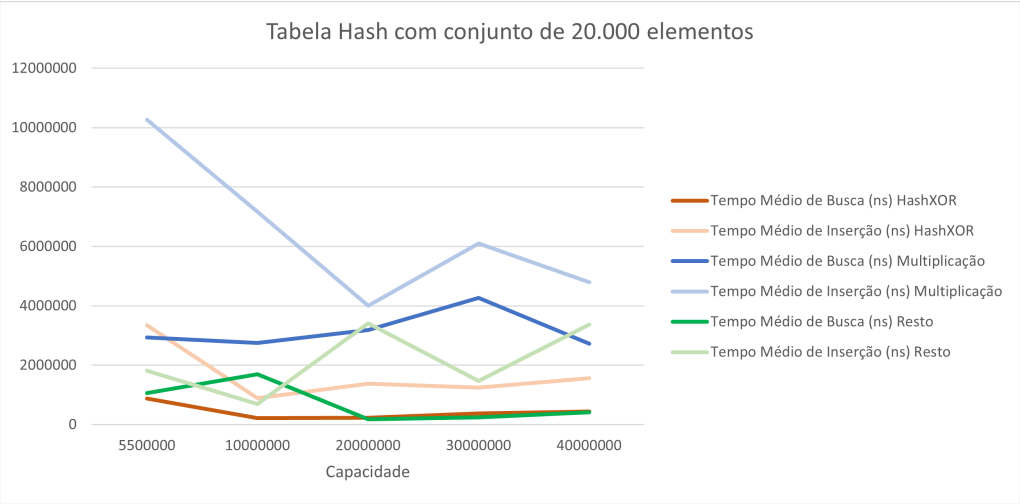

Figure 6. Gráfico mostrando o desempenho da tabela hash em termos de tempo na inserção e busca com 20 mil elementos para todas as funções hash utilizadas

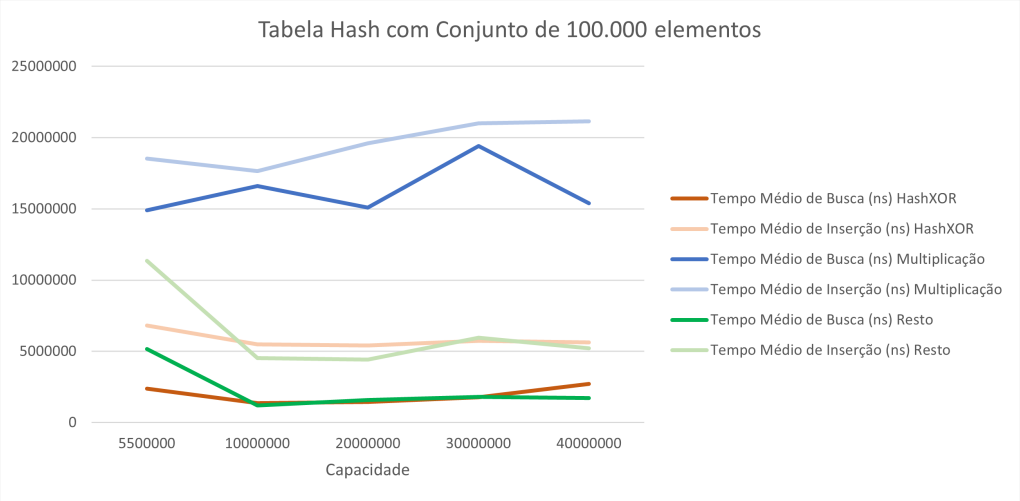

Figure 7. Gráfico mostrando o desempenho da tabela hash em termos de tempo na inserção e busca com 100mil elementos para todas as funções hash utilizadas

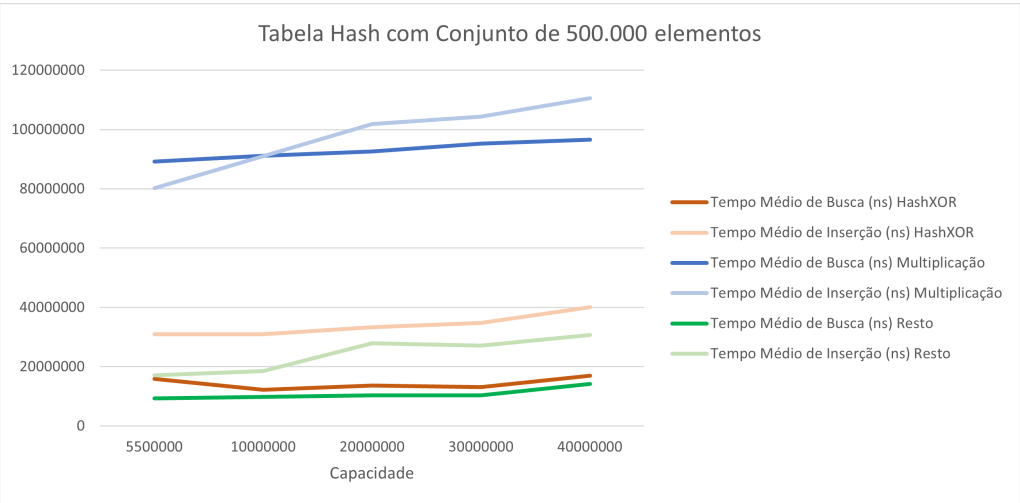

Figure 8. Gráfico mostrando o desempenho da tabela hash em termos de tempo na inserção e busca com 500mil elementos para todas as funções hash utilizadas

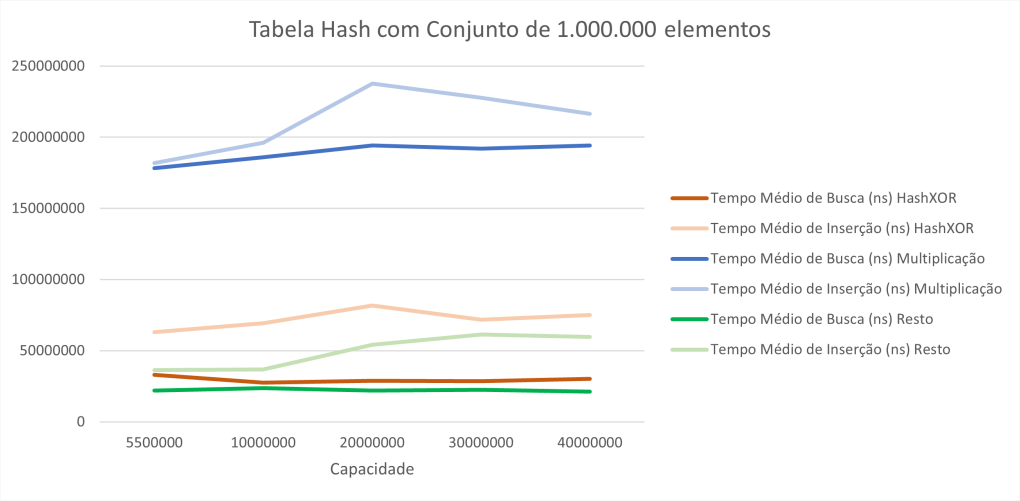

Figure 9. Gráfico mostrando o desempenho da tabela hash em termos de tempo na inserção e busca com 1 milhão de elementos para todas as funções hash utilizadas

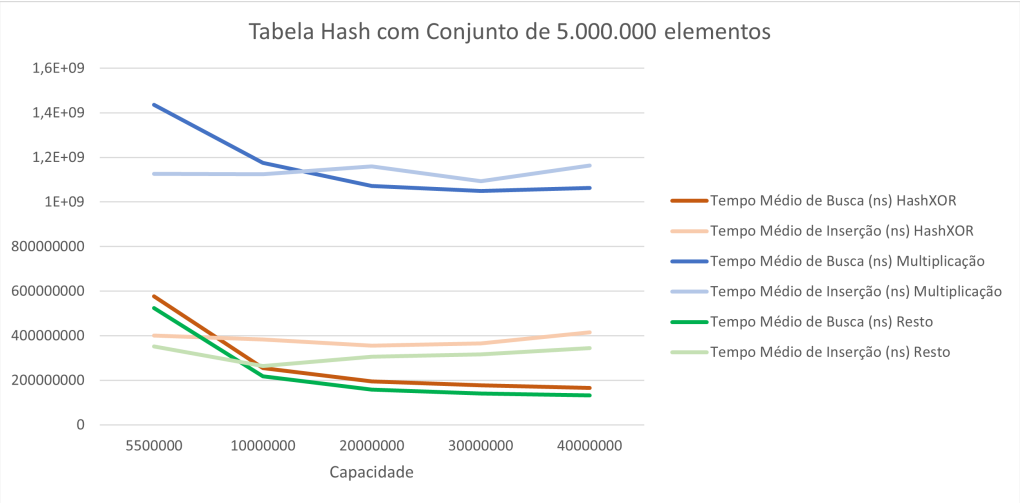

Figure 10. Gráfico mostrando o desempenho da tabela hash em termos de tempo na inserção e busca com 5 milhões de elementos para todas as funções hash utilizadas

3.3. Tabela

Table 1. Tabela de saída do código com os dados de Capacidade, Método, Conjunto, Colisões Inserção, Colisões Busca, Tempo de Inserção (ns), Tempo Médio de Busca (ns)

Capacidade	Método	Conjunto	Colisões Inserção	Colisões Busca	Tempo de Inserção (ns	Tempo Médio de Busca (ns
5500000	hashXOR	20000	40	200	3341700	872740
5500000	hashXOR	100000	933	4610	6809500	2377760
5500000	hashXOR	500000	24938	123955	30985600	15907820
5500000	hashXOR	1000000	111032	551960	63120000	33145180
5500000	hashXOR	5000000	25052893	124482160	400073300	576869340
10000000	hashXOR	20000	24	120	896600	215500
10000000	hashXOR	100000	519	2535	5483300	1367980
10000000	hashXOR	500000	13115	64885	30955700	12259960
10000000	hashXOR	1000000	55959	276865	69383200	27460700
10000000	hashXOR	5000000	2522210	12469980	383507500	255837680
20000000	hashXOR	20000	18	90	1379900	230760
20000000	hashXOR	100000	262	1250	5417500	1449800
20000000	hashXOR	500000	6514	31885	33352500	13583100
20000000	hashXOR	1000000	26749	130955	81698800	28933700
20000000	hashXOR	5000000	848182	4148215	355551500	195343700
30000000	hashXOR	20000	7	35	1249100	371100
30000000	hashXOR	100000	184	865	5747300	1761980
30000000	hashXOR	500000	4329	20975	34751600	13137700
30000000	hashXOR	1000000	17626	85435	71819100	28466400
30000000	hashXOR	5000000	510436	2469260	365959900	177966500
40000000	hashXOR	20000	7	35	1566200	436720
40000000	hashXOR	100000	145	670	5617000	2725060
40000000	hashXOR	500000	3331	15980	40129300	16968380
40000000	hashXOR	1000000	13364	64095	74949000	30336260
40000000	hashXOR	5000000	368042	1761245	415445700	165935480
5500000	multiplicacao	20000	51	255	10269000	2933640
5500000	multiplicacao	100000	950	4695	18536600	14898180
5500000	multiplicacao	500000	25101	124760	80216700	89191400
5500000	multiplicacao	1000000	111209	552830	181809900	178410080
5500000	multiplicacao	5000000	24723221	122845480	1126169700	1436196200
10000000	multiplicacao	20000	28	140	7164400	2746500
10000000	multiplicacao	100000	542	2655	17647300	16593980
10000000	multiplicacao	500000	13675	67690	31054500	31220260
10000000	multiplicacao	1000000	57471	284425	196032800	185822360
10000000	multiplicacao	5000000	2548401	12603865	1124885000	1176228060
20000000	multiplicacao	20000	12	60	4004500	3179260
20000000	multiplicacao	100000	298	1435	19593100	15086160
20000000	multiplicacao	500000	7253	35580	101858300	92662720
20000000	multiplicacao	1000000	29658	145505	237711600	194383300
20000000	multiplicacao	5000000	922815	4521920	1160554400	1071932780
30000000	multiplicacao	20000	11	55	6100300	4265720
30000000	multiplicacao	100000	233	1110	21017500	19404380
30000000	multiplicacao	500000	5859	28610	104415200	95219340
30000000	multiplicacao	1000000	23490	114725	227683900	191911440
30000000	multiplicacao	5000000	654124	3188415	1093766100	1050310020
40000000	multiplicacao	20000	9	45	4799400	2719340
40000000	multiplicacao	100000	213	1010	21136500	15388200
40000000	multiplicacao	500000	5308	25865	110598900	96637800
40000000	multiplicacao	1000000	21129	102950	216401700	194263200
40000000	multiplicacao	5000000	560967	2726670	1164202100	1062414720
5500000	resto	20000	31	155	1807600	1065400
5500000	resto	100000	953	4705	11351000	5171440
5500000	resto	500000	24809	123295	17149000	9257400
5500000	resto	1000000	111201	552840	36277600	22039260
5500000	resto	5000000	25131900	124877950	351937500	524779840
10000000	resto	20000	19	95	692900	1696360
10000000	resto	100000	524	2560	4514600	1201060
10000000	resto	500000	13058	64550	18548100	9822840
10000000	resto	1000000	55461	274290	36973100	23677560
10000000	resto	5000000	2499937	12361545	263411900	218909240
20000000	resto	20000	7	35	3408800	184020
20000000	resto	100000	273	1310	4410400	1573040
20000000	resto	500000	6442	31525	27955800	10390400
20000000	resto	1000000	26352	128935	54041800	21988500
20000000	resto	5000000	834701	4081275	305376200	157864260
30000000	resto	20000	7	35	1470500	239300
30000000	resto	100000	184	865	5945300	1801340
30000000	resto	500000	4252	20565	27147600	10337100
30000000	resto	1000000	17259	83525	61491400	22443320
30000000	resto	5000000	499864	2416280	316118200	140426880
40000000	resto	20000	2	10	3371000	410260
40000000	resto	100000	146	675	5226700	1713480
40000000	resto	500000	3171	15180	30673800	14143100
40000000	resto	1000000	12936	61965	59703900	21284700
40000000	resto	5000000	358067	1711400	345317700	132161960

4. Análise

4.1. Colisões

As colisões são usadas na avaliação da eficácia de uma função de hash, pois afetam a velocidade de busca e inserção de dados. Uma colisão ocorre quando dois valores diferentes produzidos pela função hash apontam para o mesmo local na tabela, exigindo resolução adicional.

4.1.1. HashXOR

- Eficaz na diminuição de colisões em todos os tamanhos de conjuntos testados.
- Em conjuntos menores, as colisões foram notavelmente inferiores em comparação com os outros métodos.
- Mesmo com o aumento da capacidade da tabela, o número de colisões permaneceu baixo, mostrando sua eficiência.

4.1.2. Resto da Divisão

- Em conjuntos menores, teve uma performance comparável ao hashXOR, mas sua eficiência diminuiu em conjuntos maiores.
- A expansão da capacidade da tabela hash melhorou o desempenho, sugerindo uma sensibilidade ao tamanho da tabela.

4.1.3. Multiplicação

- Demonstrou o maior número de colisões, principalmente em tamanhos de conjunto maiores.
- Indica uma distribuição menos uniforme dos dados pela tabela hash, resultando em eficiência reduzida.

4.2. Tempo de Inserção e Busca

O tempo de processamento é outro indicador importante no desempenho da função de hash, mostrando a rapidez com que os dados podem ser inseridos e buscados da tabela.

4.2.1. HashXOR

- Apresentou os tempos de inserção e busca mais rápidos em quase todos os conjunto de dados.
- A eficiência apenas diminuiu levemente em conjuntos de dados extremamente grandes.

4.2.2. Resto da divisão

- Têm tempos competitivos de inserção e busca em conjuntos menores.

- Com o aumento do tamanho do conjunto, o tempo de inserção aumentou, perdendo eficiência.
- A busca foi menos eficiente em comparação com o hashXOR em conjuntos grandes.

4.2.3. Multiplicação

- Resultou nos maiores tempos de inserção e busca.
- A eficiência diminuiu significativamente à medida que o conjunto de dados cresceu, o tornando menos adequado para tabelas hash grandes.

5. Conclusão

O desempenho das funções de hash foi avaliado com base no número de colisões e no tempo de inserção e busca. O hashXOR provou ser o mais eficiente, com menos colisões e tempos rápidos em várias condições. O resto da divisão é uma boa opção para conjuntos menores, mas sua eficiência diminui com conjuntos maiores. O método de multiplicação foi o menos eficiente em tabelas maiores. Portanto, o hashXOR é a escolha preferida para tabelas hash de propósito geral, enquanto os outros métodos podem exigir ajustes ou ser adequados para casos de uso específicos.