Tabela Hash

Victor Silva Camargo, Vin´ıcius Silva Camargo

Pontifícia Universidade Católica do Paraná´
(PUCPR) Curitiba – PR – Brazil

Abstract. *A hash function is an algorithm that transforms data into a fixed-length sequence of characters, known as a ”hash” or ”fingerprint.” These func- tions are used to optimize data retrieval, protect data integrity, and accelerate algorithms. They play a crucial role in data structures like hash tables and in cryptography. The choice or design of hash functions is critical to avoid colli- sions and ensure security and efficiency.*

Resumo. *Uma função de hash e´ um algoritmo que transforma dados em uma seque^ncia de caracteres de tamanho fixo, conhecida como ”hash” ou ”impressão digital”. Essas funções são usadas para otimizar a pesquisa de dados, proteger a integridade de informações e acelerar algoritmos. Elas desempenham um papel essencial em estruturas de dados como tabelas de dispersão (hash tables) e em criptografia. A escolha ou design de funções de hash é crucial para evitar colisões e garantir a segurança e eficiência.*

1. Hash com Lista encadeada

Para o nosso trabalho escolhemos o encadeamento porque tentamos usar o hiraishin mas estava dando problemas de mais, sobre a lista encadeada, a lista encadeada com hash e uma estrutura de dados que combina características de uma lista encadeada é uma tabela de dispersão. Ela é usada para armazenar e recuperar informações eficientemente, onde cada elemento é associado a uma chave única, calcula-se um valor de hash a partir dessa chave e usa-o para determinar a posição do elemento na estrutura. As colisões, quando várias chaves te^m o mesmo valor de hash, são resolvidas através de listas encadeadas associadas a cada slot da tabela de dispersão. A eficie^ncia depende da função de hash e do dimensionamento adequado da tabela. Essa estrutura é comum em dicionários, bancos de dados e sistemas de indexação.

2. Estrutura do Código

A princípio a estrutura do código foi dividida por conjunto de dados tendo ao todo tendo 5 códigos executáveis, onde cada um deles usa 3 func¸o˜es hash diferentes para inserir o conjunto de dados nas diferentes tabelas hash, o projeto possui também 3 códigos que implementam as 3 func¸o˜es hashs diferentes, todas as 3 usam o no´ e lista encadeada.

3. Resultados

Executamos o código com diferentes tamanhos do vetor sendo 10, 100, 1000, 10000 e 100000 com cinco conjuntos de dados aleatórios com 20 mil, 100 mil, 500 mil, 1 milhão e 5 milhões de elementos cada, na tabela tem as letras representam os métodos sendo I para inserção B para Busca, e todos os valores de tempo estão em milisegundos.

I	10	100	1000	10000	100000
Divisão Tempo	3	1	1	1	3
Divisão Colisões	19990	19900	19000	11376	1780
Multiplicação Tempo	3	1	1	2	4
Multiplicação Colisões	19990	19900	19000	11364	1922
Dobramento Tempo	2	2	1	1	2
Dobramento Colisões	19990	19941	19941	19941	19941
B					
Divisão Tempo	3883	517	137	121	153
Divisão Comparações	13341279	1344040	143300	23270	11339
Multiplicação Tempo	3779	518	189	187	207
Multiplicação Comparações	13341989	1343605	143245	23246	11340
Dobramento Tempo	4162	1454	1423	1377	1362
Dobramento Comparações	13341252	4368960	4368960	4368960	4368960

Figura 1. Tabela hash com 20 mil elementos

I	10	100	1000	10000	100000
Divisão Tempo	7	8	22	1	3
Divisão Colisões	99990	99900	99000	90000	36854
Multiplicação Tempo	6	5	49	6	5
Multiplicação Colisões	99990	99900	99000	90001	36845
Dobramento Tempo	6	5	2	2	2
Dobramento Colisões	99990	99934	99934	99934	99934
B					
Divisão Tempo	744	72	10	3	2
Divisão Comparações	133347235541997	13356010702148	1358901900008	158612997074	38391185124
Multiplicação Tempo	705	75	11	4	4
Multiplicação Comparaçõ	133348346976842	13351694607669	1358739777520	158294806801	38337063724
Dobramento Tempo	2743	976	954	930	929
Dobramento Comparaçõ	133350841702415	44010008690351	4401000869035	44010008690351	4401000869035

Figura 2. Tabela hash com 100 mil elementos

I	10	100	1000	10000	100000
Divisão Tempo	10	8	62	5	10
Divisão Colisões	499990	499900	499000	490000	400733
Multiplicação Tempo	65	36	12	12	25
Multiplicação Colisões	499990	499900	499000	490000	400689
Dobramento Tempo	52	14	11	12	15
Dobramento Colisões	499990	499930	499930	499930	499930
B					
Divisão Tempo	20465	2081	246	71	35
Divisão Comparações	16663891040323587	1666918477177151	167263098583428	17293049853358	2293137372159
Multiplicação Tempo	20524	2109	241	70	34
Multiplicação Comparações	16663908433799107	1666952191491504	167254420352573	17283069948469	2290359293070
Dobramento Tempo	20630	6611	6646	6663	45645
Dobramento Comparações	16663894758482785	5459416889547074	5459416889547074	5459416889547074	5459416889547074

Figure 3. Tabela hash com 500 mil elementos

I	10	100	1000	10000	100000
Divisão Tempo	19	166	97	19	160
Divisão Colisões	999990	999900	999000	990000	900005
Multiplicação Tempo	33	89	23	84	138
Multiplicação Colisões	999990	999900	999000	990000	900002
Dobramento Tempo	131	86	60	54	109
Dobramento Colisões	999990	999928	999928	999928	999930
B					
Divisão Tempo	98527	10466	1168	270	132
Divisão Comparações	133286683933587864	13330932239638945	1335405988931335	135769214546133	15823545023543
Multiplicação Tempo	99556	10272	1096	267	120
Multiplicação Comparações	133287089292034681	13330989203627160	1335420863244727	135777725773919	15830732328152
Dobramento Tempo	100313	32827	35158	35240	94631
Dobramento Comparações	133287073885994680	43697719861010820	43697719861010820	43697719861010820	43697719861010820

Figura 4. Tabela hash com 1 milhão de elementos

4. Conclusa˜o

Notou-se que quanto maior a tabela hash, melhor e o desempenho de todas as funções hash devido às colisões serem menos frequentes que as tabelas menores, isso favorece também a busca.

Entre as funções hash, a de multiplicação se saiu melhor na inserção e a divisão se saiu melhor nas buscas nas tabelas hash menores, nas tabelas maiores que dominou foi a func¸a˜o hash de divisão, a função hash de dobramento em poucos casos foi superior às outras duas.

Esses resultados podem variar dependendo do tamanho da tabela hash e da máquina que está executando.

O primeiro conjunto de dados foram executados em um pouco mais que 1 segundo, o conjunto de 100 mil elementos demora cerca de 5 a 10 segundos, o de 500 mil demora de 20 a 30 minutos, 1 milhão demora cerca de 1 hora e meia e o de 5 milhões demora 8 horas para executar.