RA3 - TABELAS HASH
Análise de Desempenho

Gabriel Felipe Jess Meira

[1]Engenharia de Software - Pontíficia Universidade Católica do Paraná (PUCPR)

meira.gabriel@tecpuc.com.br

Abstract. *This report aims to provide data relating to the performance of hash tables that use remainder methods of division, multiplication and folding to be generated together with the rehashing technique. With this report, it will be possible to identify which of the three methods is most efficient for generating tables and searching for elements in a hash table.*

Resumo. *Este relatório visa disponibilizar dados referentes ao desempenho de tabelas hash que utilizam métodos de resto de divisão, multiplicação e dobramento para serem geradas junto com a técnica de encadeamento. Com este relatório, será possível identificar qual dos três métodos é mais eficiente para gerar tabelas e buscar elementos em uma tabela hash.*

1. Tabelas Hash

Tabelas hash são estruturas de dados que utilizam funções de hash para mapear chaves a valores. Elas são usadas para implementar arrays associativos, ou seja, estruturas que mapeiam chaves a valores. Uma função de hash é uma função que converte inputs (como strings) em valores numéricos. A tabela hash usa essa função para calcular um índice no qual o valor associado a uma determinada chave será armazenado ou recuperado.

As tabelas hash oferecem tempos de acesso médio muito eficientes, frequentemente próximos a O(1) (tempo constante) para operações de inserção, remoção e busca, tornando-as uma escolha popular em uma ampla variedade de aplicações que requerem eficiência no acesso a dados.

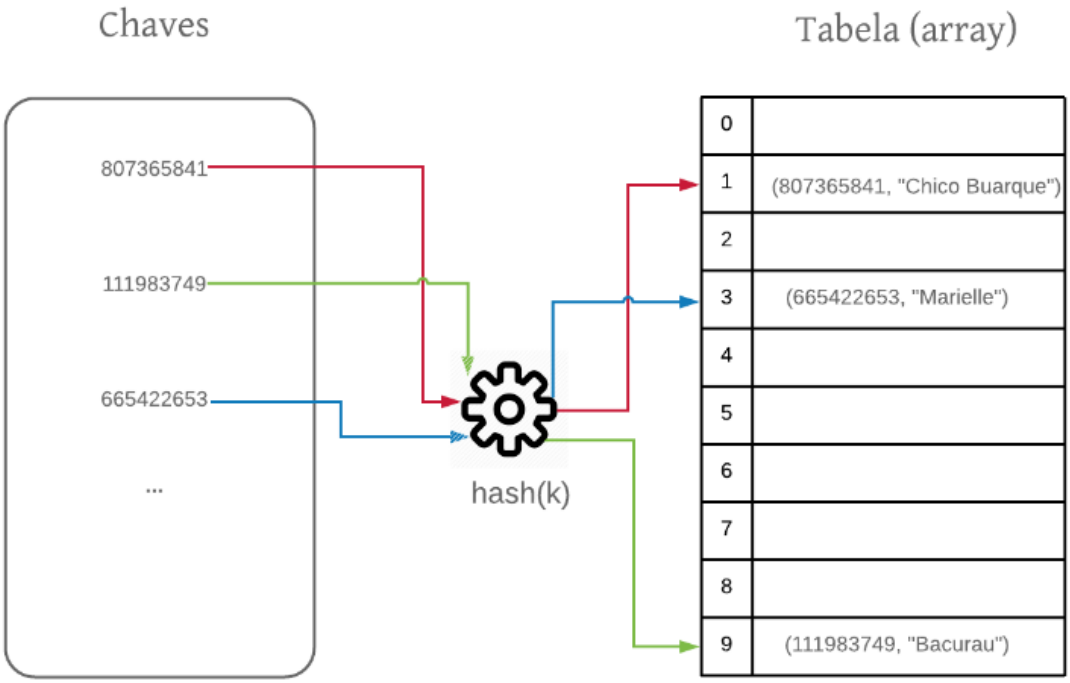


Tabela hash de exemplo

2. Código

O látex desformatou o código, por isso será possível acessar o código pelo seguinte link no github: https://github.com/gabrielfjmeira/AvaliacaoRA3.

Também há um print da implementação no relatório.

3. Geração de Tabela Hash Utilizando o Método de Resto de Divisão

O método de resto de divisão é utilizado para se gerar tabelas hash de uma forma muito simples. Para descobrirmos a posição de um registro na tabela hash, basta descobrir o valor do resto da divisão do código do registro com o tamanho da tabela hash. Com este valor descoberto, basta inserirmos o registro que deve ser inserido na posição calculada.

Caso algum registro já esteja na posição referenciada, então ocorre uma reordenação na lista encadeada de registros que já estejam inseridos na posição, de forma a ordenar os sucessores do registro que está alocado na tabela hash de forma crescente.

4. Geração de Tabela Hash Utilizando o Método de Multiplição

O método de multiplicação também é muito utilizado para se gerar tabelas hash. É um método que utiliza uma constante matemática (hashing de fibonacci = 0,618) para calcular o index do registro no array da tabela hash. Para descobrirmos a posição de um registro na tabela hash, basta multiplicar o valor do do código do registro pelo hashing de fibonacci, em seguida, com o resultado da multiplicação, se deve realizar uma operação do resto da divisão do resultado obtido com o número 1. Assim, basta inserirmos o registro na posição calculada.

Caso algum registro já esteja na posição referenciada, então ocorre uma reordenação na lista encadeada de registros que já estejam inseridos na posição, de forma a ordenar os sucessores do registro que está alocado na tabela hash de forma crescente.

5. Geração de Tabela Hash Utilizando o Método de Dobramento

O método de dobramento, como o próprio nome sugere, é uma técnica que soma os algarismos do código do registro e realiza um cálculo matemático com este valor para se descobrir a posição do registro na tabela hash.

O cálculo é simples, o algoritmo soma os dois último algarismos do código do número do registro por rodada do loop, enquanto este código for maior que 0, a cada soma realizada, o valor é armazenado e utilizado para somar a soma dos próximos elementos em uma soma total. Depois de somar todos os números do código do registro basta realizarmos a operação de resto da divisão da soma total com o tamanho da tabela hash. Assim, basta inserirmos o registro na posição calculada.

Caso algum registro já esteja na posição referenciada, então ocorre uma reordenação na lista encadeada de registros que já estejam inseridos na posição, de forma a ordenar os sucessores do registro que está alocado na tabela hash de forma crescente.

6. Implementação dos Métodos

Nos parâmetros de entrada das funções que utilizam os métodos é passado o código do registro a ser inserido/buscado e um código de método onde: 1 (Resto de Divisão), 2 (Multiplicação) e 3 (Dobramento). Que passa para as funções encontrar e localizar o índice onde o registro deve ser inserido/buscado:

```
//Método de inserção da tabela hash
6 usages
public long insercaoTabelaHash(Registro noInserido, int metodo){
```

```
//Método de busca
4 usages
public boolean buscaCodRegistroTabelaHash(int codRegistro, int metodo){
```

Cálculo do index:

```
//Definição do index referente ao método escolhido
if (metodo == 1){
    //Index do vetor baseado no método de resto de divisão
    index = codRegistro % this.tamanho;
}else if(metodo == 2){
    //Index do vetor baseado no método de multiplicação (Fibonacci Hashing)
    index = (int) (this.tamanho * ((codRegistro * 0.618) % 1));
} else if (metodo == 3) {
    //Index do vetor baseado no método de dobramento
    int codRegistroAux = codRegistro;
    int soma = 0;
    while (codRegistroAux > 0) {
        //Seleciona os dois últimos dígitos de cada bloco
        soma = soma + (codRegistroAux % 100);
        //Atualiza a chave auxiliar (diminuindo suas casas decimais)
        codRegistroAux = codRegistroAux / 100;
    }
    //Gera o index baseado no método de dobramento (resto da divisão da soma dos dois dígitos de cada bloco pelo tamanho da tabela hash)
    index = soma % this.tamanho;
}
```

7. Desempenho Durante a Criação das Tabelas Hash

O desempenho foi medido executando a geração de cada tabela em cada um dos 5 seeds diferentes, em seguida foi elaborado gráficos para visualizar o desempenho de criação

de cada tabela hash usando os 5 seeds diferentes tanto em tempo(µs) e em número de colisões.

	CRIAÇÃO TABELA HASH - 10 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	1.004.263	9.456.635	37.588.097	86.627.798	273.712.460
	653.884	6.157.626	37.248.895	86.490.379	274.659.161
	650.592	6.145.017	36.926.876	86.661.545	273.871.340
	648.787	6.036.703	37.623.681	86.542.123	274.871.249
	674.459	6.130.031	37.237.984	85.924.890	274.145.421
Multiplicação	3.585.839	8.225.971	35.460.802	81.800.756	274.241.459
	3.005.605	5.958.941	34.651.492	85.270.841	267.509.363
	2.809.476	6.202.504	34.118.201	84.330.040	269.345.789
	2.875.415	6.140.509	34.520.005	83.124.091	270.824.512
	2.641.365	6.285.049	34.988.276	84.562.451	273.902.141
Dobramento	1.133.359	7.907.259	32.392.636	86.477.742	263.530.278
	1.477.435	6.124.317	32.515.601	84.475.780	266.123.561
	1.263.931	6.234.086	33.089.559	88.464.337	264.567.891
	1.327.519	5.890.153	36.951.728	85.412.424	267.515.152
	1.248.947	5.946.032	32.527.705	87.167.757	265.124.568
	TEMPO (µs)				

	CRIAÇÃO TABELA HASH - 10 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	62.406.281	251.400.341	1.563.245.517	1.955.251.663	25.005.198.637
	62.406.281	251.400.341	1.563.245.517	1.955.251.663	25.005.198.637
	62.406.281	251.400.341	1.563.245.517	1.955.251.663	25.005.198.637
	62.406.281	251.400.341	1.563.245.517	1.955.251.663	25.005.198.637
	62.406.281	251.400.341	1.563.245.517	1.955.251.663	25.005.198.637
Multiplicação	62.403.677	251.403.926	1.563.168.136	1.955.601.278	25.006.863.436
	62.403.677	251.403.926	1.563.168.136	1.955.601.278	25.006.863.436
	62.403.677	251.403.926	1.563.168.136	1.955.601.278	25.006.863.436
	62.403.677	251.403.926	1.563.168.136	1.955.601.278	25.006.863.436
	62.403.677	251.403.926	1.563.168.136	1.955.601.278	25.006.863.436
Dobramento	62.397.025	251.402.431	1.563.195.755	1.955.266.281	25.005.076.870
	62.397.025	251.402.431	1.563.195.755	1.955.266.281	25.005.076.870
	62.397.025	251.402.431	1.563.195.755	1.955.266.281	25.005.076.870
	62.397.025	251.402.431	1.563.195.755	1.955.266.281	25.005.076.870
	62.397.025	251.402.431	1.563.195.755	1.955.266.281	25.005.076.870
	COLISÕES				

	CRIAÇÃO TABELA HASH - 100 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	164.991	733.386	4.233.343	9.154.218	28.481.087
	173.106	742.875	3.899.287	8.820.767	27.913.482
	160.729	931.712	3.835.619	8.822.980	27.489.878
	169.159	891.122	3.849.640	8.762.969	27.712.055
	205.475	948.248	3.867.417	8.704.147	27.615.683
Multiplicação	346.155	752.400	3.543.047	8.903.561	29.178.245
	322.930	675.805	3.483.861	9.186.053	28.276.162
	357.705	855.864	3.558.596	8.731.647	27.726.891
	397.909	828.155	3.502.631	8.885.862	27.506.095
	413.427	868.680	3.512.295	8.051.748	28.177.030
Dobramento	213.295	643.672	3.301.940	8.754.638	25.005.940
	181.788	645.822	3.648.552	9.147.864	25.036.734
	182.432	822.300	3.516.520	9.771.001	25.583.544
	172.056	802.635	3.501.061	8.461.391	26.509.213
	243.740	820.066	3.394.385	7.864.271	26.436.386
	TEMPO (µs)				

	CRIAÇÃO TABELA HASH - 100 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	6.172.675	25.002.846	155.960.445	624.313.696	2.499.089.695
	6.172.675	25.002.846	155.960.445	624.313.696	2.499.089.695
	6.172.675	25.002.846	155.960.445	624.313.696	2.499.089.695
	6.172.675	25.002.846	155.960.445	624.313.696	2.499.089.695
	6.172.675	25.002.846	155.960.445	624.313.696	2.499.089.695
Multiplicação	6.206.498	25.151.378	156.808.146	627.719.712	2.512.696.202
	6.206.498	25.151.378	156.808.146	627.719.712	2.512.696.202
	6.206.498	25.151.378	156.808.146	627.719.712	2.512.696.202
	6.206.498	25.151.378	156.808.146	627.719.712	2.512.696.202
	6.206.498	25.151.378	156.808.146	627.719.712	2.512.696.202
Dobramento	6.175.513	25.008.467	155.978.711	624.362.523	2.499.161.782
	6.175.513	25.008.467	155.978.711	624.362.523	2.499.161.782
	6.175.513	25.008.467	155.978.711	624.362.523	2.499.161.782
	6.175.513	25.008.467	155.978.711	624.362.523	2.499.161.782
	6.175.513	25.008.467	155.978.711	624.362.523	2.499.161.782
	COLISÕES				

	CRIAÇÃO TABELA HASH - 1.000 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	129.280	259.512	667.603	1.391.439	4.059.112
	118.551	243.224	639.085	1.278.698	4.040.732
	116.518	240.363	661.351	1.415.894	3.925.094
	113.430	252.398	635.430	1.281.824	3.832.539
	98.922	237.949	670.124	1.338.484	5.360.560
Multiplicação	128.228	232.811	648.712	1.605.967	5.622.349
	124.087	219.484	655.596	1.581.244	4.830.461
	126.919	224.152	640.863	1.470.182	4.975.319
	113.302	217.479	656.437	1.498.512	5.213.782
	103.001	239.623	610.697	1.608.885	6.185.624
Dobramento	155.385	471.152	1.767.133	4.071.047	17.871.802
	140.945	433.343	1.562.041	5.418.277	14.146.107
	136.479	397.009	1.574.461	5.378.130	12.988.292
	110.338	429.752	1.627.039	4.744.426	17.832.140
	121.422	425.621	1.723.313	5.623.321	17.879.253
	TEMPO (µs)				

	CRIAÇÃO TABELA HASH - 1.000 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	555.005	2.373.371	15.267.131	61.738.124	248.571.026
	555.005	2.373.371	15.267.131	61.738.124	248.571.026
	555.005	2.373.371	15.267.131	61.738.124	248.571.026
	555.005	2.373.371	15.267.131	61.738.124	248.571.026
	555.005	2.373.371	15.267.131	61.738.124	248.571.026
Multiplicação	767.551	3.229.934	20.620.597	83.181.773	335.084.347
	767.551	3.229.934	20.620.597	83.181.773	335.084.347
	767.551	3.229.934	20.620.597	83.181.773	335.084.347
	767.551	3.229.934	20.620.597	83.181.773	335.084.347
	767.551	3.229.934	20.620.597	83.181.773	335.084.347
Dobramento	2.934.743	11.896.605	74.508.229	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.508.229	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.508.229	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.508.229	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.508.229	298.648.015	1.196.330.369
	COLISÕES				

	CRIAÇÃO TABELA HASH - 10.000 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	114.421	160.542	292.454	499.735	1.385.299
	103.394	215.117	287.477	490.442	1.400.810
	91.531	232.274	293.597	485.919	1.477.399
	88.906	207.944	300.607	488.658	1.339.420
	88.762	239.824	303.301	498.146	1.297.897
Multiplicação	109.804	248.564	659.006	1.630.018	6.755.552
	103.674	203.174	666.721	1.607.464	6.052.279
	113.119	198.536	648.929	1.609.174	5.838.587
	110.479	207.483	651.415	1.601.054	6.145.874
	102.057	217.042	656.715	1.562.398	5.984.771
Dobramento	129.828	470.286	1.627.479	5.701.674	22.780.678
	113.919	344.235	1.545.513	5.795.841	19.789.557
	128.983	357.402	1.606.885	5.491.727	18.172.770
	127.213	352.417	1.615.871	5.645.806	18.840.536
	129.415	363.145	1.689.832	5.331.102	25.331.901
	TEMPO (µs)				

	CRIAÇÃO TABELA HASH - 10.000 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	21.475	144.718	1.238.811	5.554.753	23.573.492
	21.475	144.718	1.238.811	5.554.753	23.573.492
	21.475	144.718	1.238.811	5.554.753	23.573.492
	21.475	144.718	1.238.811	5.554.753	23.573.492
	21.475	144.718	1.238.811	5.554.753	23.573.492
Multiplicação	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
Dobramento	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	COLISÕES				

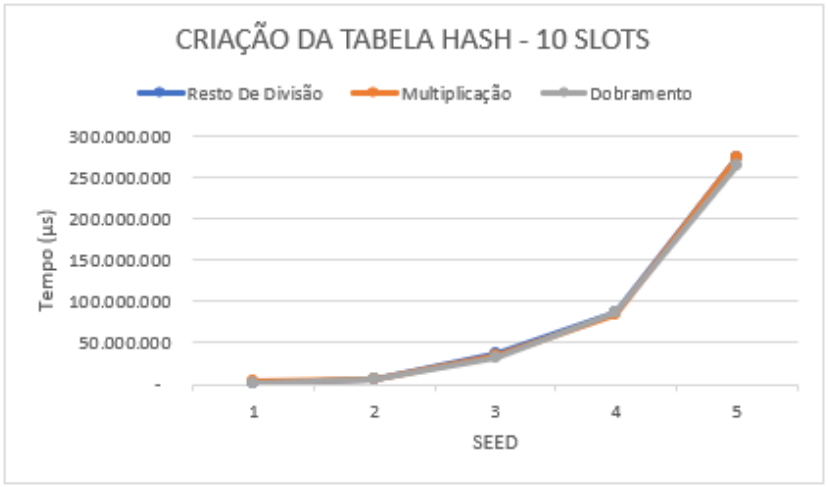
	CRIAÇÃO TABELA HASH - 100.000 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	94.367	136.448	250.932	390.218	845.967
	85.742	146.565	243.681	446.446	852.516
	97.365	146.135	244.181	412.703	847.031
	91.353	143.323	254.887	397.134	843.989
	87.001	135.329	284.488	391.344	847.272
Multiplicação	103.227	218.401	658.847	1.612.563	7.213.058
	117.108	207.245	652.324	1.612.969	7.167.932
	110.905	204.827	702.781	1.621.238	7.047.857
	107.892	229.814	661.058	1.792.666	6.309.699
	119.369	208.822	662.713	1.663.219	7.116.664
Dobramento	130.815	339.652	1.668.704	4.297.927	22.431.275
	119.682	415.792	1.803.956	5.754.213	19.920.176
	146.238	386.137	1.558.439	5.864.216	24.454.149
	116.913	371.272	1.655.872	5.877.520	20.136.823
	132.387	351.830	1.640.146	5.615.157	21.192.435
	TEMPO (µs)				

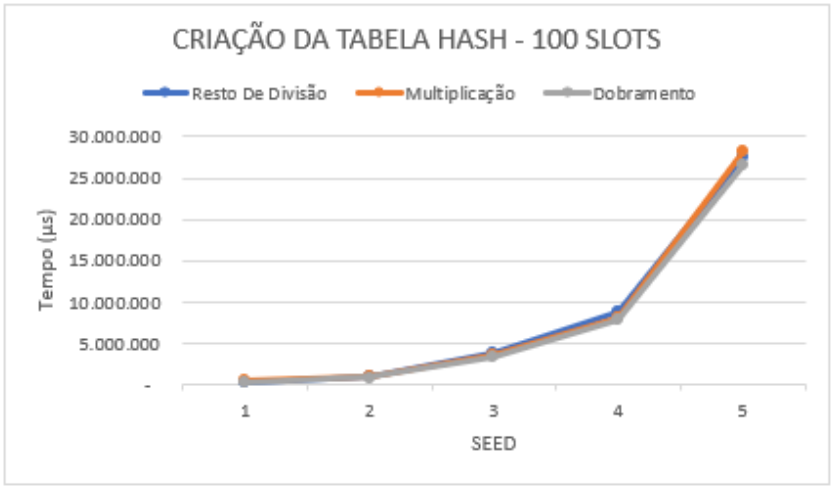
	CRIAÇÃO TABELA HASH - 100.000 slots				
	SEED				
	1	2	3	4	5
Resto De Divisão	65	1.004	25.005	216.201	1.426.506
	65	1.004	25.005	216.201	1.426.506
	65	1.004	25.005	216.201	1.426.506
	65	1.004	25.005	216.201	1.426.506
	65	1.004	25.005	216.201	1.426.506
Multiplicação	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
	767.551	3.229.934	20.700.935	83.181.773	335.084.347
Dobramento	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	2.934.743	11.896.605	74.486.720	298.648.015	1.196.330.369
	COLISÕES				

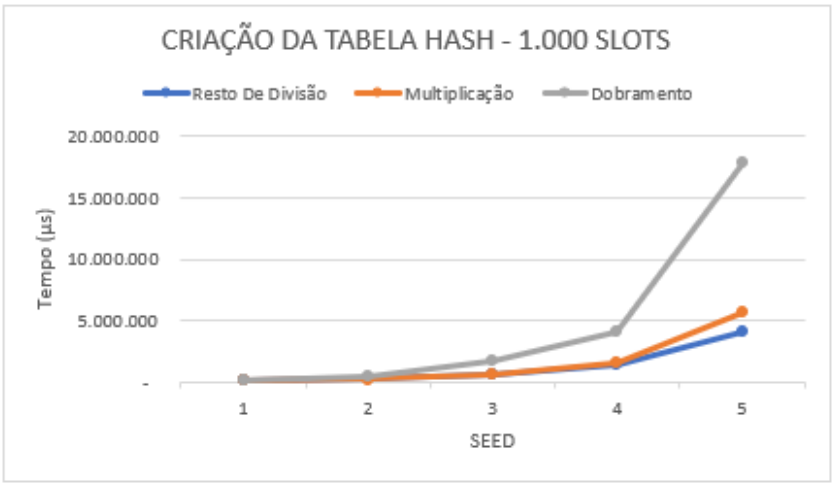
8. Conclusão do Desempenho das Criações nas Tabelas Hash

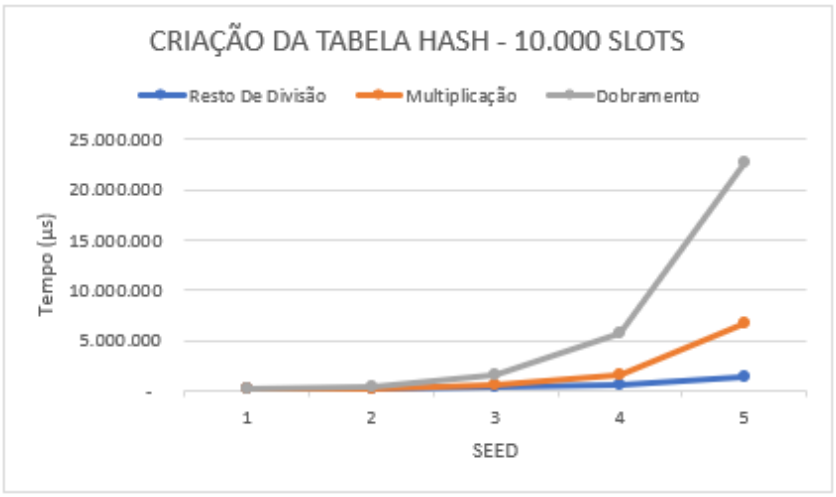
Para melhor visualização foram elaborados os gráficos abaixo.

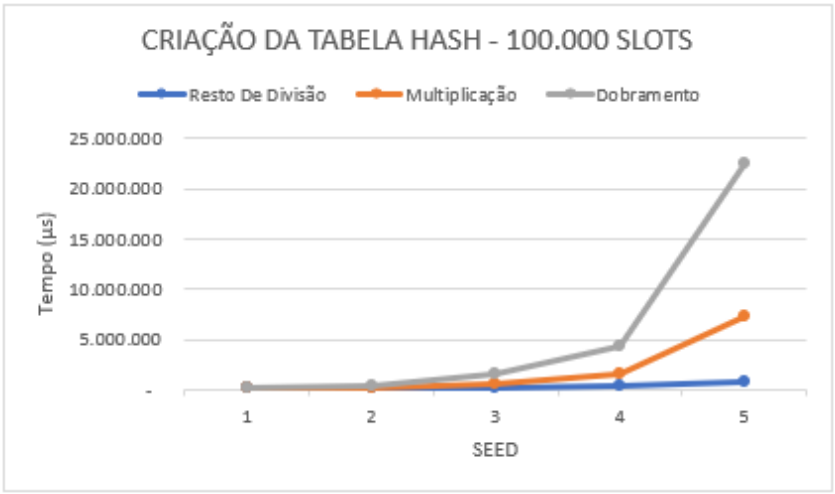
Refente ao tempo(µs):


CRIAÇÃO DA TABELA HASH - 10 SLOTS
Resto De Divisão
Multiplicação
Dobramento
300.000.000
250.000.000
200.000.000
150.000.000
100.000.000
50.000.000
-
Tempo (µs)
1
2
3
4
5
SEED


CRIAÇÃO DA TABELA HASH - 100 SLOTS
Resto De Divisão
Multiplicação
Dobramento
30.000.000
25.000.000
20.000.000
15.000.000
10.000.000
5.000.000
-
Tempo (µs)
1
2
3
4
5
SEED

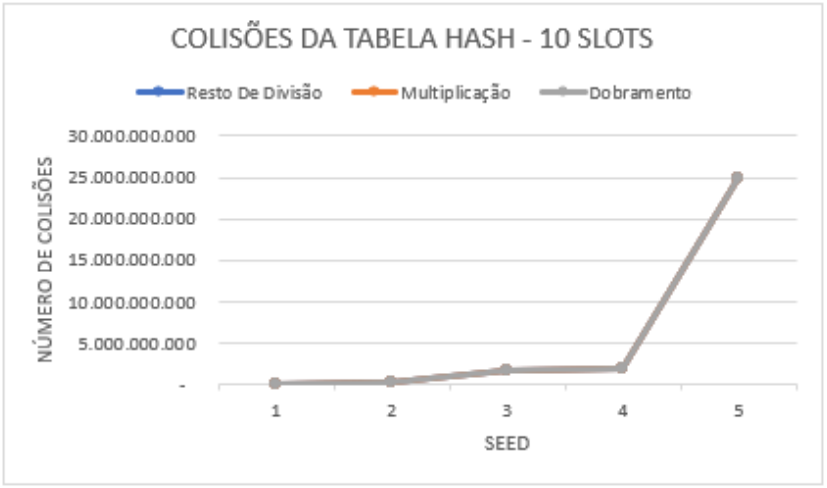

CRIAÇÃO DA TABELA HASH - 1.000 SLOTS
Resto De Divisão
Multiplicação
Dobramento
20.000.000
15.000.000
10.000.000
5.000.000
-
Tempo (µs)
1
2
3
4
5
SEED

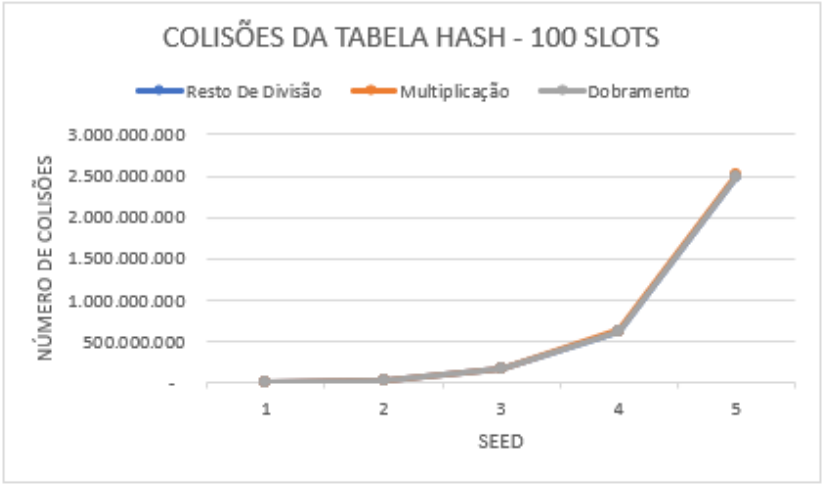


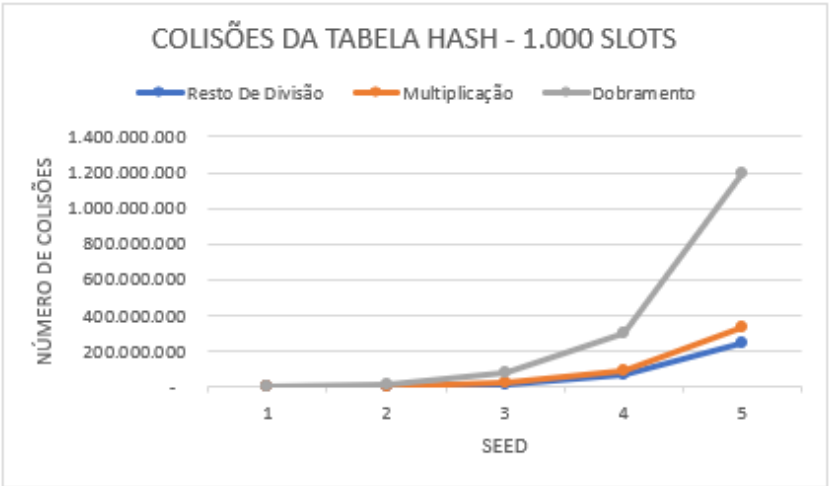


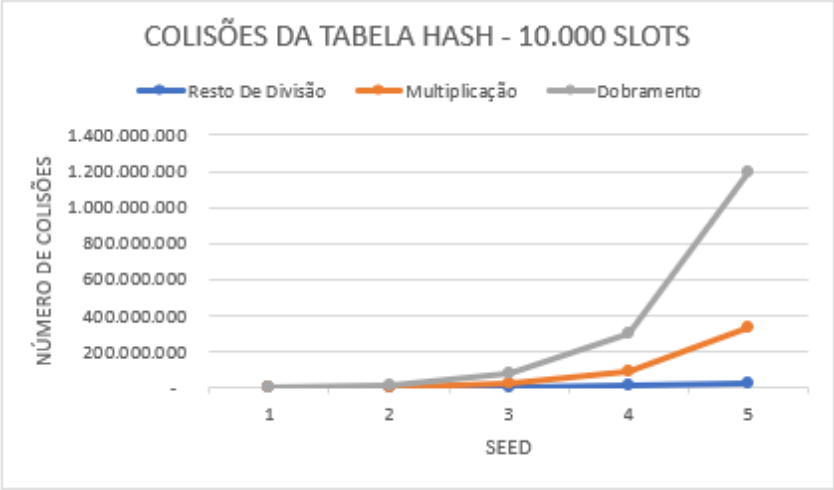
Quando se trata de tempo podemos perceber que o método de dobramento possuí destaque quando se trata de menos slots no array. Em compensação o método de resto de divisão é o que gera as tabelas hash de uma forma mais rápida quando se trata de muitos slots. O método de multiplicação não possuiu um destaque, se performou bem nestes dois ambientes.

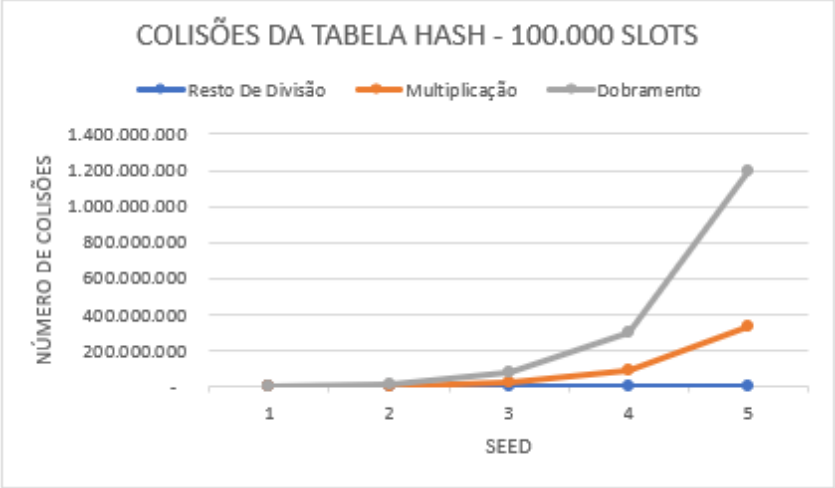
Refente a Colisões:


COLISÕES DA TABELA HASH - 10 SLOTS
Resto De Divisão
Multiplicação
Dobramento
NÚMERO DE COLISÕES
30.000.000.000
25.000.000.000
20.000.000.000
15.000.000.000
10.000.000.000
5.000.000.000
-
1
2
3
4
5
SEED


COLISÕES DA TABELA HASH - 100 SLOTS
Resto De Divisão
Multiplicação
Dobramento
NÚMERO DE COLISÕES
3.000.000.000
2.500.000.000
2.000.000.000
1.500.000.000
1.000.000.000
500.000.000
-
1
2
3
4
5
SEED


COLISÕES DA TABELA HASH - 1.000 SLOTS
Resto De Divisão
Multiplicação
Dobramento
NÚMERO DE COLISÕES
1.400.000.000
1.200.000.000
1.000.000.000
800.000.000
600.000.000
400.000.000
200.000.000
-
1
2
3
4
5
SEED





Os métodos até certo tamanho de vetor possuem basicamente o mesmo número de colisões totais, porém quando há muitos elementos presentes no vetor, o método de dobramento acaba tendo muito mais colisões que os demais. O que menos possuí colisões é o método de resto de divisão.

A geração da tabela hash varia muito de acordo com o número de slots e o número de elementos a serem inseridos, devido ao surgimento de várias colisões.

9. Desempenho Durante a Busca nas Tabelas Hash

O desempenho foi medido executando a busca de 5 registros diferentes em cada tabela no seed de 250 mil elementos (seed3). Os registros para busca foram inseridos após a geração de cada tabela hash.

Obtive os seguintes dados:

BUSCA TABELA HASH - 10 slots				
	MÉTODOS			
	RESTO DE DIVISÃO	MULTIPLICAÇÃO	DOBRAMENTO	
123456789	291	262	264	µs
COLISÕES	3044	3015	2963	
421567834	707	700	145	µs
COLISÕES	10473	10599	10386	
124572893	117	22	52	µs
COLISÕES	3131	3117	3174	
178361585	82	32	69	µs
COLISÕES	4429	4573	4369	
156847823	75	36	59	µs
COLISÕES	3899	3890	3928	

BUSCA TABELA HASH - 100 slots				
	MÉTODOS			
	RESTO DE DIVISÃO	MULTIPLICAÇÃO	DOBRAMENTO	
123456789	82	55	45	µs
COLISÕES	306	344	309	
421567834	108	67	77	µs
COLISÕES	1056	1025	1033	
124572893	87	47	52	µs
COLISÕES	325	333	310	
178361585	135	63	63	µs
COLISÕES	440	555	444	
156847823	75	32	44	µs
COLISÕES	374	339	411	

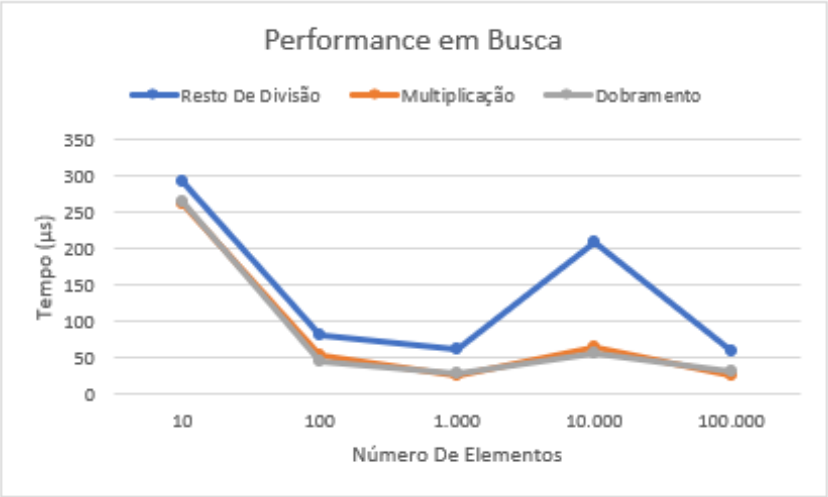
BUSCA TABELA HASH - 1.000 slots				
	MÉTODOS			
	RESTO DE DIVISÃO	MULTIPLICAÇÃO	DOBRAMENTO	
123456789	63	25	30	µs
COLISÕES	39	67	160	
421567834	71	30	57	µs
COLISÕES	105	221	695	
124572893	70	21	30	µs
COLISÕES	40	72	191	
178361585	64	31	49	µs
COLISÕES	58	91	276	
156847823	60	18	25	µs
COLISÕES	37	18	190	

BUSCA TABELA HASH - 10.000 slots				
	MÉTODOS			
	RESTO DE DIVISÃO	MULTIPLICAÇÃO	DOBRAMENTO	
123456789	210	65	56	µs
COLISÕES	7	67	160	
421567834	83	38	70	µs
COLISÕES	9	221	695	
124572893	57	20	26	µs
COLISÕES	7	72	191	
178361585	55	21	35	µs
COLISÕES	5	91	276	
156847823	56	17	41	µs
COLISÕES	6	18	190	

BUSCA TABELA HASH - 100.000 slots				
	MÉTODOS			
	RESTO DE DIVISÃO	MULTIPLICAÇÃO	DOBRAMENTO	
123456789	60	25	31	µs
COLISÕES	1	67	160	
421567834	54	30	58	µs
COLISÕES	1	221	695	
124572893	76	30	47	µs
COLISÕES	3	72	191	
178361585	57	21	31	µs
COLISÕES	1	91	276	
156847823	63	22	28	µs
COLISÕES	1	18	190	

10. Conclusão do Desempenho das Buscas nas Tabelas Hash

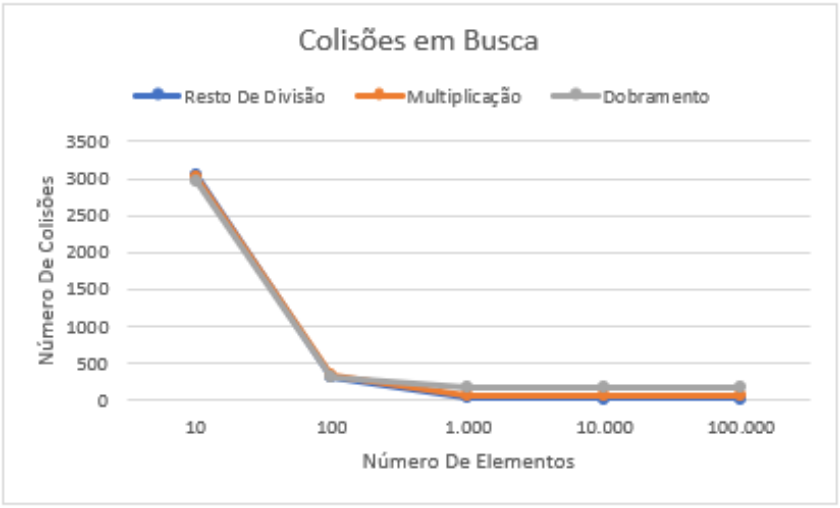
Para uma melhor visualização da performance em buscas, foi gerado o seguinte gráfico de tempo(µs):



Os métodos que possuíram uma melhor eficiência na busca foram o de multiplicação e de dobramento, enquanto o de resto da divisão chegou a demorar em certos casos, 2x mais tempo que estes métodos.

Isto ocorre devido a rápidez com que os cálculos de cada posição é realizado de acordo com o método(capacidade de agrupamento e aproveitamento da memória) e a quantidade de registros na lista encadeada.

Gráfico referente ao número de colisões:



O número de colisões diminuí de acordo com o tamanho do vetor. Isto ocorre devido o vetor possuir mais slots para distribuir os registros.

11. Conclusão

Foram implementados três métodos de geração de tabela hash neste trabalho, cada um se performou melhor em um cenário em específico.

Portanto, quando se for implementar uma tabela hash, deve se observar o seu tamanho e o número de elementos que a mesma deve suportar, para escolhermos o melhor método tanto para criação tanto para a busca de um registro.

References

SC. CAMILA TAUMATURGO, M. Disciplina: Estruturas de Dados Professora: Camila Taumaturgo e-mail: camila.taumaturgo@ifrn.edu.br. Disponível em: ¡https://docente.ifrn.edu.br/camilataumaturgo/disciplinas/2014.2/estruturas-de-dados/tabela-hash¿. Acesso em: 3 nov. 2023.

BACKES, A. TABELA HASH. Disponível em: ¡https://www.facom.ufu.br/ backes/gsi011/Aula07-TabelaHash.pdf¿. Acesso em: 6 nov. 2023.

Método de dobragem em hash – Acervo Lima. Disponível em: ¡https://acervolima.com/metodo-de-dobragem-em-hash/¿. Acesso em: 3 nov. 2023.

Materiais disponibilizados no Canvas da matéria de Resolução de Problemas Estruturados em Computação.