Análise da Tabela de Hashing de Cuco

Docente: João Dias

Discente: Diogo Fonseca nº79858

Análise de qualidade das funções de hashing

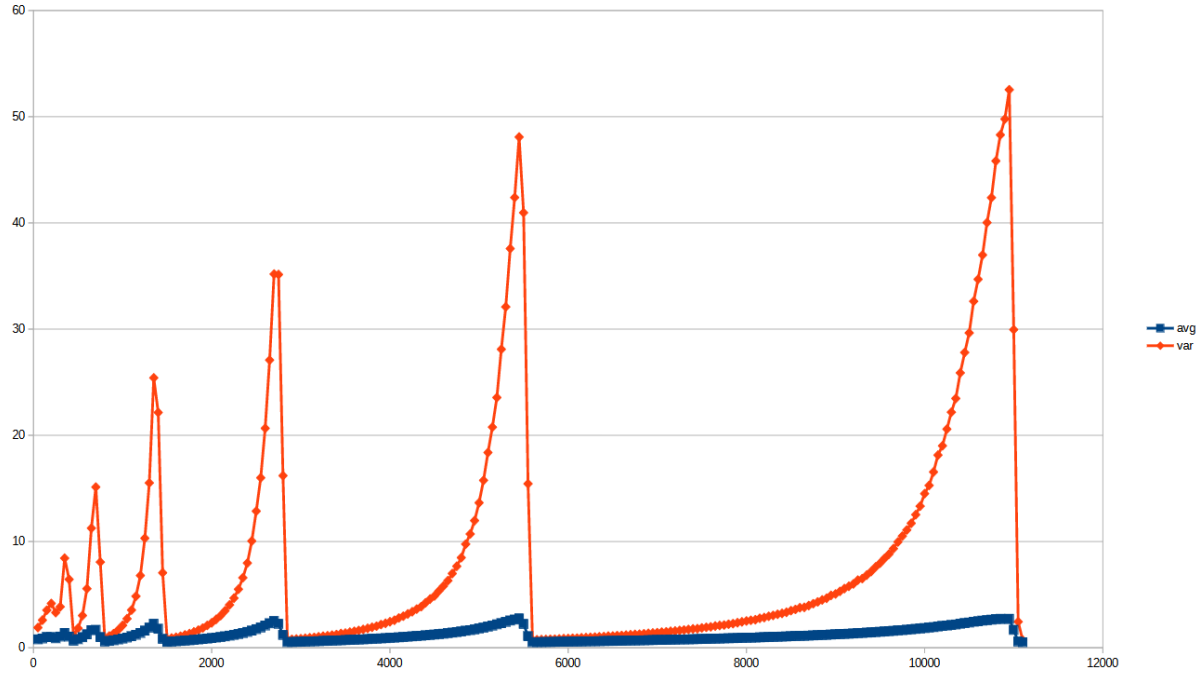

Gráfico I. Média e Variância do número de colisões por tamanho da tabela de hash

Obs: Os dados do gráfico I (a tabela 1) estão apresentados no final do relatório.

Testes: Foi enchida uma tabela com *n* elementos (aos quais as chaves são inteiros para não haver mais de 2 hashcodes iguais, o qual a tabela não suporta). Posteriormente ao enchimento da tabela é anotado os valores da variância (var) e da média (avg) do número de colisões. Este processo é repetido 10000 para cada *n*, ao qual se faz a média geométrica dos valores. A este teste, iterou-se *n* por incrementos de 50, de 50 a 11000.

Durante os testes a hashtable manteve um número máximo de colisões consecutivas a 100 principalmente para evitar loops infinitos, e o critério de redimensionamento é a partir de um fator de carga > 0.5.

Olhando para o Gráfico I é fácil observar o funcionamento da hashtable em função de *n*. Conforme *n* se aproxima da capacidade da hashtable, por outras palavras, conforme o fator de carga cresce, a média de colisões sobe. Isto vem do facto de que quanto maior o fator de carga (quanto mais elementos, para o mesmo tamanho da array), maior a probabilidade de haver uma colisão, pois há menos espaços livres no array.

Pela natureza da cuckoo hash, conforme a probabilidade de colisão sobe, o número médio de colisões sobe exponencialmente. A variância reflete isto bem, já que vai haver uma variância muito maior conforme o fator de carga sobe (enquanto alguns elementos podem ser colocados na primeira iteração, alguns outros podem andar na ordem das centenas de iterações até encontrarem um espaço livre, criando uma variância muito grande quando o fator de carga é grande).

Pelo gráfico é também possível ver o redimensionamento em funcionamento, sempre que *n* se aproxima de 50% da capacidade a um certo ponto, vê-se um redimensionamento, resultante de uma diminuição drástica da variância (pois o redimensionamento baixa muito significativamente a chance de colisão).

É também possível observar na periodicidade dos picos apresentados que o redimensionamento é sempre feito para o dobro do anterior.

Fora a funcionalidade normal, as funções de hash parecem distribuir uniformemente os objetos pelo array, caso contrário veríamos uma "linearidade" mais forte e uns valores da média muito maiores quando o fator de carga é mais baixo. Mas pode-se observar o contrário, até ao fator de carga subir, é possível ver uma média de colisões muito baixa (e uma variância muito baixa também, indicando que não há valores a divergir muito da média).

Também é possível observar uma (muito) ligeira "suavização" da queda da variância nos redimensionamentos. Isto dá-se, pois, conforme o fator de carga se aproxima de 0.5, a probabilidade de ficar num ciclo muito grande, ou mesmo infinito, e então acionar as 100 iterações máximas, aumenta e o redimensionamento é realizado prematuramente.

Análise do impacto do esquecimento

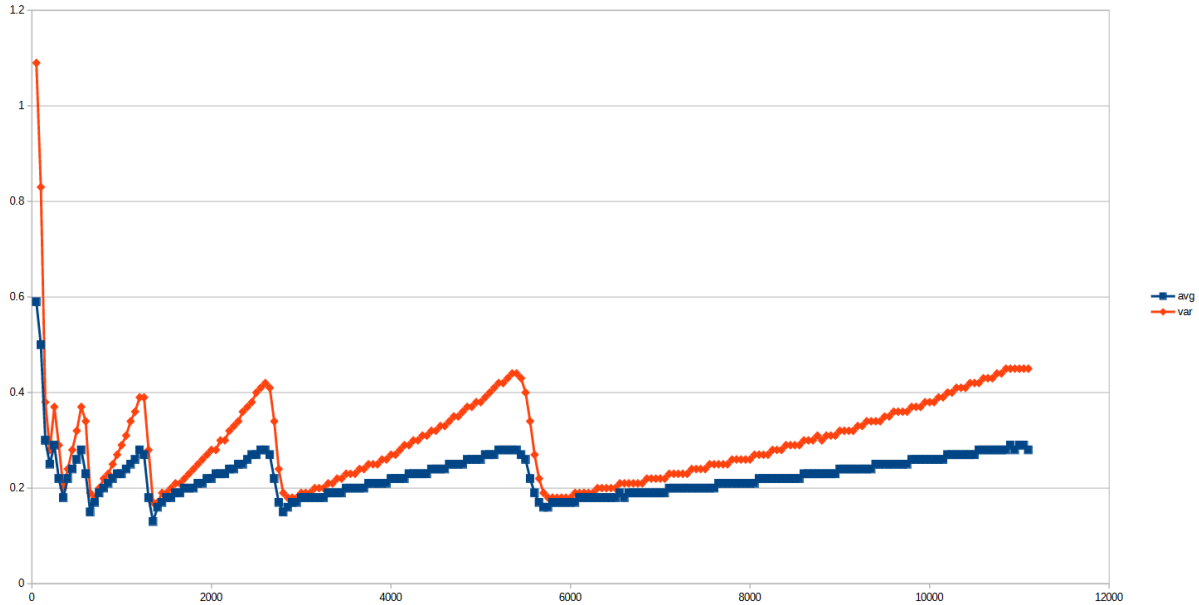

Gráfico II. Média e Variância do número de colisões por tamanho da tabela de hash com esquecimento

Obs: Os dados do gráfico I (a tabela 1) estão apresentados no final do relatório.

Testes: Análogos aos anteriores, embora desta vez são calculados todos os dados a por previamente e ~20% deles são postos numa lista para guardar quais serão constantemente acedidos. São então colocados os dados na hashtable, tendo em conta que a cada objeto inserido, é incrementado o tempo da tabela por 1 (hora) e a cada 23 incrementos da hora é usado o método get() na lista com 20% dos dados para os manter "utilizados".

Com o sistema de esquecimento podemos observar uma diferença brutal tanto na média de colisões como na variância. Sendo que 80% da tabela na verdade não vai ter uma colisão, será apenas reescrito por cima, então perto de um fator de carga de 50% a tabela age como se tivesse um fator de carga de 10% (0.2×0.5). O valor 0.2 vem do facto de que só 20% da tabela é que realmente tem dados capazes de colidir (vão precisar de uma reinserção consequente).

Isto obviamente faz uma tabela híper eficiente, enquanto sem o sistema de esquecimento os valores da média e da variância rondavam em média os ~[0.5; 3.0] e ~[0.2; 50] respetivamente, com o sistema de esquecimento os valores em média variam entre ~[0.1; 0.3] e ~[0.2; 0.45]. De um máximo de ~50 na variância para um máximo de ~0.45.

A "suavização" das quedas neste gráfico é mais acentuada pois há uma maior dispersão da probabilidade de redimensionamento. Neste caso isto dá-se devido ao fator de carga de 50% ser atingido ligeiramente mais aleatoriamente, já que se os objetos forem postos em lugares vazios quase sempre, o redimensionamento vai acontecer mais cedo, e se os objetos calharem em lugares onde havia outro objeto o qual foi esquecido, o redimensionamento vai acontecer mais tarde.

n	Tempo Médio de Execução (ns)	Razão
250	416	0.0
500	230	0.6
1000	177	0.8
2000	605	3.4
4000	1053	1.7
8000	875	0.8
16000	1349	1.5
32000	2531	1.9
64000	2591	1.0
128000	2716	1.0
256000	2802	1.0
512000	2867	1.0
1024000	2923	1.0
2048000	3043	1.0
4096000	3045	1.0

Tabela 3. Testes de razão dobrada da função put() do CuckooHash (redimensionamento a 50%)

n	Tempo Médio de Execução (ns)	Razão
250	71	0
500	39	0.5
1000	44	1.1
2000	42	1.0
4000	79	1.9
8000	86	1.1
16000	87	1.0
32000	95	1.1
64000	124	1.3
128000	162	1.3
256000	232	1.4
512000	301	1.3
1024000	359	1.2
2048000	362	1.0
4096000	377	1.0

Tabela 4. Testes de razão dobrada da função get() do CuckooHash (redimensionamento a 50%)

n	Tempo Médio de Execução (ns)	Razão
250	576	0.0
500	265	0.5
1000	258	1.0
2000	545	2.1
4000	901	1.7
8000	1336	1.5
16000	2194	1.6
32000	2553	1.2
64000	2911	1.1
128000	2963	1.0
256000	2937	1.0
512000	3171	1.1
1024000	3164	1.0
2048000	3287	1.0
4096000	3238	1.0

Tabela 5. Testes de razão dobrada da função put() de Linear Probing Hash (redimensionamento a 50%)

n	Tempo Médio de Execução (ns)	Razão
250	84	0.0
500	47	0.6
1000	60	1.3
2000	50	0.8
4000	68	1.4
8000	67	1.0
16000	75	1.1
32000	73	1.0
64000	91	1.2
128000	107	1.2

256000	202	1.9
512000	393	1.9
1024000	337	0.9
2048000	344	1.0
4096000	370	1.1

Tabela 6. Testes de razão dobrada da função get() de Linear Probing Hash (redimensionamento a 50%)

Análise da complexidade temporal e comparação com linear probing

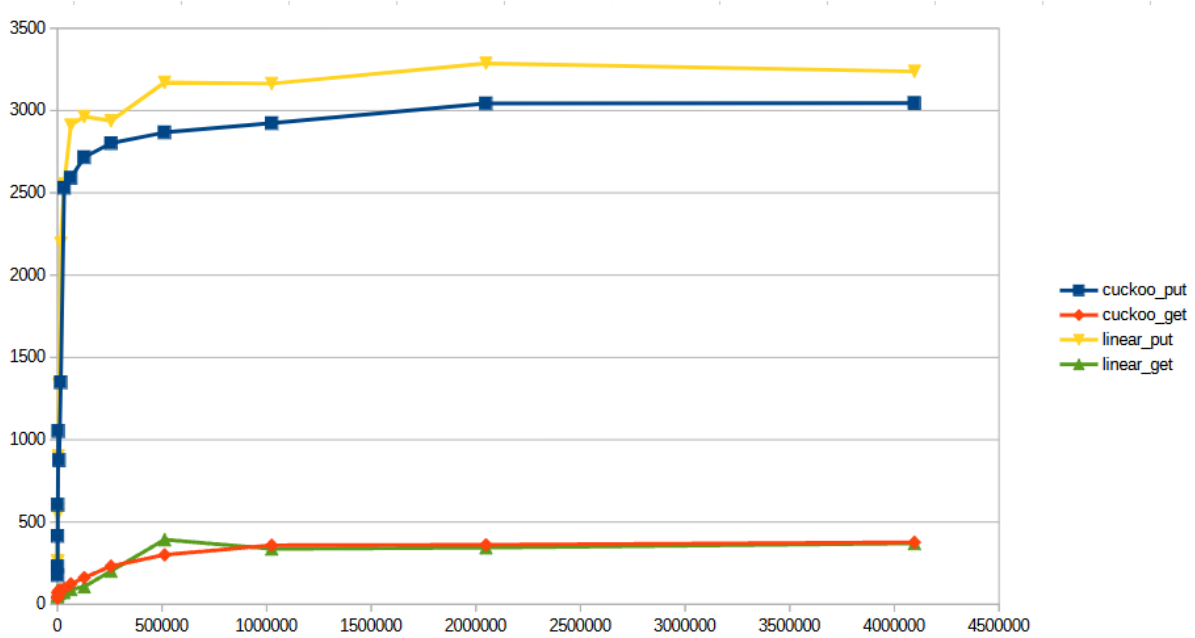

Gráfico III-VI. Testes de razão dobrada das funções get() e put() da Linear Probing Hash e da Cuckoo Hash (redimensionamento a 50%)

Obs: (tempo de execução em função de n em nanossegundos, n sendo o nº de objetos na hashtable)

Testes das funções put(): foram feitos testes de razão dobrada, aos quais após n inserções (com chaves inteiras aleatórias) é colocado mais um elemento (com chave inteira aleatória) e medido o seu tempo de execução. Isto é feito 1000 vezes para cada n ao qual depois se faz a média geométrica destes.

Testes das funções get(): foram feitos testes de razão dobrada, aos quais após n inserções (com chaves inteiras aleatórias) são feitos 100000 gets de chaves aleatórias existentes na tabela e medido o seu tempo de execução, ao qual se faz a média geométrica dos mesmos.

Olhando para o gráfico III-VI pode-se observar ambas funções de ambas as hashtables a executar em tempo constante a partir da iteração $n = 512000$. O seu carácter logarítmico inicialmente vem dos redimensionamentos, os quais duplicam o tamanho da tabela.

Observando a razão nas 4 tabelas podemos tirar a mesma conclusão. Vê-se que a razão tende para 1.0, ou, constante O(1).

Comparando a função get() da linear probing hash com a cuckoo hash, podemos concluir que para um fator de carga de 50% no máximo, são mais ou menos idênticas, com uma ligeira vantagem por parte da linear probing hash, isto é de esperar já que para uma tabela com fator de carga < 50% a linear probing hash tem mais ou menos o mesmo número de comparações, mas só tem de calcular a hash uma vez.

Comparando a função put() da linear probing hash com a cuckoo hash, podemos ver que empiricamente para as funções e testes utilizados a cuckoo hash é mais rápida que a linear probing hash. Com base nos testes realizados é difícil perceber o porquê deste ser o caso, sendo que se esperaria da linear probing hash com open addressing ser mais rápida, já que esta só necessita de calcular o hash uma vez em vez de duas. Para além de um menor overhead e menos comparações por inserção (maior simplicidade do algoritmo).

A conclusão que pode ser tirada após uma análise lógica do código para os testes realizados (por falta de tempo para testes suficientes), a única causa provável é a ligeira discrepância no tamanho dos arrays, os números primos usados como tamanho dos arrays são diferentes para o mesmo n, o tamanho da linear probing hash é ligeiramente menor, causando ligeiramente mais colisões, e por consequência ligeiramente pior eficiência (diferença de ~150ns). Por falta de mais testes não é possível chegar a uma conclusão empírica da causa destes valores.

n	Tempo Médio de Execução (ns)	Razão
250	576	0.0
500	235	0.4
1000	211	0.9
2000	543	2.6
4000	514	1.0
8000	733	1.4
16000	1081	1.5
32000	1446	1.3
64000	2274	1.6
128000	2122	0.9
256000	2590	1.2
512000	3026	1.2
1024000	2997	1.0
2048000	3053	1.0
4096000	2901	1.0

Tabela 7. Testes de razão dobrada da função put() do CuckooHash (redimensionamento a 75%)

n	Tempo Médio de Execução (ns)	Razão
250	75	0.0
500	41	0.5

1000	56	1.4
2000	51	0.9
4000	50	1.0
8000	75	1.5
16000	72	1.0
32000	91	1.3
64000	144	1.6
128000	168	1.2
256000	191	1.1
512000	289	1.5
1024000	319	1.1
2048000	346	1.1
4096000	340	1.0

Tabela 8. Testes de razão dobrada da função get() do CuckooHash (redimensionamento a 75%)

n	Tempo Médio de Execução (ns)	Razão
250	436	0.0
500	261	0.6
1000	255	1.0
2000	1271	5.0
4000	3792	3.0
8000	3919	1.0
16000	3737	1.0
32000	3707	1.0
64000	3923	1.0
128000	3900	1.0
256000	3336	0.9
512000	3741	1.1
1024000	3571	1.0
2048000	3955	1.1
4096000	4195	1.1

Tabela 9. Testes de razão dobrada da função put() de Linear Probing Hash (redimensionamento a 75%)

n	Tempo Médio de Execução (ns)	Razão
250	75	0.0
500	51	0.7
1000	50	1.0
2000	48	1.0
4000	52	1.1
8000	59	1.1
16000	89	1.5
32000	87	1.0
64000	91	1.1
128000	225	2.5
256000	220	1.0
512000	338	1.5
1024000	321	0.9
2048000	359	1.1
4096000	353	1.0

Tabela 10. Testes de razão dobrada da função get() de Linear Probing Hash (redimensionamento a 75%)

Comparação com linear probing para redimensionamento a 75%

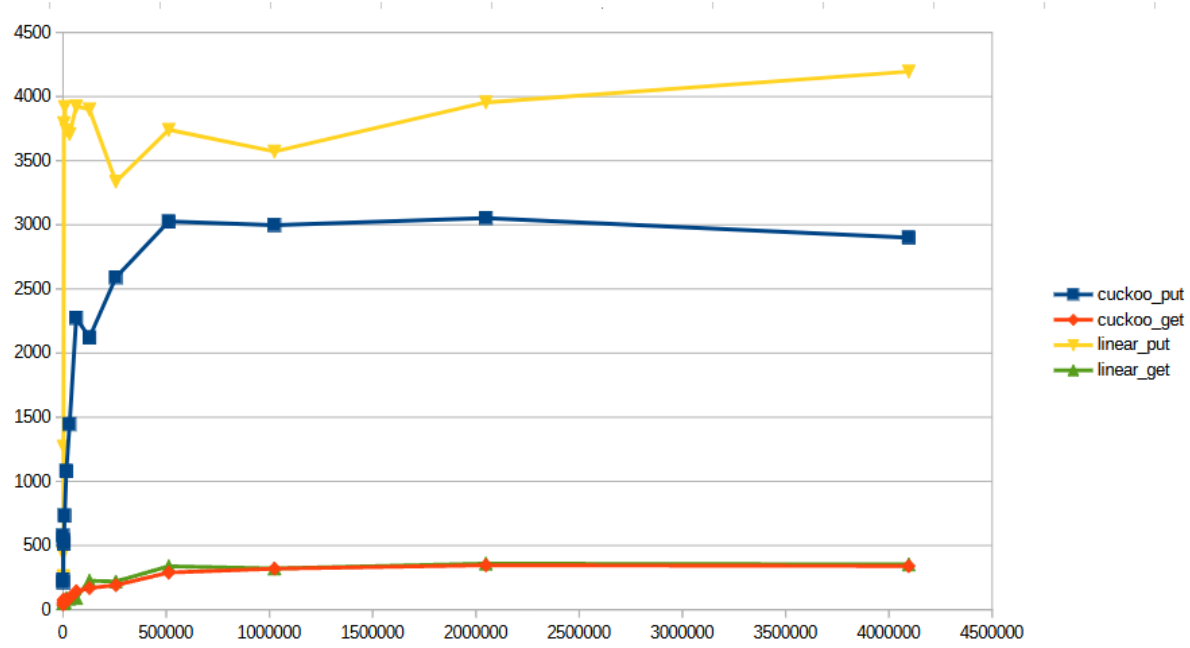

Gráfico VII-X. Testes de razão dobrada das funções get() e put() da Linear Probing Hash e da Cuckoo Hash (redimensionamento a 75%)

Obs: (tempo de execução em função de *n* em nanossegundos, *n* sendo o nº de objetos na hashtable para um redimensionamento a 75% de fator de carga)

Testes: análogos aos anteriores, mas com só redimensionamento o array a partir de um fator de carga de 75%.

Comparando a função get() da linear probing hash com a cuckoo hash, podemos concluir que com um redimensionamento a 75% de fator de carga a cuckoo hash porta-se melhor, isto é esperado pois a linear probing hash dá-se mal com fatores de carga grandes (começa a ficar demasiado linear devido à sua natureza, é necessário atravessar mais chaves para chegar à desejada em média). A cuckoo hash, porta-se igual no get() para qualquer fator de carga, já que esta só precisa de fazer duas operações de hashing e duas comparações para encontrar a chave.

Comparando a função put() da linear probing hash com a cuckoo hash, podemos ver que com um redimensionamento a 75% de fator de carga, embora ambas sejam mais lentas, a discrepância entre as duas aumentou substancialmente, que é o comportamento esperado pois conforme a tabela é toda preenchida haverão mais “clusters” onde a linear probing hash vai funcionar linearmente, enquanto a cuckoo hash vai probabilisticamente encontrar espaços vazios mais frequentemente pois as inserções consecutivas são praticamente aleatórias (assumindo que as funções de hashing dispersam bem as chaves) e não será tão afetado por “clusters”.

Apesar da perda de eficiência de ambas as funções, pode-se confirmar que continuam a operar com complexidade constante.

Dados para o gráfico I e II

n	Média de colisões	Variância de colisões
50	0.79	1.9
100	0.87	2.6
150	1.01	3.54
200	1	4.17
250	0.89	3.31
300	1.05	3.86
350	1.39	8.43
400	1.05	6.44
450	0.66	1.2
500	0.8	1.84
550	0.99	3.03
600	1.26	5.56
650	1.66	11.26
700	1.68	15.13
750	1	8.07
800	0.58	0.91
850	0.64	1.1
900	0.7	1.34
950	0.78	1.67
1000	0.86	2.11
1050	0.96	2.73
1100	1.08	3.55
1150	1.22	4.85
1200	1.39	6.8
1250	1.61	10.31
1300	1.89	15.52
1350	2.25	25.41
1400	1.78	22.15
1450	0.83	7.06
1500	0.55	0.82
1550	0.58	0.91
1600	0.61	0.99
1650	0.64	1.09
1700	0.67	1.21
1750	0.7	1.33
1800	0.74	1.49
1850	0.78	1.66
1900	0.82	1.86
1950	0.86	2.1
2000	0.91	2.35
2050	0.96	2.67
2100	1.01	3.03
2150	1.07	3.48
2200	1.14	4.04
2250	1.21	4.67
2300	1.29	5.51
2350	1.38	6.59

2400	1.49	7.97
2450	1.61	10.05
2500	1.75	12.86
2550	1.9	16
2600	2.09	20.66
2650	2.3	27.08
2700	2.51	35.18
2750	2.25	35.14
2800	1.2	16.2
2850	0.53	0.75
2900	0.54	0.79
2950	0.55	0.82
3000	0.56	0.85
3050	0.58	0.89
3100	0.59	0.94
3150	0.61	0.98
3200	0.62	1.03
3250	0.64	1.08
3300	0.65	1.13
3350	0.67	1.19
3400	0.68	1.25
3450	0.7	1.34
3500	0.72	1.38
3550	0.74	1.47
3600	0.75	1.55
3650	0.77	1.63
3700	0.79	1.72
3750	0.81	1.83
3800	0.84	1.93
3850	0.85	2.03
3900	0.88	2.18
3950	0.9	2.3
4000	0.93	2.45
4050	0.95	2.59
4100	0.98	2.8
4150	1.01	2.97
4200	1.04	3.18
4250	1.07	3.39
4300	1.1	3.64
4350	1.13	3.86
4400	1.17	4.24
4450	1.2	4.58
4500	1.24	4.88
4550	1.28	5.35
4600	1.32	5.81
4650	1.37	6.32
4700	1.42	6.99
4750	1.47	7.67
4800	1.53	8.48
4850	1.6	9.76
4900	1.66	10.71
4950	1.72	11.97

5000	1.81	13.65
5050	1.89	15.76
5100	1.99	18.38
5150	2.08	20.77
5200	2.2	23.56
5250	2.31	28.1
5300	2.43	32.09
5350	2.56	37.58
5400	2.65	42.39
5450	2.76	48.08
5500	2.24	40.96
5550	1.08	15.44
5600	0.52	0.73
5650	0.53	0.75
5700	0.53	0.77
5750	0.54	0.78
5800	0.54	0.8
5850	0.55	0.81
5900	0.56	0.83
5950	0.56	0.86
6000	0.57	0.87
6050	0.58	0.9
6100	0.58	0.91
6150	0.59	0.94
6200	0.6	0.96
6250	0.61	0.98
6300	0.61	1.01
6350	0.62	1.03
6400	0.63	1.05
6450	0.64	1.09
6500	0.64	1.1
6550	0.65	1.14
6600	0.66	1.16
6650	0.67	1.19
6700	0.68	1.23
6750	0.68	1.26
6800	0.69	1.28
6850	0.7	1.32
6900	0.71	1.35
6950	0.72	1.39
7000	0.73	1.42
7050	0.73	1.47
7100	0.74	1.5
7150	0.76	1.55
7200	0.76	1.58
7250	0.78	1.63
7300	0.78	1.67
7350	0.79	1.72
7400	0.8	1.77
7450	0.81	1.79
7500	0.82	1.87
7550	0.83	1.92

7600	0.84	1.97
7650	0.86	2.04
7700	0.86	2.1
7750	0.88	2.14
7800	0.89	2.22
7850	0.9	2.27
7900	0.91	2.36
7950	0.93	2.44
8000	0.94	2.52
8050	0.95	2.59
8100	0.96	2.64
8150	0.98	2.77
8200	0.99	2.84
8250	1.01	2.96
8300	1.02	3.04
8350	1.03	3.15
8400	1.05	3.24
8450	1.06	3.34
8500	1.08	3.49
8550	1.1	3.61
8600	1.11	3.77
8650	1.12	3.8
8700	1.14	3.97
8750	1.16	4.15
8800	1.18	4.32
8850	1.2	4.5
8900	1.21	4.66
8950	1.24	4.93
9000	1.26	5.06
9050	1.28	5.31
9100	1.3	5.57
9150	1.32	5.79
9200	1.34	6.01
9250	1.37	6.37
9300	1.39	6.51
9350	1.42	6.84
9400	1.44	7.18
9450	1.47	7.61
9500	1.5	7.96
9550	1.53	8.41
9600	1.56	8.81
9650	1.59	9.33
9700	1.63	9.96
9750	1.66	10.51
9800	1.69	11.09
9850	1.73	11.72
9900	1.77	12.53
9950	1.81	13.33
10000	1.85	14.51
10050	1.9	15.28
10100	1.94	16.55
10150	2	18.14

10200	2.05	19.01
10250	2.09	20.58
10300	2.14	22.18
10350	2.19	23.47
10400	2.26	25.88
10450	2.32	27.8
10500	2.37	29.64
10550	2.45	32.62
10600	2.49	34.69
10650	2.55	36.98
10700	2.59	40.03
10750	2.64	42.38
10800	2.69	45.83
10850	2.72	48.29
10900	2.71	49.78
10950	2.72	52.54
11000	1.68	29.95
11050	0.58	2.45
11100	0.52	0.72

Tabela 1. Dados estatísticos para n

n	Média de colisões	Variância de colisões
50	0.59	1.09
100	0.5	0.83
150	0.3	0.38
200	0.25	0.28
250	0.29	0.37
300	0.22	0.29
350	0.18	0.2
400	0.22	0.24
450	0.24	0.28
500	0.26	0.32
550	0.28	0.37
600	0.23	0.34
650	0.15	0.19
700	0.17	0.18
750	0.19	0.2
800	0.2	0.22
850	0.21	0.23
900	0.22	0.25
950	0.23	0.27
1000	0.23	0.29
1050	0.24	0.31
1100	0.25	0.34
1150	0.26	0.36
1200	0.28	0.39
1250	0.27	0.39
1300	0.18	0.28

1350	0.13	0.17
1400	0.16	0.17
1450	0.17	0.19
1500	0.18	0.19
1550	0.18	0.2
1600	0.19	0.21
1650	0.19	0.21
1700	0.2	0.22
1750	0.2	0.23
1800	0.2	0.24
1850	0.21	0.25
1900	0.21	0.26
1950	0.22	0.27
2000	0.22	0.28
2050	0.23	0.28
2100	0.23	0.3
2150	0.23	0.3
2200	0.24	0.32
2250	0.24	0.33
2300	0.25	0.34
2350	0.25	0.36
2400	0.26	0.37
2450	0.27	0.38
2500	0.27	0.4
2550	0.28	0.41
2600	0.28	0.42
2650	0.27	0.41
2700	0.22	0.34
2750	0.17	0.24
2800	0.15	0.19
2850	0.16	0.18
2900	0.17	0.18
2950	0.17	0.18
3000	0.18	0.19
3050	0.18	0.19
3100	0.18	0.19
3150	0.18	0.2
3200	0.18	0.2
3250	0.18	0.2
3300	0.19	0.21
3350	0.19	0.21
3400	0.19	0.22
3450	0.19	0.22
3500	0.2	0.23
3550	0.2	0.23
3600	0.2	0.23
3650	0.2	0.24
3700	0.2	0.24
3750	0.21	0.25
3800	0.21	0.25
3850	0.21	0.25
3900	0.21	0.26

3950	0.21	0.26
4000	0.22	0.27
4050	0.22	0.27
4100	0.22	0.28
4150	0.22	0.29
4200	0.23	0.29
4250	0.23	0.3
4300	0.23	0.3
4350	0.23	0.31
4400	0.23	0.31
4450	0.24	0.32
4500	0.24	0.32
4550	0.24	0.33
4600	0.24	0.33
4650	0.25	0.34
4700	0.25	0.35
4750	0.25	0.35
4800	0.25	0.36
4850	0.26	0.37
4900	0.26	0.37
4950	0.26	0.38
5000	0.26	0.38
5050	0.27	0.39
5100	0.27	0.4
5150	0.27	0.41
5200	0.28	0.42
5250	0.28	0.42
5300	0.28	0.43
5350	0.28	0.44
5400	0.28	0.44
5450	0.27	0.43
5500	0.26	0.4
5550	0.22	0.34
5600	0.19	0.27
5650	0.17	0.22
5700	0.16	0.19
5750	0.16	0.18
5800	0.17	0.18
5850	0.17	0.18
5900	0.17	0.18
5950	0.17	0.18
6000	0.17	0.18
6050	0.17	0.19
6100	0.18	0.19
6150	0.18	0.19
6200	0.18	0.19
6250	0.18	0.19
6300	0.18	0.2
6350	0.18	0.2
6400	0.18	0.2
6450	0.18	0.2
6500	0.18	0.2

6550	0.19	0.21
6600	0.18	0.21
6650	0.19	0.21
6700	0.19	0.21
6750	0.19	0.21
6800	0.19	0.21
6850	0.19	0.22
6900	0.19	0.22
6950	0.19	0.22
7000	0.19	0.22
7050	0.19	0.22
7100	0.2	0.23
7150	0.2	0.23
7200	0.2	0.23
7250	0.2	0.23
7300	0.2	0.23
7350	0.2	0.24
7400	0.2	0.24
7450	0.2	0.24
7500	0.2	0.24
7550	0.2	0.25
7600	0.2	0.25
7650	0.21	0.25
7700	0.21	0.25
7750	0.21	0.25
7800	0.21	0.26
7850	0.21	0.26
7900	0.21	0.26
7950	0.21	0.26
8000	0.21	0.26
8050	0.21	0.27
8100	0.22	0.27
8150	0.22	0.27
8200	0.22	0.27
8250	0.22	0.28
8300	0.22	0.28
8350	0.22	0.28
8400	0.22	0.29
8450	0.22	0.29
8500	0.22	0.29
8550	0.22	0.29
8600	0.23	0.3
8650	0.23	0.3
8700	0.23	0.3
8750	0.23	0.31
8800	0.23	0.3
8850	0.23	0.31
8900	0.23	0.31
8950	0.23	0.31
9000	0.24	0.32
9050	0.24	0.32
9100	0.24	0.32

9150	0.24	0.32
9200	0.24	0.33
9250	0.24	0.33
9300	0.24	0.34
9350	0.24	0.34
9400	0.25	0.34
9450	0.25	0.34
9500	0.25	0.35
9550	0.25	0.35
9600	0.25	0.36
9650	0.25	0.36
9700	0.25	0.36
9750	0.25	0.36
9800	0.26	0.37
9850	0.26	0.37
9900	0.26	0.37
9950	0.26	0.38
10000	0.26	0.38
10050	0.26	0.38
10100	0.26	0.39
10150	0.26	0.39
10200	0.27	0.4
10250	0.27	0.4
10300	0.27	0.41
10350	0.27	0.41
10400	0.27	0.41
10450	0.27	0.42
10500	0.27	0.42
10550	0.28	0.42
10600	0.28	0.43
10650	0.28	0.43
10700	0.28	0.43
10750	0.28	0.44
10800	0.28	0.44
10850	0.28	0.45
10900	0.29	0.45
10950	0.28	0.45
11000	0.29	0.45
11050	0.29	0.45
11100	0.28	0.45

Tabela 2. Dados estatísticos para n com esquecimento