Relatório HASH RA03

Gustavo Albiero, Gustavo Fraiz

November 23, 2023

1 Introdução

https://github.com/gustavoFraiz/AvaliacaoRA4.git

Neste trabalho foi implementado e analisado o desempenho de diferentes tabelas Hash em Java, foi analisado e comparado o desempenho de diferentes situações de inserções e tamanhos de tabelas, observando o número de colisões, tempo de busca e tempo de inserção nas 3 diferentes operações implementadas. Para as análises de, foram escolhidas cinco diferentes tamanhos de tabela para um número fixo de inserções, o tamanho das tabelas escolhidas foi escolhido a partir da quantidade de inserções, com a primeira tendo como tamanho a metade da quantidade de inserções, a segunda com o tamanho da tabela igual ao de inserções, a terceira com o tamanho sendo o dobro do número de inserções, a quarta com cinco vezes, e a última com a tabela de tamanho referente à dez vezes o número de inserções.

2 Tabelas Hash

2.0.1 20.000 Inserções

Operação	Tamanho	Colisões	Busca	Tempo de Inserção(ms)
1	10.000	11.361	8	6
2	10.000	11.412	10	9
3	10.000	11.380	6	10

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	20.000	7403	16	5
2	20.000	7419	15	9
3	20.000	7417	7	10

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	40.000	4289	19	6
2	40.000	4507	10	9
3	40.000	4251	12	8

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	100.000	1897	27	6
2	100.000	2866	20	11
3	100.000	1940	10	8

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	200.000	976	24	6
2	200.000	2866	19	11
3	200.000	1007	9	9

2.0.2 100.000 Inserções

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	50.000	56.730	55	21
2	50.000	58.914	48	31
3	50.000	56.909	25	30

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	100.000	36.679	36	16
2	100.000	47.785	33	43
3	100.000	36.920	30	31

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	200.000	21.111	34	20
2	200.000	47.785	31	39
3	200.000	21.395	28	29

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	500.000	9396	11	25
2	500.000	47.785	26	41
3	500.000	9493	20	26

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	1.000.000	4846	37	20
2	1.000.000	47.785	33	42
3	1.000.000	4984	22	29

2.0.3 500.000 Inserções

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	250.000	283.779	39	96
2	250.000	422.892	78	282
3	250.000	283.871	40	173

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	500.000	184.139	36	112
2	500.000	422.892	62	311
3	500.000	184.044	36	140

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	1.000.000	106.760	77	88
2	1.000.000	422.892	77	310
3	1.000.000	106.639	50	142

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	5.000.000	24.431	174	98
2	5.000.000	422.892	129	312
3	5.000.000	24.191	76	147

2.0.4 1.000.000 Inserções

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	500.000	567.834	35	208
2	500.000	917.977	96	677
3	500.000	567.613	62	303

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	1.000.000	368.000	105	186
2	1.000.000	917.977	113	703
3	1.000.000	367.949	81	315

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	2.000.000	213.097	168	173
2	2.000.000	917.977	125	686
3	2.000.000	213.087	80	282

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	5.000.000	93.864	256	197
2	5.000.000	917.977	191	712
3	5.000.000	93.578	109	311

2.0.5 5.000.000 Inserções

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	2.500.000	2.835.796	747	1377
2	2.500.000	4.907.867	467	7691
3	2.500.000	2.836.499	403	1948

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	5.000.000	1.837.865	813	1217
2	5.000.000	4.907.867	518	7884
3	5.000.000	1.837.501	364	1668

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	10.000.000	1.062.873	945	1028
2	10.000.000	4.887.665	560	7624
3	10.000.000	1.064.935	403	1583

Operação	Tamanho	Colisões	Busca(ms)	Tempo de Inserção(ms)
1	25.000.000	465.753	1440	1152
2	25.000.000	4.853.156	887	7693
3	25.000.000	470.467	543	1517

3 Conclusões Finais

Após analisar o desempenho da tabela Hash e das operações implementadas, é possível observar que dependendo do número de inserções e o tamanho da tabela, o funcionamento acaba sendo bastante comprometido quando a quantidade de elementos a serem inseridos é absurdamente maior que a tabela.

Em relação ao uso de diferentes funções hash foi observado que a função 2(multiplicação por uma constante) não obteem um resultado positivo quando número da tabela aumenta em relação ao número de inserções. Em quanto as outras funções fazem o lógico, que é diminuir o número de colisões quanto mais espaço tivermos na tabela, a função dois chega ao um ponto que ela para de utilizar o espaço disponível e o número de colisões se mantem o mesmo.

Comparando a tabela Hash com outra estrutura aprendida na disciplina, árvore AVL:

Em termos de complexidade de tempo e big O, a tabela hash parece se sobresair em quase todas as operações em relação a árvore binária balanceada mas tudo depende da função hash escolhida. Se a função hash escolhida para a tabela não for de boa qualidade o algoritmo perde muito em eficiencia sendo o pior caso sendo 0(n). Se a escolha da tabela hash foi eficiente então essa solução se apresenta mais rápida. Porém a árvore também tem suas vantagens: Utiliza menos mémoria do que a tabela hash visto que a árvore aloca dinamicamente ao tempo que se é requisitado a inserção de algum elemento, já o hash os ponteiros para cada espaço de mémoria dentro da tabela já são criados desde o começo, sendo esses usados ou não. A árvore também é melhor para fazer buscas, range searchs e é bem mais fácil achar chaves que estejam pertos uma das outras. A árvore também lida melhor com sets de dados pequenos. Em conclusão é melhor utilizar a tabela hash quando se tem os números de elementos que precisam ser

inseridos já conhecidos antes da criação da tabela, e quando não se tem a necessidade de fazer inserções constantes. Nesses casos ou quando se tem o objetivo de analisar melhor esses dados utilizando buscas detalhadas o recomendado é utilizar uma árvore binária.