RA03 - Trabalho Tabela Hash

Gustavo Cesar Regnel

November 23, 2023

1 Introdução

Neste relatório, irei apresentar os resultados da implementação do código. Primeiramente, será citada a metodologia usada no projeto e em seguida os gráficos e tabelas.

2 Tamanho da Tabela Hash

Eu escolhi usar os tamanhos (50, 500, 5000, 50000, 500000) de tabela hash. Não escolhi esses tamanhos por nenhum motivo específico, apenas para não seguir a sugestão do professor na tarefa.

Como o funcionamento do código é afetado por essa alteração: Em tamanhos menores (ex: 10, 50): As tabelas hash com tamanhos menores têm um aumento nas colisões, especialmente quando o conjunto de dados é grande. O espaçamento dos dados nas buckets pode não ser ideal, resultando em colisões frequentes.

Tamanhos maiores (ex: 1000, 50000, 500000): Tabelas hash maiores têm menores números de colisões. À medida que o tamanho aumenta, o espaçamento dos dados nas buckets tende a ser mais uniforme, o que pode melhorar o desempenho geral.

3 Escolha das Funções Hash

Eu escolhi as mesmas que foram sugeridas pelo professor (Multiplicação, Resto de Divisão e Dobramento), pois tive dificuldades na implementação e achei que assim seria menos complicado.

4 Sobre o código

4.1 Classe Tabela Hash

Esta é a classe principal do projeto, onde é criada a tabela hash e é realizada a implementação da lógica de cada uma das funções hash. Permite a inserção de registros em diferentes posições da tabela, dependendo da função hash escolhida.

4.2 Classe Registro

Esta classe possui o elemento código que representa a chave atual da lista e o atributo próximo, que representa a próxima chave.

4.3 Classe Main

Nesta classe é onde nossos testes serão realizados. Ela mede o número de inserções, comparações e busca, bem como seus respectivos tempos. Nela, cada função hash é instanciada 5 vezes com os tamanhos de tabela hash escolhidos (50, 500, 5000, 50000 e 500000), e também é permitido setar o tamanho do conjunto de dados.

5 Resultados

Nesta seção, estão os resultados em tabelas e gráficos, comparando o desempenho das diferentes tabelas hash e funções hash.

6 Testes

6.1 Função hash: Resto de divisão

Tamanho do conjunto: 500000

	50	500	5000	50000	500000
Inserção	0ms	0ms	0ms	4ms	16ms
Busca	0ms	0ms	0ms	4ms	2ms

Table 1: Para tamanhos menores, a inserção é muito rápida, indicando que a divisão de hash funciona muito bem com números menores. No entanto, em números maiores o tempo de resposta aumenta consideravelmente, devido ao maior número de colisões. Já para a busca, podemos concluir que para números menores ela também é muito rápida. Já para tamanhos maiores, a busca leva um pouco mais de tempo. Novamente, isso sugere que o tempo de busca aumenta à medida que a tabela fica maior, possivelmente devido a um maior número de colisões.

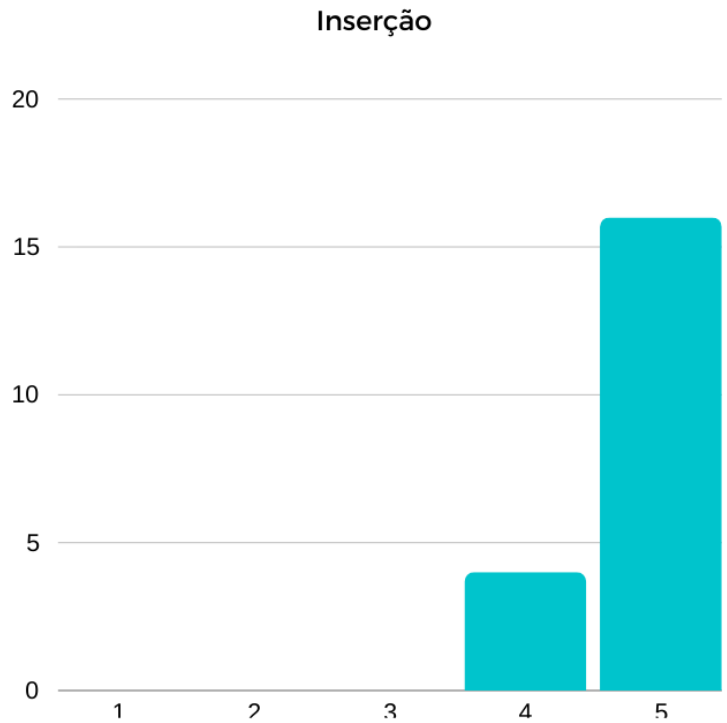

Figure 1: Resultado gráfico do tempo de inserção na Divisão Hash

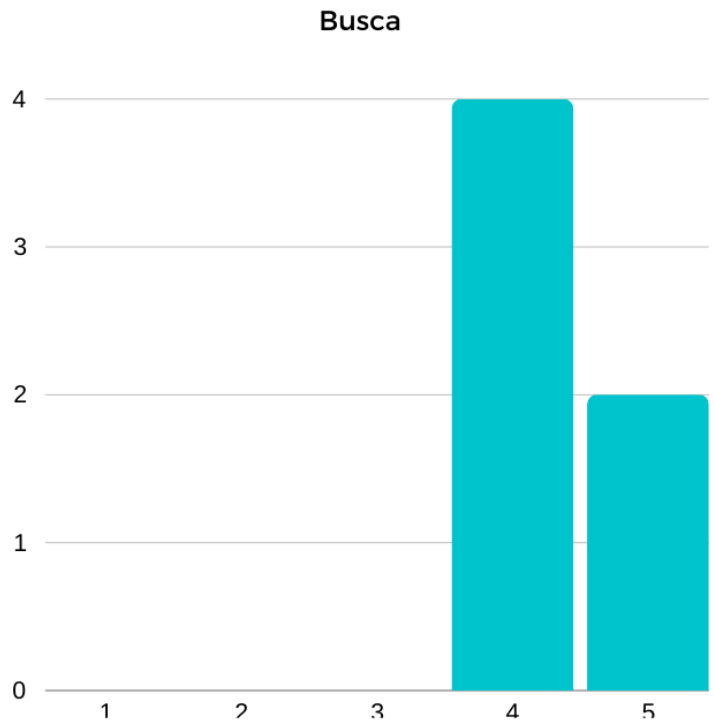

Figure 2: Resultado gráfico do tempo da busca na Divisão Hash

6.2 Função hash: Multiplicação

Tamanho do conjunto: 500000

	50	500	5000	50000	500000
Inserção	0ms	0ms	1ms	6ms	115ms
Busca	0ms	0ms	1ms	5ms	58ms

Table 2: Os resultados indicam que, para tamanhos menores, o tempo de inserção e busca é praticamente instantâneo (0ms). Conforme o tamanho da tabela aumenta, o tempo de inserção e busca cresce, sendo mais significativo em tamanhos maiores, como 50000 e 500000, onde a inserção atinge 115ms e a busca 58ms. Este comportamento sugere um aumento proporcional do tempo de execução com o aumento do tamanho da tabela.

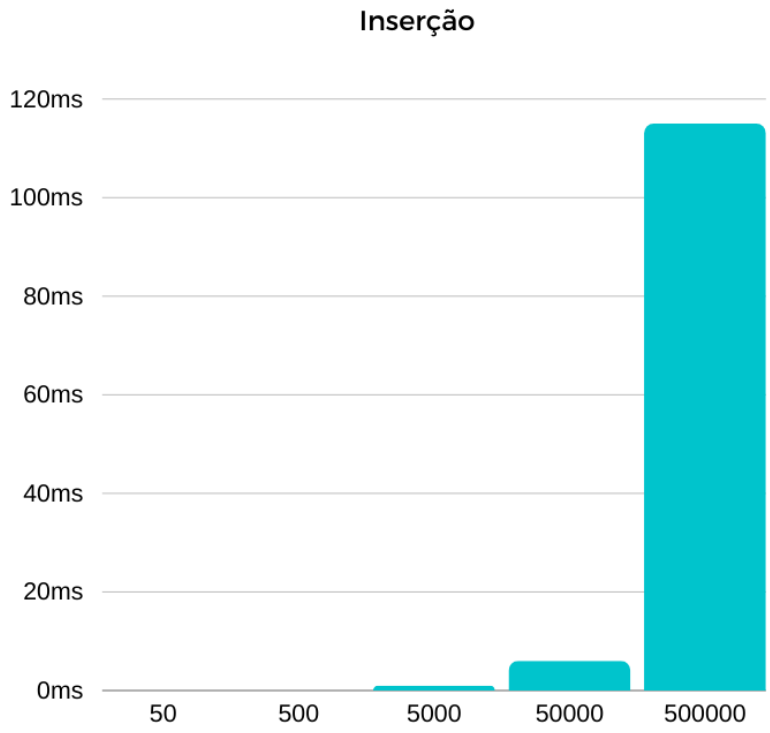

Figure 3: Resultado gráfico do tempo de inserção na Multiplicação Hash

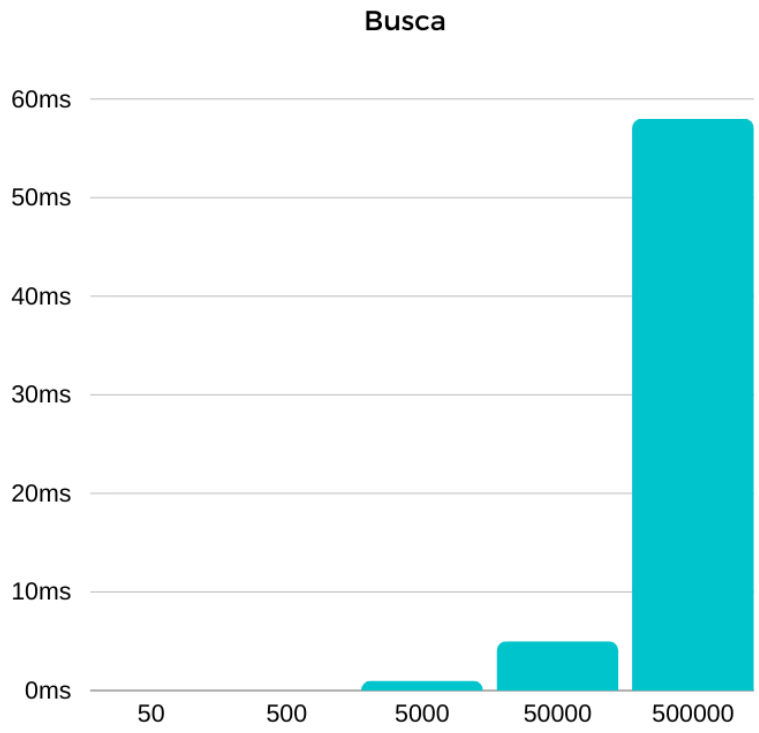

Figure 4: Resultado gráfico do tempo da busca na Multiplicação Hash

6.3 Função hash: Dobramento

Tamanho do conjunto: 500000

	50	500	5000	50000	500000
Inserção	0ms	0ms	2ms	383ms	175791ms
Busca	0ms	0ms	0ms	1ms	10ms

Table 3: Os resultados do teste para a função hash de dobramento mostram que, como nas funções anteriores, para tamanhos menores o tempo de inserção e busca é praticamente instantâneo. No entanto, à medida que o tamanho da tabela aumenta, o tempo de inserção cresce bastante, atingindo 175791ms para 500000. Por outro lado, o tempo de busca permanece relativamente baixo, mesmo para tamanhos maiores, indicando um desempenho melhor na operação de busca em comparação com a inserção. Este comportamento sugere uma sensibilidade da função de dobramento ao aumento do tamanho da tabela na operação de inserção.

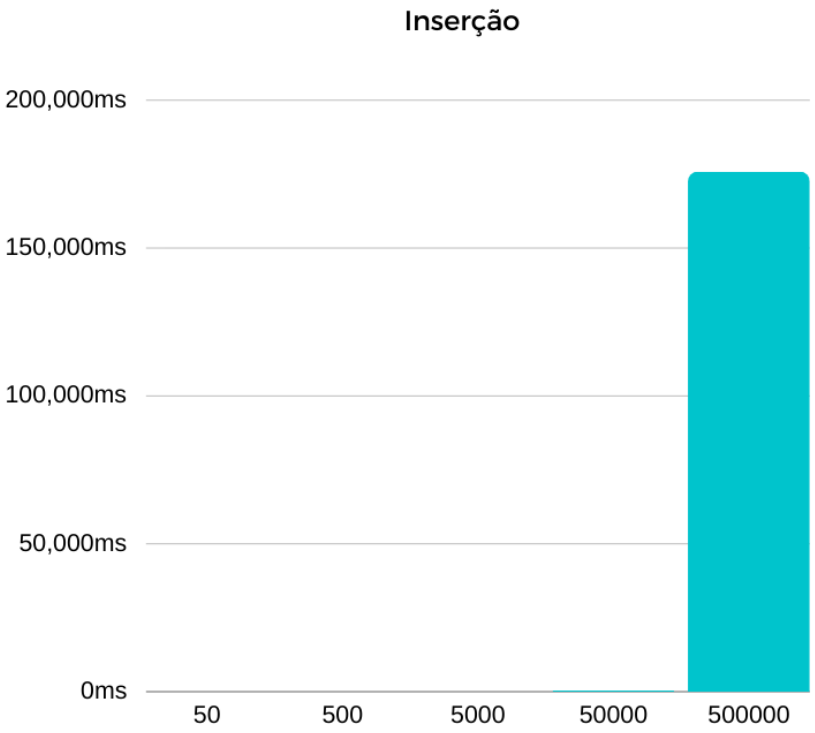

Figure 5: Resultado gráfico do tempo de inserção no Dobramento Hash

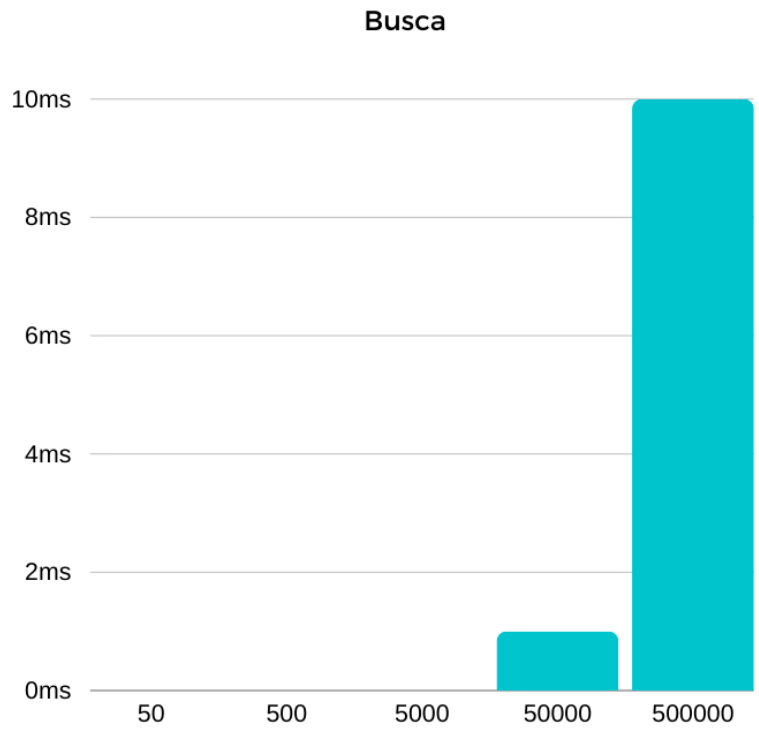

Figure 6: Resultado gráfico do tempo da busca no Dobramento Hash

6.4 Média de tempo dos testes

	Inserção	Busca
Divisão	5ms	1.2ms
Multiplicação	24.4ms	12.8ms
Dobramento	35.235,2ms	2.2ms

Table 4: Nota-se que a Função Hash de Dobramento é a que apresenta um maior tempo de inserção, devido ao aumento exponencial do número de colisões.

7 Conclusão

Ao analisar os resultados dos testes para as funções hash de multiplicação, dobramento e resto de divisão, observa-se que a função de resto de divisão apresenta desempenho mais consistente em diferentes tamanhos de tabela. A função de multiplicação demonstrou tempos de inserção e busca crescentes, tornando-se mais significativos em tamanhos maiores. Já a função de dobramento mostrou um aumento expressivo no tempo de inserção à medida que a tabela cresce, enquanto o tempo de busca permaneceu relativamente baixo. Portanto, considerando eficiência e consistência, a função de resto de divisão destaca-se como uma escolha mais robusta para a implementação de tabelas hash.