Relatório de Análise de Funções de Hash e Desempenho em Diferentes Tamanhos de Tabela

Alunos: Andrei Silva e Vitor Coradin
Professor: Andrey

Objetivo

Este relatório visa analisar o desempenho de três funções de hash (`resto`, `multiplicação` e `dobramento`) em tabelas de hash de tamanhos variados (10, 100 e 1000) para diferentes tamanhos de dados (1.000.000, 5.000.000 e 20.000.000). As métricas analisadas incluem número de colisões, tempo de inserção, tempo de busca e número de comparações necessárias para realizar as operações.

Metodologia

O programa realiza uma série de inserções e buscas em diferentes configurações de tabela e função de hash. Cada operação é testada cinco vezes para obter uma média representativa. O código em Java utiliza as seguintes funções de hash:

1. **Função de Resto**: Realiza o hash pela operação de módulo (`chave % tamanho_da_tabela`).
2. **Função de Multiplicação**: Aplica uma constante de multiplicação recomendada (`A = 0.6180339887`) antes de calcular o índice.
3. **Função de Dobramento**: Soma segmentos da chave dividida e aplica módulo sobre o tamanho da tabela.

Resultados

Tamanho da Tabela	Tamanho dos Dados	Função Hash	Colisões	Tempo de Inserção (ms)	Tempo de Busca (ms)	Comparações
10	1.000.000	Resto	49.999.588.250	56,98	0,21	6
10	1.000.000	Multiplicação	50.000.146.943	206,04	0,08	6
100	1.000.000	Dobramento	4.999.915.719	19,78	0,07	5
1000	5.000.000	Multiplicação	12.500.007.475	800,98	0,04	5

Table 1: Resultados de desempenho das funções de hash

Análise de Desempenho

1. **Colisões**: Observou-se um aumento substancial no número de colisões ao utilizar tabelas menores com funções de hash menos eficientes. A função de `resto`, embora simples, gerou menos colisões em muitos casos comparada à `multiplicação`.

2. **Tempo de Inserção e Busca**: A função `multiplicação` apresentou tempos de inserção mais altos. A função de `dobramento`, em geral, obteve tempos de inserção e busca mais curtos para tabelas pequenas.

3. **Comparações**: A média de comparações foi estável em 5 ou 6, independentemente do tamanho da tabela, mas a função `dobramento` se destacou, reduzindo ligeiramente o número de comparações necessárias em buscas.

Conclusão

Dentre as funções testadas, a função `dobramento` se mostrou mais eficiente em tabelas menores, oferecendo um tempo de execução mais rápido e um número de colisões reduzido em algumas configurações. A função `multiplicação` tem desempenho moderado, porém com maior custo de tempo para inserções. Em grandes volumes de dados, a função `resto` pode ser uma escolha vantajosa pela simplicidade.