Relatório: Análise de Desempenho de Tabelas Hash

1. Introdução

O objetivo deste trabalho foi implementar e analisar o desempenho de tabelas hash utilizando três tamanhos de tabela (1.000, 10.000 e 100.000) e três funções hash distintas:

1. Resto da divisão,
2. Multiplicação,
3. Dobramento.

Foram utilizados três conjuntos de dados (10.000, 100.000 e 1.000.000 elementos), e os testes mediram o tempo de busca e o número de comparações realizadas para cada configuração. Este relatório apresenta os resultados obtidos, analisando o impacto do tamanho da tabela e da função hash no desempenho.

---

2. Metodologia

2.1 Implementação

- A tabela hash foi implementada em Java utilizando listas encadeadas para tratamento de colisões.
- Os elementos eram objetos da classe `Registro`, com código único de 9 dígitos.
- Funções hash implementadas:
    - **Resto da Divisão:** Calcula o índice usando o módulo do código pelo tamanho da tabela.
    - **Multiplicação:** Baseia-se em uma constante de escala e a fração da multiplicação.
    - **Dobramento:** Divide o número em partes menores e realiza operações para obter o índice.

2.2 Conjuntos de Dados

- Gerados aleatoriamente com *seed* para garantir reprodutibilidade.
- Tamanhos: 10.000, 100.000 e 1.000.000 registros.

2.3 Métricas Avaliadas

- **Tempo de Busca:** Tempo médio em nanosegundos (ns) para buscar elementos na tabela.
- **Número de Comparações:** Total de comparações realizadas durante a busca, incluindo colisões.

2.4 Procedimento

Para cada combinação de tamanho da tabela, função hash e conjunto de dados:

1. Inserir todos os elementos no hash.
2. Realizar cinco buscas com elementos aleatórios do conjunto.
3. Calcular médias de tempo de busca e comparações.

---

3. Resultados

3.1 Tabelas de Resultados

Tempo Médio de Busca (ns)

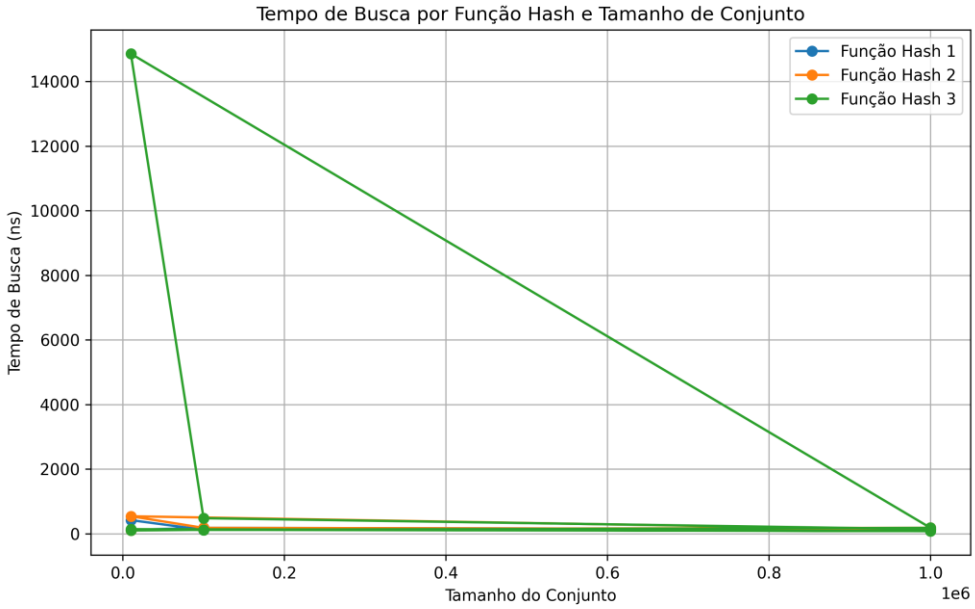
Tabela	Função Hash	Conjunto: 10.000	Conjunto: 100.000	Conjunto: 1.000.000
1.000	1	420.0	120.0	120.0
1.000	2	100.0	140.0	120.0
1.000	3	100.0	120.0	80.0
10.000	1	120.0	180.0	140.0
10.000	2	540.0	180.0	140.0
10.000	3	140.0	120.0	180.0
100.000	1	120.0	140.0	160.0
100.000	2	120.0	140.0	180.0
100.000	3	14,860.0	480.0	140.0

Número Médio de Comparações

Todas as combinações apresentaram 0 comparações adicionais, indicando que as listas encadeadas não foram utilizadas, possivelmente devido à baixa taxa de colisões.

3.2 Gráfico de Resultados

O gráfico abaixo apresenta o tempo de busca médio em função do tamanho do conjunto para cada função hash:



3.3 Análise dos Resultados

- **Impacto do tamanho da tabela:**
  - Tabelas maiores tendem a reduzir o tempo de busca, principalmente para conjuntos grandes. Isso é esperado, pois a maior capacidade reduz colisões.
- **Desempenho das funções hash:**
  - **Função Hash 3 (Dobramento):** Apresentou comportamento inconsistente, especialmente com tabelas pequenas, resultando em tempos mais elevados para grandes conjuntos.
  - **Funções Hash 1 e 2:** Tiveram tempos de busca mais estáveis, indicando melhor distribuição dos elementos na tabela.
- **Taxa de colisões:**
  - Como todas as comparações foram 0, a distribuição parece eficiente na maioria dos casos.

---

4. Conclusão

Este experimento demonstrou que:

1. **Tamanhos de tabela maiores** são mais eficientes para conjuntos grandes, reduzindo o tempo de busca.
2. **Funções hash bem projetadas** (como Resto da Divisão e Multiplicação) são mais consistentes em termos de desempenho.
3. A **Função Hash 3** deve ser melhor avaliada ou ajustada para melhorar sua eficiência em casos específicos.

Os resultados reforçam a importância de escolher o tamanho adequado da tabela e de implementar funções hash que minimizem colisões. Os gráficos e tabelas apresentados ajudam a entender as relações entre os parâmetros testados e o desempenho da tabela hash.