PJBL3

Relatório de Análise de Tabelas Hash

Contexto

O experimento envolveu a implementação de duas variantes de tabelas hash em Java, ambas estendendo a classe abstrata `AbstractHashTable`.

Implementações

1. **HashTable1**
   - Utiliza o algoritmo **FNV-1a 32-bit** com um leve mix final.
   - O índice é calculado com `modPositivo` para garantir valores não-negativos.
2. **HashTable2**
   - Utiliza o algoritmo **djb2** (`h*33 ^ c`) com mistura adicional.
   - Também aplica `modPositivo` para o cálculo do índice.

Ambas as implementações têm capacidade máxima limitada a 32 posições, conforme a regra fornecida.

Resultados Obtidos

Métrica	Tabela 1 (FNV-1a)	Tabela 2 (djb2)
Tempo de inserção (ns)	52.300.811	69.192.761
Tempo de busca (ns)	32.457.969	38.405.097

Análise de Desempenho

1. **Tempo de Inserção**
   - A **Tabela 1** apresentou melhor desempenho, sendo aproximadamente **32,29% mais rápida** que a Tabela 2.
   - A diferença pode ser atribuída à eficiência do hash FNV-1a, que realiza menos operações aritméticas em comparação ao djb2 com mix.
2. **Tempo de Busca**
   - Novamente, a **Tabela 1** teve desempenho superior, com um tempo de busca cerca de **18,32% menor** que a Tabela 2.

- Isso sugere que a distribuição dos elementos da Tabela 1 é ligeiramente mais uniforme, resultando em **listas encadeadas menores por bucket**.

Conclusões

- **HashTable1 (FNV-1a)** apresenta **melhor desempenho** geral, tanto na inserção quanto na busca.
- **HashTable2 (djb2)**, embora funcional, é ligeiramente menos eficiente nesse cenário de capacidade limitada.
- Para tabelas hash pequenas (até 32 buckets), **FNV-1a é recomendado**, a ter acesso mais rápido.

Observações Finais

- A limitação da tabela a 32 buckets cria um ambiente de teste onde a **distribuição do hash** é crítica.
- Em aplicações com tabelas maiores, a diferença entre os dois hashes pode variar, dependendo do tamanho e padrão das chaves.