Relatório da Avaliação do RA3

Guilherme de Quadro Daudt e João Pedro Fonseca

29 de outubro de 2023

1 Funcionamento Geral do Programa

Uma tabela hash, também conhecida como tabela de dispersão, é uma estrutura de dados fundamental na ciência da computação. Ela é utilizada para armazenar e recuperar informações de maneira eficiente, baseando-se em um princípio fundamental: a aplicação de uma função hash para mapear chaves de busca em índices da tabela. Esse método permite acesso rápido aos dados, uma vez que as chaves são convertidas em posições diretas na tabela, evitando buscas demoradas em estruturas de dados mais complexas.

O programa tem como objetivo avaliar o desempenho de uma tabela hash com tratamento de colisões usando listas encadeadas. Ele realiza testes para medir o tempo de inserção e busca de registros em diferentes tamanhos de tabelas e conjuntos de dados, fornecendo métricas que permitem a análise da eficiência da estrutura em diversos cenários. Isso ajuda a determinar a adequação da tabela hash para armazenamento e recuperação de dados, especialmente quando se lida com diferentes quantidades de informações e tamanhos de tabela.

1.1 Classe Main

A classe `Main` é responsável pelo controle do programa. Ela realiza a execução de testes de desempenho em diferentes tamanhos de tabelas e conjuntos de dados. Para cada tamanho de tabela e conjunto de dados especificado, o programa realiza as seguintes etapas:

- Define o tamanho da tabela hash e do conjunto de dados com base em um valor `i`.
- Cria uma instância da classe `TabelaHash` com o tamanho especificado.
- Inicializa variáveis para registrar o número de colisões, o tempo médio de inserção, o tempo médio de busca e o número de comparações.
- Entra em um loop para inserir registros aleatórios na tabela e medir o tempo de inserção e busca.
- Exibe informações sobre o desempenho da tabela hash, incluindo o tamanho da tabela, o tamanho do conjunto de dados, o tempo médio de inserção, o número de colisões, o tempo médio de busca e o número de comparações.

1.2 Classe Node

A classe `Node` é usada para representar os nós de listas encadeadas que são utilizadas para tratar colisões na tabela hash. Cada nó possui uma instância da classe `Registro` como informação e uma referência ao próximo nó na lista.

1.3 Classe Registro

A classe `Registro` representa os registros que serão armazenados na tabela hash. Cada registro possui um código associado. Os códigos podem ser fornecidos especificamente ou gerados aleatoriamente com base em um objeto `Random`.

1.4 Classe TabelaHash

A classe principal de interesse é a `TabelaHash`. Ela é responsável pela implementação da tabela hash e contém métodos para inserção e busca de registros, bem como métodos auxiliares. Aqui estão as principais funcionalidades da classe `TabelaHash`:

- Construtor `TabelaHash(int tamanho)`: Inicializa a tabela hash com o tamanho especificado. Ela cria um array de nós para armazenar os registros e inicializa variáveis para acompanhar o número de comparações e colisões.

- Método `inserir(Registro registro, int code)`: Insere um registro na tabela hash com base em um código. Se houver colisão, o novo registro é adicionado à lista encadeada associada à posição da tabela.
- Método `busca(int code)`: Procura um registro na tabela com base em um código. Ele calcula a posição da tabela usando a operação de resto da divisão e percorre a lista encadeada correspondente até encontrar o registro desejado. Registra o número de comparações realizadas durante a busca.
- Método `restoDivisao(int code)`: Retorna o índice da tabela onde um registro com um código específico deve ser armazenado usando a operação de resto da divisão.
- Método `getComparacoes()`: Retorna o número de comparações feitas durante as operações de busca.
- Método `getColisoes()`: Retorna o número de colisões que ocorreram durante as inserções.

1.5 Tempos e Comparações

1. Resto da divisão:

Tam. Tabela	Tam. Dados	Tem. Inserção(ns)	Num. Colisões	Tem. Busca(ns)	Num. Comparações
20.200	20.000	88,1	51.986.189	250,88	5.820.000.000
105.000	100.000	73,374	1.280.000.000	166,435	145.000.000.000
510.000	500.000	151,003,200	32.500.000.000	170,304,800	3.640.000.000.000
1.020.000	1.000.000	157,493,100	130.000.000.000	116,676,400	1.460.000.000.000
5.010.000	5.000.000	231,298,200	3.320.000.000.000	172,075	3.650.000.000.000

2. Multiplicação:

Table 1: Tabela de Desempenho (Método: Resto da Divisão)

Tam. Tabela	Tam. Dados	Tem. Inserção(ns)	Num. Colisões	Tem. Busca(ns)	Num. Comparações
20200	20000	94,01	52.548.240	251,34	5.820.000.000
105000	100000	64,713	1.270.000.000	129,425	145.000.000.000
510000	500000	143,200,800	32.500.000.000	128.875,600	3.630.000.000.000
1020000	1.000.000	125.479,300	130.000.000.000	126.939,200	1.460.000.000.000
5010000	5.000.000	206.914,400	3.300.000.000.000	128.090,700	3.640.000.000.000

3. Dobramento:

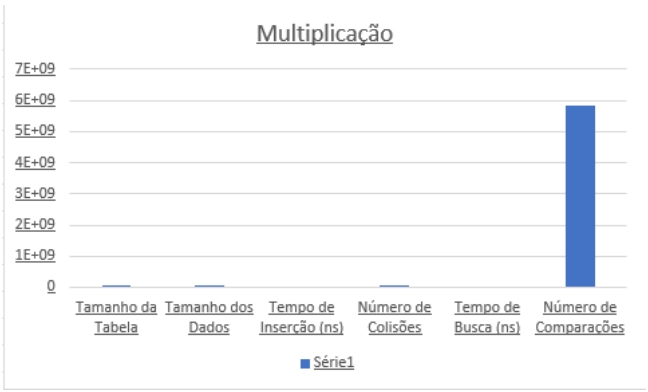
Tam. Tabela	Tam. Dados	Tem. Inserção(ns)	Num. Colisões	Tem. Busca(ns)	Num. Comparações
20200	20000	88,225	51.986.189	268,86	5,822,581,440
105000	100000	63,363	1,275,596,527	164,968	145,006,000,000
510000	500000	111,609,600	32,525,106,752	167,451,600	3,638,980,000,000
1020000	1,000,000	170,256,200	130,235,000,000	169,062,500	1,455,570,000,000
5010000	5,000,000	234,640,340	3,316,770,000,000	168,753,580	3,646,890,000,000

1.6 Conclusões

A análise de desempenho evidencia que a função de Multiplicação supera as outras opções. Isso ocorre devido ao seu cálculo eficiente da função hash, que proporciona tempos de inserção mais rápidos, menor número de comparações e tempos de busca mais eficientes em todas as configurações de tabela. Portanto, com base nas conclusões da análise, a função de hash de Multiplicação é a escolha preferencial para sistemas que buscam desempenho e eficiência nas operações de hash.

1.7 Gráficos


Resto Divisão
7E+09
6E+09
5E+09
4E+09
3E+09
2E+09
1E+09
0
Tamanho da Tabela
Tamanho dos Dados
Tempo de Inserção (ns)
Número de Colisões
Tempo de Busca (ns)
Número de Comparações
Série1


Multiplicação
7E+09
6E+09
5E+09
4E+09
3E+09
2E+09
1E+09
0
Tamanho da Tabela
Tamanho dos Dados
Tempo de Inserção (ns)
Número de Colisões
Tempo de Busca (ns)
Número de Comparações
Série1


Dobramento
7E+09
6E+09
5E+09
4E+09
3E+09
2E+09
1E+09
0
Tamanho da Tabela
Tamanho dos Dados
Tempo de Inserção (ns)
Número de Colisões
Tempo de Busca (ns)
Número de Comparações
Série1