Tabela de Espalhamento, Tabela Hash

1 Definição

Uma Tabela de Espalhamento é uma estrutura de dados que associa chaves a valores. Utiliza uma função de hash para calcular um índice em uma matriz de slots ou buckets, onde o valor desejado pode ser encontrado.

2 Função de Hash

A função de hash é um algoritmo que transforma a chave em um índice. A qualidade da função de hash é crucial para a eficiência da tabela, pois uma boa função distribui as chaves uniformemente pelos buckets.

Evitar colisões em tabelas de espalhamento é essencial para manter a eficiência dessa estrutura de dados. Aqui estão algumas técnicas e estratégias para minimizar colisões:

2.1 Escolha de uma Boa Função de Hash

Distribuição Uniforme: A função de hash deve distribuir as chaves uniformemente pelos buckets.
Números Primos: Usar números primos no cálculo do hash pode ajudar a reduzir colisões.

2.2 Técnicas de Tratamento de Colisões

Encadeamento Separado: Cada bucket aponta para uma lista ligada de entradas. Quando ocorre uma colisão, o novo elemento é adicionado à lista correspondente.

Endereçamento Aberto: Procura-se o próximo bucket disponível de acordo com uma sequência predefinida. Existem várias estratégias de endereçamento aberto:

Sondagem Linear: Procura sequencialmente o próximo bucket disponível.

Sondagem Quadrática: Usa uma função quadrática para determinar o próximo bucket.

Hash Duplo: Aplica uma segunda função de hash para determinar o próximo bucket.

2.3 Redimensionamento da Tabela

Tabela Dinâmica: Aumentar o tamanho da tabela e recalcular os hashes quando a carga (número de elementos/buckets) atinge um certo limite. Isso ajuda a manter a eficiência da tabela.

2.4 Funções de Hash Perfeitas

Hashing Perfeito: Se todas as chaves são conhecidas de antemão, é possível criar uma função de hash que evita colisões completamente. No entanto, isso é raro e difícil de implementar na prática.

2.5 Uso de Estruturas Auxiliares

Bloom Filters: Uma estrutura probabilística que pode ser usada para testar se um elemento está presente em um conjunto, ajudando a reduzir colisões em algumas aplicações específicas.

2.6 Exemplo Prático

Encadeamento Separado: Imagine uma tabela de espalhamento onde cada bucket é uma lista ligada. Se duas chaves diferentes, “chave1” e “chave2”, resultarem no mesmo índice, ambas serão armazenadas na lista ligada do bucket correspondente. Isso permite que a tabela lide eficientemente com colisões sem perder dados.

Implementar essas técnicas pode ajudar a manter a eficiência e a performance da sua tabela de espalhamento

3 Colisões

Colisões ocorrem quando duas chaves diferentes são mapeadas para o mesmo índice. Existem várias técnicas para resolver colisões, como:

Encadeamento: Cada bucket aponta para uma lista ligada de entradas.

Endereçamento Aberto: Procura-se o próximo bucket disponível de acordo com uma sequência predefinida.

3.1 Fator de carga

O fator de colisão em uma tabela de espalhamento, também conhecido como **fator de carga** ou **load factor**, é uma métrica que indica a densidade de ocupação da tabela. Ele é calculado pela razão entre o número de elementos armazenados na tabela e o número total de buckets disponíveis. Aqui estão os principais pontos sobre o fator de colisão:

3.1.1 Fator de carga

O fator de colisão é dado pela fórmula:

$\alpha = m / n$

onde:

- (n) é o número de elementos armazenados na tabela.
- (m) é o número total de buckets na tabela.

3.1.2 Importância

- **Desempenho**: Um fator de colisão alto indica que muitos buckets estão sendo compartilhados por múltiplos elementos, aumentando a probabilidade de colisões e, consequentemente, o tempo de busca, inserção e remoção.

- **Redimensionamento**: Quando o fator de colisão atinge um certo limite, pode ser necessário redimensionar a tabela para manter a eficiência. Isso geralmente envolve aumentar o número de buckets e recalcular os hashes de todos os elementos.

3.1.3 Valores Típicos

- **Baixo Fator de Colisão**: Geralmente, um fator de colisão abaixo de 0.7 é considerado bom, pois indica que a maioria dos buckets contém no máximo um elemento.
- **Alto Fator de Colisão**: Valores acima de 1 indicam que há mais elementos do que buckets, o que pode levar a um aumento significativo no número de colisões.

3.1.4 Exemplo Prático

- Se uma tabela de espalhamento tem 100 buckets e armazena 70 elementos, o fator de colisão será:

  $\alpha = 100 / 70 = 0.7$

- Se a tabela armazena 150 elementos, o fator de colisão será:

  $\alpha = 100 / 150 = 1.5$

3.1.5 Gerenciamento

- **Redimensionamento Dinâmico**: A tabela pode ser redimensionada (geralmente dobrando o número de buckets) quando o fator de colisão atinge um certo limite, como 0.75 ou 1.0.
- **Função de Hash Eficiente**: Usar uma função de hash que distribua os elementos uniformemente ajuda a manter um fator de colisão baixo.

Manter um fator de colisão adequado é crucial para garantir a eficiência da tabela de espalhamento.

4 Operações Básicas

•**Inserção**: Adiciona um par chave-valor na tabela.

•**Busca**: Recupera o valor associado a uma chave.

•**Remoção**: Remove o par chave-valor da tabela.

5 Vantagens

•**Velocidade**: Operações de busca, inserção e remoção são, em média, muito rápidas, com complexidade O(1).

•**Flexibilidade**: Pode armazenar qualquer tipo de dado, desde que a chave possa ser transformada em um índice.

6. Desvantagens

•**Colisões**: Podem degradar o desempenho se não forem bem gerenciadas.

•**Espaço**: Pode consumir mais memória do que outras estruturas de dados, especialmente se a tabela for esparsa.

7. Aplicações

•**Dicionários**: Implementação de dicionários em linguagens de programação.

•**Caches**: Armazenamento temporário de dados para acesso rápido.

•**Bancos de Dados**: Índices para acesso rápido a registros.