TABELAS

# Organização interna:

Resumidamente, as tabelas de dados exibem informações em um formato semelhante a uma grade de linhas e colunas. Elas organizam as informações de uma maneira fácil de verificar, permitindo que os usuários possam procurar padrões e desenvolver insights a partir dos dados.

A partir dessas informações, podemos tratar do hashing, que é uma técnica usada para identificar, exclusivamente, um objeto específico de um grupo de objetos semelhantes.

A ideia do hash é distribuir entradas (pares chave/valor) uniformemente em uma matriz e a cada elemento é atribuída uma chave (chave convertida). Usando essa chave, podemos acessar o elemento em tempo O(1). Ainda usando a chave, o algoritmo (função hash) calcula um índice que sugere onde uma entrada pode ser encontrada ou inserida. O hash é implementado em duas etapas:

* Um elemento é convertido em um inteiro usando uma função hash. Esse elemento pode ser usado como índice para armazenar o elemento original, que cai na tabela hash;
* O elemento é armazenado na tabela hash, onde pode ser recuperado rapidamente usando a chave de hash.

# Aspectos de implementação:

Qualquer implementação de tabela hash deve conter três componentes:

* Uma boa função hash para mapear chaves para valores;
* Uma estrutura de dados de tabela hash que suporta operações de inserção, pesquisa e exclusão;
* Uma estrutura de dados para levar em conta a colisão de chaves;
* Todo o benefício de usar uma tabela hash é devido ao seu tempo de acesso muito rápido.

Considerando isso, o primeiro passo para a implementação é escolher uma função hash razoavelmente boa e que tenha uma baixa chance de colisão.

# Tabelas de espalhamento:

Técnicas para gerenciar dados com eficiência são tópicos tradicionais em ciência da computação. Além de armazenar dados, recuperá-los com eficiência é outra preocupação relevante, para tanto, algoritmos são utilizados.

Um algoritmo de hash é uma fórmula matemática que recebe determinada entrada de dados e gera um valor de comprimento fixo chamado valor de hash, que atua como uma representação resumida do valor original. Mesmo com o melhor algoritmo fazendo algum processamento de dados específico, teremos um desempenho ruim se o gerenciamento de dados não for otimizado.

Linha do tempo

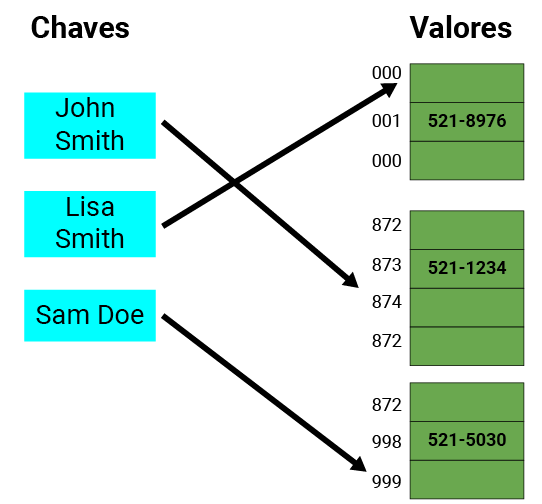
Descrição gerada automaticamente

É relevante notar que a função hash pode aumentar ou diminuir o número de bytes da entrada de comprimento variável. Assim, se a entrada for maior que o código hash, o número de bytes diminuirá. Caso contrário, aumentará.

## Funções de espalhamento:

Resumindo, o hashing tem várias aplicações em ciência da computação, como armazenar e verificar senhas, criar assinaturas de mensagens e fornecer estruturas de gerenciamento de dados.

As tabelas hash são estruturas de dados que associam chaves específicas a valores correspondentes. Essas tabelas geralmente são implementadas com uma matriz associativa para armazenar os dados. Além disso, elas usam uma função hash para calcular em qual ponto da matriz os dados devem ser armazenados.



Como uma das principais estruturas de dados, uma tabela hash pode ser usada para armazenar dados no formato de valor-chave com acesso direto a seus itens constantemente. Existem quatro aspectos distintos sobre como funcionam as tabelas hash:

* Com base no armazenamento;
* Com base nos pares de valor chave;
* Com base em uma função hash;
* Com base em operações de tabela.

Às vezes, os dados contêm uma propriedade individual que pode naturalmente assumir a responsabilidade pela chave. Por exemplo, se presumirmos que um usuário tem um e-mail e um nome de usuário, o e-mail ou o nome de usuário podem ser usados como a chave se forem garantidos como exclusivos. Pensando em todos os aspectos sobre as funções de hash, também temos outras características marcantes:

* Deve gerar valores de hash diferentes para a string semelhante;
* É fácil de entender e simples de calcular;
* Deve produzir as chaves que serão distribuídas uniformemente em uma matriz;
* Um número de colisões deve ser menor ao colocar os dados na tabela hash;
* É considerada perfeita quando faz uso de todos os dados de entrada.

## Tratamento de colisões:

Como as funções de hash mapeiam chaves de tamanho variável para índices de tamanho fixo, elas mapeiam um conjunto infinito para um finito. Dessa forma, as colisões eventualmente ocorrerão. Ou seja, quando a função hash gera o mesmo índice para várias chaves, haverá um conflito.

Para resolver uma colisão colocando o item em outro lugar é a prática chamada de Open Addressing, ou Endereçamento Aberto, que é uma forma para resolver as colisões utilizando encadeamento.

Quando o endereço calculado estiver ocupado, então a pesquisa linear é usada para encontrar o próximo bucket disponível. Se a sondagem linear terminar e ainda não tiver encontrado uma posição, esta poderá circular em torno do início da matriz e continuar pesquisando a partir daí.

Quanto mais itens houver em uma tabela hash, maiores são as chances de uma colisão. Uma maneira de lidar com isso é tornar a tabela hash maior para a quantidade total de dados que se espera. Colisões podem ocorrer, mas podem ser significativamente reduzidas com a ajuda de várias técnicas de resolução de colisões. A seguir indicaremos e analisaremos as mais relevantes:

* Open Hashing ou Hash aberto (encadeamento separado);
* Closed Hashing ou Hash fechado (endereçamento aberto);
* Linear Probing ou Sondagem Linear;
* Quadratic Probing ou Sondagem quadrática;
* Double Hashing ou Hash duplo.

**Open Hashing** - As colisões são resolvidas usando uma lista de elementos para armazenar objetos com a mesma chave juntos. O tamanho da tabela pode ser significativamente grande.

**Closed Hashing** - Durante a inserção, se uma colisão for encontrada, células alternativas são tentadas até que um bucket vazio seja encontrado.

## Problemas envolvendo hashing

Embora as pesquisas de tabela hash usem tempo constante em média, o tempo gasto pode ser significativo. Avaliar uma boa função hash pode ser uma operação lenta.

As tabelas hash, em geral, exibem baixa localidade de referência, ou seja, os dados a serem acessados são distribuídos aparentemente de forma aleatória na memória.

Como as tabelas hash causam padrões de acesso que variam, isso pode acionar faltas de cache do microprocessador que causam longos atrasos.