Definição

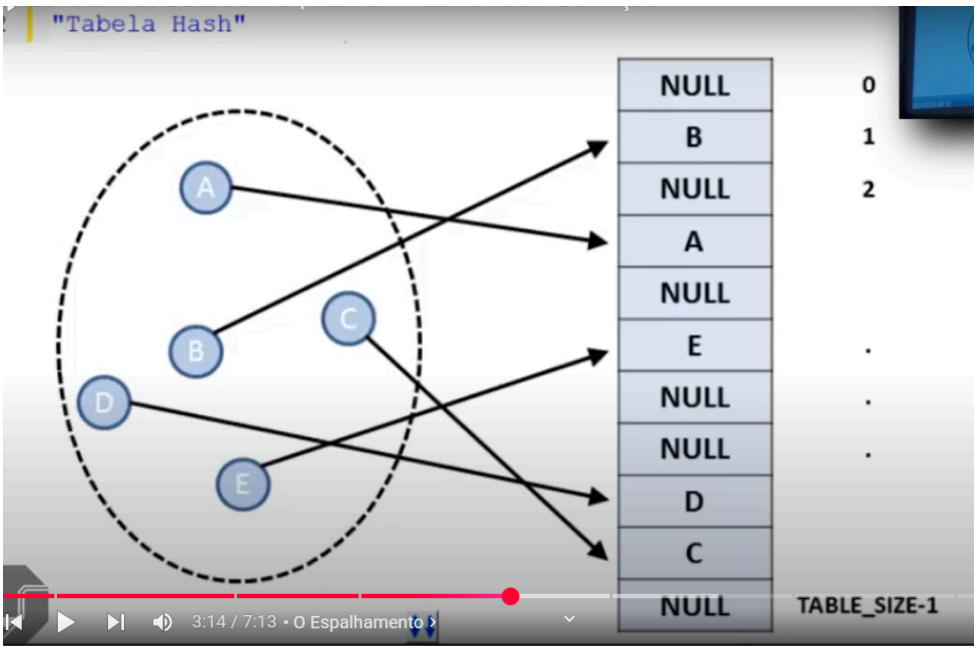
Em geral, os métodos de busca comparam o valor de seus componentes. O problema é que, para algoritmos eficientes, os elementos precisam estar ordenados, mas o custo total de ordenar e depois buscar pode ser alto.

Em uma busca ideal, o acesso ao elemento procurado é direto,sem nenhuma etapa de comparação de chaves: custo “O (1)”.

Se olharmos para um array: são estruturas que utilizam “índices” para armazenar informações, ou seja, permite acessar uma determinada posição do array com custo “O (1)”. A **Tabela Hash** surge para aproveitar esse mecanismo do array para auxiliar na busca de elementos.

Tabela Hash:

- *Também conhecidas como tabelas de “indexação” ou de “espalhamento”;*
- *É uma generalização da ideia de “array”;*
- *Utiliza uma função para espalhar os elementos que queremos armazenar na tabela;*
- *O espalhamento faz com que os elementos fiquem dispersos de forma não ordenada dentro do “array” que define a tabela;*
- *FUNÇÃO DE HASHING: função que espalhar os elementos na tabela;*



Espalhar os elementos melhora a busca, pois a tabela permite a associação de **‘valores’** a **‘chaves’** ->

CHAVE: Parte da informação que compõe o elemento a ser inserido ou buscado na tabela;

VALOR: È a posição (índice) onde o elemento se encontra no array que define a tabela;

Dessa forma, a partir de uma chave podemos acessar de forma rápida uma determinada posição do array.

Na média, essa operação tem custo “O(1)”

Vantagens

- Alta eficiência na operação de busca;
- Tempo de busca é praticamente independente do número de “chaves” armazenadas na tabela;
- Implementação simples;

Desvantagens

- Alto custo para recuperar os elementos da tabela ordenados pela chave. Nesse caso, é preciso ordenar a tabela;
- Pior caso é “O(N)”, sendo N o tamanho da tabela: alto número de colisões;

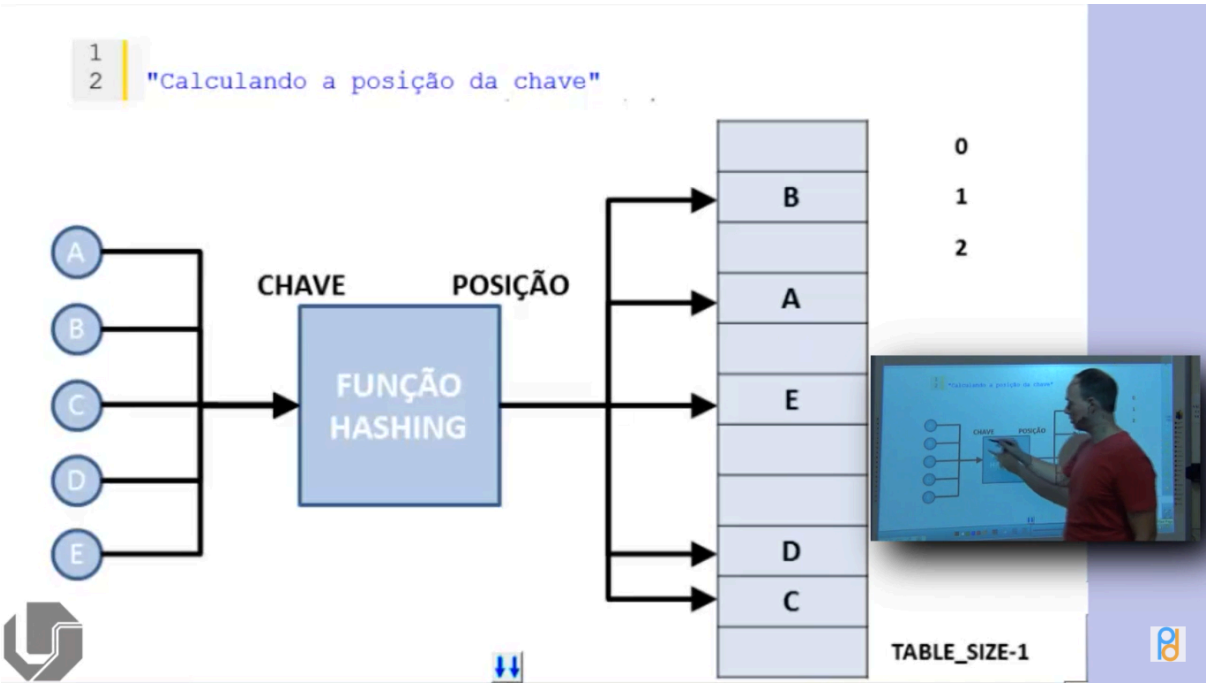
Aplicações

- Busca em elementos em base de dados;
- Criptografia: MD5 e família SHA (Secure Hash Algorithm);
- Implementação da tabela de símbolos dos compiladores;
- Armazenamento de senhas com segurança: a senha não é armazenada no servidor, mas sim o resultado da função hash;
- Verificação de integridade de dados e autenticação de mensagens;

Implementação

- Utiliza uma estrutura similar a da “Lista Sequencial Estática”;
- Utiliza um “array” para armazenar os elementos;
- Limita o número de elementos que podemos armazenar;

Calculando a posição da chave:



Tenho meus dados, pega a chave dos dados, joga na função de hashing, ela calcula uma posição e ai, coloca os dados na tabela, não é de forma ordenada, é de forma aleatória;

Método da divisão ou da Congruência Linear

- Consiste em calcular o resto da divisão do valor inteiro que representa o elemento pelo tamanho ta tabela, TABLE_SIZE;
- Função simples e direta

Método da Multiplicação (Ou da Congruência Linear Multiplicativo)

- Usa um valor “A”, “0 < A < 1”, para multiplicar o valor da “chave” que representa o elemento. Em seguida, a “parte fracionária” resultante é multiplicada pelo tamanho da tabela para calcular a posição do elemento;

Tratando uma string como uma chave

- Calcular um valor numérico a partir dos valores ASCII dos caracteres;
- Devemos considerar a posição da letra
- Cuidado: Não devemos apenas somar os caracteres, pois palavras com letras trocadas irão produzir o mesmo valor;

```
int valorString (char *str)
{
        int i, valor = 7;
        int tam = strlen(str);
```

```
        for (i = 0; i < tam; i++)
        {
                valor = 31 * valor + int str [i];
        }
        return valor;
}
```

Inserção e Busca sem tratamento de colisão

Inserção

- Calcular a posição da cave no array;
- Alocar espaço para os dados;
- Armazenar os dados na posição calculada

Problema:

A função de hashing está sujeita ao problema de gerar posições iguais para chaves diferentes;

Por se tratar de uma função determinística, ela pode ser manipulada de forma determinística, ela pode ser manipulada de forma indesejada;

Conhecendo a função de hashing, pode-se escolher as chaves de entrada de modo que todas colidam, diminuindo o desempenho na busca para “O (N)”

Hashing Universal

- É uma estratégia que busca minimizar esse problema de colisões;
- Basicamente, devemos escolher aleatoriamente (em tempo de execução) a função que será utilizada;
- Para tanto, construimos um conjunto, ou família, de funções hashing;