Hashing
(Pesquisa por Cálculo de Endereço)

Prof. Alberto Costa Neto

Pesquisa por Cálculo de Endereço (Hashing)

Baseia-se na idéia de calcular o endereço de armazenamento do dado a partir do valor da chave

> *Aplica-se uma* ***função de cálculo de endereço (função Hashing) sobre a chave, obtendo-se como resultado o endereço de armazenamento na tabela.***

Este método é mais do que um método de pesquisa.

- É um **método de organização física das tabelas**.

Função Hashing

A **função hashing ideal** seria aquela que:

- **gerasse um endereço diferente para cada um dos N diferentes valores das chaves** presentes na tabela

Entretanto, este tipo de função é difícil de ser obtida.

- As **funções normalmente utilizadas podem provocar colisões**, ou seja, eventualmente atribuem um mesmo endereço a diferentes valores de chave.

Função Hashing

Uma das funções hashing mais utilizadas é:

Hash(chave) = chave % TamanhoDaTabela

- Calcula o resto da divisão da chave pelo tamanho da tabela

Para chave **201805191115** e uma tabela com tamanho **100:**

- Teríamos o **endereço 15** ao aplicar a função Hash

Inclusão na Tabela Hash

Como dispor as chaves na tabela

Tamanho = 10

Chave										
Endereço	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Passo 1: Calcular o endereço usando a função Hash

Passo 2: Se a posição nunca foi usada, pode incluir na posição.

Incluindo Chaves na Tabela

Incluir Chave 35

Passo 1: hash(35) => 5

Passo 2: Se a posição nunca foi usada, pode incluir na posição.

Chave						35				
Endereço	0	1	2	3	4	5	6	7	8	9

Incluindo Chaves na Tabela

Incluir Chave 43

Passo 1: hash(43) => 3

Passo 2: Se a posição nunca foi usada, pode incluir na posição.

Chave				43		35				
Endereço	0	1	2	3	4	5	6	7	8	9

Incluindo Chaves na Tabela

Incluir Chave 8044

Passo 1: hash(8044) => 4

Passo 2: Se a posição nunca foi usada, pode incluir na posição.

Chave				43	8044	35				
Endereço	0	1	2	3	4	5	6	7	8	9

Incluindo Chaves na Tabela

Incluir Chave 999

Passo 1: hash(999) => 9

Passo 2: Se a posição nunca foi usada, pode incluir na posição.

Chave				43	8044	35				999
Endereço	0	1	2	3	4	5	6	7	8	9

Busca na Tabela Hash

Buscando Chave na Tabela

Buscando Chave 999

Passo 1: hash(999) => 9

Passo 2: Se a posição estiver ocupada e a chave coincide, achou!

Chave	1000	11	222		8044	35	656	47	88	999
Endereço	0	1	2	3	4	5	6	7	8	9

Buscando Chave na Tabela

Buscando Chave 43

Passo 1: hash(43) => 3

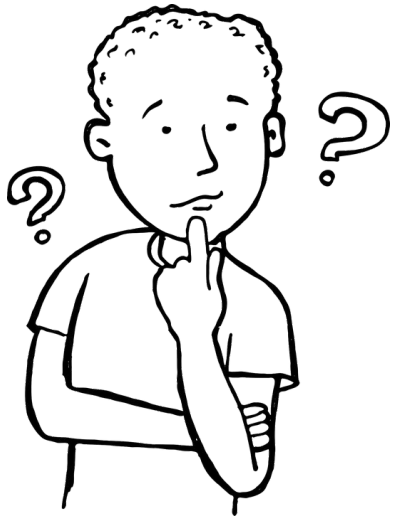
Passo 2: A posição não está ocupada. Não achou.

Chave	1000	11	222		8044	35	656	47	88	999
Endereço	0	1	2	3	4	5	6	7	8	9

Buscando Chave na Tabela

- A busca por cálculo de endereço é muito eficiente porque já obtém a posição exata na tabela.

- Porém, as colisões podem afetar este desempenho. **Como?**



E quando os endereços coincidem?

Tratamento de Colisões

Tratamento de Colisão

Quando o endereço calculado é o mesmo de outra chave já contida na Tabela Hash, ocorre o que chamamos de **Colisão**.

Há diferentes formas de tratá-la:

- Usar a **próxima posição desocupada** após a posição calculada.
- Usar uma **função hash alternativa** (que ainda pode gerar nova colisão).

Tratamento de Colisão

- Utilizaremos a solução de usar a próxima posição disponível
- Para isto, vamos marcar cada posição da tabela com uma FLAG para indicar:
  - **NuncaUsada (NU):** Posição nunca foi usada.
  - **Ocupada (O):** Está sendo ocupada por uma chave.
  - **Desocupada (D):** Está desocupada, mas já foi ocupada anteriormente.

Buscando Chave na Tabela

Buscando Chave 43

Passo 1: hash(43) => 3

Passo 2: A posição 3 nunca foi usada. Portanto, não há chance de 43 estar em outras posições. Portanto, não achou.

Chave	1000	11	222		8044	35	656	47	88	999
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	NU	O	O	O	O	O	O

Buscando Chave na Tabela

Buscando Chave 43

Passo 1: hash(43) => 3

Passo 2: A posição 3 está ocupada. Portanto, continuamos a busca sequencialmente nas próximas posições até achar a chave ou encontrar uma posição com FLAG NuncaUsada (elimina a chance de encontrar deste ponto em diante). Não achou 43.

Chave	1000	11	222	53	63	83	656	47		
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	O	O	O	O	O	NU	NU

Inserindo Chave na Tabela

Inserindo Chave 43

Passo 1: hash(43) => 3

Passo 2: A posição 3 está ocupada. Portanto, procuramos um posição que não esteja com FLAG Ocupada. Se esta posição tem FLAG NuncaUsada, podemos inserir nela. Se estiver como Desocupada, é preciso confirmar se a chave não está adiante.

Chave	1000	11	222	53	63		656	47		
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	O	O	D	O	O	NU	NU

Inserindo Chave na Tabela

Inserindo Chave 43

Passo 3: Chave 43 inserida e FLAG atualizado para Ocupada

Chave	1000	11	222	53	63	43	656	47		
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	O	O	O	O	O	NU	NU

Inserindo Chave na Tabela

Inserindo Chave 75

Passo 1: hash(75) => 5

Passo 2: A posição 5 está com FLAG Desocupada. Portanto, se 75 não existir na Tabela, esta será a posição de inserção. Porém, é preciso buscar sequencialmente o 75 até achá-lo ou encontrar uma posição com FLAG NuncaUsada. **Não pode inserir.**

Chave	1000	11	222	53	63		656	47		75
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	O	O	D	O	O	D	O

Excluindo Chave da Tabela

Excluindo Chave 45

Passo 1: hash(45) => 5

Passo 2: A posição 5 está com FLAG Ocupada e contém a chave 45.

Chave	1000	11	222	53	63	45	656	47		75
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	O	O	O	O	O	D	O

Excluindo Chave da Tabela

Excluindo Chave 45

Passo 3: Portanto, basta mudar o FLAG para Desocupada.

Chave	1000	11	222	53	63		656	47		75
Endereço	0	1	2	3	4	5	6	7	8	9
FLAG	O	O	O	O	O	D	O	O	D	O

Funções Hash para outros tipos de dados

Função Hashing (outros tipos)

A função Hash pode ser adaptada a tipos de chaves não-numéricas.

- Atribuindo um valor numérico diferente a cada letra
  - Podendo usar o código ASCII (ou não)

O importante é que valores de chaves diferentes produzam endereços diferentes.

Função Hashing (String)

Teríamos para uma chave do tipo String a seguinte função Hashing:

```
int hash(char *chave)
{
  int comprimento = strlen(chave);
  int soma = 0;
  for (int i = 0; i < comprimento; i++)
     soma += chave[i];
  return soma % TamanhoTabela;
}
```

Sugestões de Estudo

Estruturas de dados (Paulo Veloso)

- Seção 9.3

Projeto de Algoritmos com implementações em Java e C++ (Nivio Ziviani)

- Seção 5.3.1