

Tabelas de hash

Acabamos de estudar como implementar uma tabela hashing aberta e estudaremos agora como implementar uma tabela hashing fechada ou também denominada de tabela hashing com endereçamento aberto.

Neste tipo de implementação, a tabela hash é um vetor com m posições. Todas as chaves são armazenadas na própria tabela sem a necessidade de espaços extras ou ponteiros. Este método é aplicado quando o número de chaves a serem armazenadas é reduzido e as posições vazias na tabela são usadas para o tratamento de colisões.

Tabelas de hash

Quando uma chave x é endereçada na posição $h(x)$ e esta já está ocupada, outras posições vazias na tabela são procuradas para armazenar x . Caso nenhuma seja encontrada, a tabela está totalmente preenchida e x não pode ser armazenada.

Veremos duas maneiras de efetuar a busca por uma posição livre para armazenar x : tentativa linear e tentativa quadrática.

Tabelas de hash fechada – tentativa linear

Na tentativa linear, quando uma chave x deve ser inserida e ocorre uma colisão, a seguinte função é utilizada:

$$h'(x) = (h(x)+j) \bmod m$$

para $1 \leq j \leq m-1$, sendo que $h(x) = x \bmod m$

O objetivo é armazenar a chave no endereço consecutivo $h(x)+1$, $h(x)+2$, ..., até encontrar uma posição vazia.

Tabelas de hash fechada – tentativa linear

A operação de remoção é delicada, não se pode remover de fato uma chave do endereço, pois haveria perda da sequência de tentativas.

Com isso, cada endereço da tabela é marcado como livre (L), ocupado (O) ou removido (R). Livre quando a posição ainda não foi usada, ocupado quando uma chave está armazenada, e removido quando armazena uma chave que já foi removida. Para uma compreensão adequada do processo, analisaremos um exemplo.

Tabelas de hash fechada – tentativa linear

Visando manter um paralelo com o método usado anteriormente também nos utilizaremos de uma tabela com 8 entradas.

Inicialmente temos:

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | L | |

Inserir chave 16
 $16\%8=0$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | L | |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | L | |

Inserir chave 23
 $23\%8=7$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Inserir chave 41
 $41\%8=1$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | O | 41 |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | O | 41 |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Inserir chave 25
 $25\%8=1$
 $(1+1)\%8=2$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | O | 41 |
| 2 | O | 25 |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | O | 41 |
| 2 | O | 25 |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Inserir chave 39

$$\begin{aligned}39 \% 8 &= 7 \\ (7+1) \% 8 &= 0 \\ (7+2) \% 8 &= 1 \\ (7+3) \% 8 &= 2 \\ (7+4) \% 8 &= 3\end{aligned}$$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | O | 41 |
| 2 | O | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | O | 41 |
| 2 | O | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Remover chave 41
 $41 \% 8 = 1$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | O | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | O | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | O | 23 |

Remover chave 23
 $23\%8=7$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | O | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | R | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | O | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | R | 23 |

Remover chave 25

$$25\%8=1$$

$$(1+1)\%8=2$$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | R | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | R | 23 |

Tabelas de hash fechada – tentativa linear

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | R | 25 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | R | 23 |

Inserir chave 34
 $34\%8=2$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | O | 16 |
| 1 | R | 41 |
| 2 | O | 34 |
| 3 | O | 39 |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | R | 23 |

Tabelas de hash fechada: Exercício

Com base no que foi apresentado defina a(s) estrutura(s) de dados necessária(s) para a implementação de uma tabela hashing fechada com tentativa linear.

```
#define tam 8
typedef struct
{
    int chave;
    char livre; /* L = livre, O = ocupado, R = removido*/
}Hash;
typedef Hash Tabela[tam];
```

Tabelas de hash fechada: Exercício

Agora, implemente a função de inserção de uma chave na tabela hashing fechada em questão.

```
void inserir(Tabela tabela, int n) {
    int i=0;
    int pos = funcaoHashing(n);
    while (i < tam && tabela[(pos+i)%tam].livre!='L'
    && tabela[(pos+i)%tam].livre !='R')
        i = i+1;
    if (i < tam) {
        tabela[(pos+i)%tam].chave = n;
        tabela[(pos+i)%tam].livre = 'O';
    }else
        printf ("\nTabela cheia!"); }
```

Tabelas de hash fechada: Exercício

```
int funcaoHashing(int num)
{
    return num % tam;
}
```

Tabelas de hash fechada: Exercício

Implemente a função de remoção de uma chave na tabela hashing fechada em questão.

```
void remover(Tabela tabela, int n)
{
    int posicao = buscar(tabela, n);
    if (posicao < tam)
        tabela[posicao].livre = 'R';
    else
        printf ("\nElemento nao esta presente.");
}
```

Tabelas de hash fechada: Exercício

```
int buscar(Tabela tabela, int n)
{
    int i=0;
    int pos=funcaoHashing(n);
    while(i < tam && tabela[(pos+i)%tam].livre != 'L'
    && tabela[(pos+i)%tam].chave != n)
        i = i+1;
    if (tabela[(pos+i)%tam].chave == n &&
    tabela[(pos+i)%tam].livre == 'O')
        return (pos+i)%tam;
    else
        return tam; // não encontrado
}
```

Tabelas de hash fechada: Exercício

Para uma melhor fixação do tópico em estudo com base na definição do TAD abaixo implemente suas funções ainda não definidas.

```
#define tam 8
typedef struct
{
    int chave;
    char livre; // L = livre, O = ocupado, R = removido
}Hash;
typedef Hash TabelaHash [tam];
void inicializarHash(TabelaHash *);
int funcaoHashing(int);
void mostrarHash(TabelaHash);
void inserirChave(TabelaHash, int);
int buscarChave(TabelaHash, int);
void removerChave(TabelaHash, int);
```

Tabelas de hash fechada – tentativa quadrática

Na tentativa linear as chaves que colidem geram os chamados agrupamentos primários, o que aumenta o tempo de busca.

Agora estudaremos a tentativa quadrática. Neste tipo de tentativa, outro tipo de agrupamento também é gerado, denominado agrupamento secundário, mas as degradações são reduzidas se comparadas com a tentativa linear.

Tabelas de hash fechada – tentativa quadrática

Segundo Szwarcfiter (1994), para a aplicação deste método os endereços calculados pela função de hashing $h'(x, k)$ devem corresponder à varredura de toda tabela, para $k=0, \dots, m-1$. As equações recorrentes abaixo fornecem uma maneira de calcular os endereços diretamente.

$$h'(x, 0) = h(x)$$

$$h'(x, k) = (h'(x, k-1) + k) \bmod m$$

para $1 \leq k \leq m-1$, sendo que $h(x) = x \bmod m$

Tabelas de hash fechada – tentativa quadrática

Para uma melhor compreensão analisaremos agora um exemplo, também nos utilizaremos de uma tabela com 8 entradas. Inicialmente temos:

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | L | |
| 5 | L | |
| 6 | L | |
| 7 | L | |

Inserir chave 12
 $12\%8=4$

| Índice | Situação | Chave |
|--------|----------|-----------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | O | 12 |
| 5 | L | |
| 6 | L | |
| 7 | L | |

Tabelas de hash fechada – tentativa quadrática

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | O | 12 |
| 5 | L | |
| 6 | L | |
| 7 | L | |

Inserir chave 20
 $20\%8=4$
 $(4+1) \bmod 8=5$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | O | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | L | |

Tabelas de hash fechada – tentativa quadrática

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | O | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | L | |

Inserir chave 28
 $28 \% 8 = 4$
 $(4+1) \bmod 8 = 5$
 $(5+2) \bmod 8 = 7$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | O | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | O | 28 |

Tabelas de hash fechada – tentativa quadrática

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | L | |
| 3 | L | |
| 4 | O | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | O | 28 |

Inserir chave 36

$$36 \% 8 = 4$$

$$(4+1) \bmod 8 = 5$$

$$(5+2) \bmod 8 = 7$$

$$(7+3) \bmod 8 = 2$$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | O | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | O | 28 |

Tabelas de hash fechada – tentativa quadrática

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | O | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | O | 28 |

Remover chave 12
 $12 \% 8 = 4$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | R | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | O | 28 |

Tabelas de hash fechada – tentativa quadrática

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | R | 12 |
| 5 | O | 20 |
| 6 | L | |
| 7 | O | 28 |

Remover chave 20

$$20 \% 8 = 4$$
$$(4 + 1) \% 8 = 5$$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | R | 12 |
| 5 | R | 20 |
| 6 | L | |
| 7 | O | 28 |

Tabelas de hash fechada – tentativa quadrática

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | R | 12 |
| 5 | R | 20 |
| 6 | L | |
| 7 | O | 28 |

Inserir chave 44
 $44 \% 8 = 4$

| Índice | Situação | Chave |
|--------|----------|-------|
| 0 | L | |
| 1 | L | |
| 2 | O | 36 |
| 3 | L | |
| 4 | O | 44 |
| 5 | R | 20 |
| 6 | L | |
| 7 | O | 28 |

Tabelas de hash fechada: Exercício

Com base no que foi apresentado defina a(s) estrutura(s) de dados necessária(s) para a implementação de uma tabela hashing fechada com tentativa quadrática.

```
#define tam 8
```

```
typedef struct
```

```
{
```

```
    int chave;
```

```
    char livre; /* L = livre, O = ocupado, R = removido*/
```

```
}Hash;
```

```
typedef Hash Tabela[tam];
```

Tabelas de hash fechada: Exercício

Agora, implemente a função de inserção de uma chave na tabela hashing fechada em questão.

