

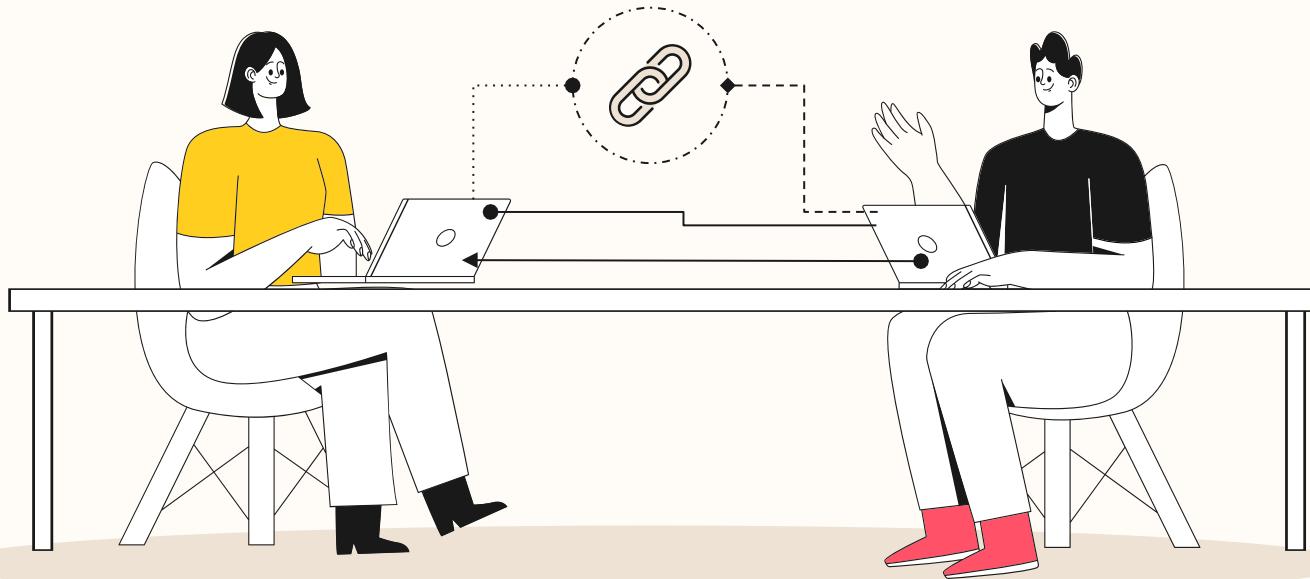
Algorit os e Estrutura de Dados

**Unidade 3 - Definição e Usos de Tabela de
Espalhamento**

Prof. Me. Anderson E. Macedo Gonçalves



Aula 2 - Operações em tabelas de espalhamento



Objetivo da aula

Operações em tabela

1.

de espalhamento

Apresentar as operações básicas da *hash table*.

2.

Função de Espalhamento

Vamos abordar a função de espalhamento.

3.

Redimensionamento dinâmico

Ajuste do tamanho da tabela de espalhamento.

4.

Palavras-chave

Operações, função de espalhamento, dinâmico.

Operações em tabelas *hash*

O método de "Transformação de Chave", ou funções de hashing, é completamente diferente, pois os registro de uma tabela são diretamente endereçados a partir de uma transformação aritmética sobre a chave de pesquisa.

De acordo com a tradução, a palavra *hash* em português quer dizer "fazer picadinho". O significado do termo realmente é apropriado para o método (ZIVIANI, 2011).

Operações em tabelas *hash*

As duas etapas principais de um método de pesquisa com o uso de transformação de chave são:

- 1. Cálculo da função de transformação (função hashing):** essa função é responsável por converter a chave original, que pode ser qualquer dado, em um valor numérico chamado de "hash" ou "código hash". A função de transformação é projetada de maneira que, idealmente, cada chave seja mapeada para um valor de hash exclusivo na tabela de espalhamento. Isso é fundamental para distribuir as chaves de forma uniforme na tabela, garantindo que a recuperação de dados seja eficiente.

Operações em tabelas *hash*

As duas etapas principais de um método de pesquisa com o uso de transformação de chave são:

2. **Resolução de colisões:** a segunda etapa lida com o problema das colisões, que ocorre quando duas ou mais chaves são transformadas em um mesmo endereço da tabela, ou seja, quando dois ou mais valores de *hash* são idênticos. Para resolver colisões, é necessário existir um método eficaz que permita armazenar e recuperar múltiplas chaves associadas ao mesmo endereço.

Operações em tabelas *hash*

Considerando chaves de valores inteiro de 1 a n, pode-se armazenar o registro com chave de i na posição i da tabela, e poderia ser acessado qualquer registro a partir do valor da chave. Porém supondo que uma tabela fosse capaz de armazenar $M = 97$ chaves, considerando ainda que cada chave pode ser um número decimal de quatro dígitos. Então agora, existem $N = 10.000$ chaves possíveis e a função de espalhamento não pode ser um para um: mesmo os elementos que serão armazenados forem bem menores que o total, ou seja, 97. As colisões irão acontecer e elas tem que ser resolvidas de alguma maneira (ZIVIANI, 2011)

Operações em tabelas *hash*

"Paradoxo do aniversário"

O "paradoxo do aniversário", nos diz que em um grupo de 23 pessoas existe a probabilidade de mais que 50% de duas ou mais pessoas fazerem aniversário no mesmo dia, portanto se for utilizada uma função de espalhamento que enderece 23 chaves aleatórias em uma tabela de tamanho 365, a probabilidade de que haja colisões é maior que 50%.

Operações em tabelas *hash*

"Paradoxo do aniversário"

A probabilidade p de se inserirem N itens consecutivos sem colisão em uma tabela de tamanho M seria a fórmula como pode ser observada a seguir:

$$p = \frac{M-1}{M} \times \frac{M-2}{M} \times \dots \times \frac{M-N+1}{M} = \prod_{i=1}^N \frac{M-i+1}{M} = \frac{M!}{(M-N)!M^N}.$$

Fonte: adaptada de Ziviani (2011).

Função de espalhamento

Algoritmo que aceita uma chave como entrada e produz um valor numérico chamado de "código hash", **objetivo:**

- Distribuir as chaves de forma uniforme na tabela de espalhamento.
- Garantir que as operações de busca, inserção e exclusão sejam eficientes, uma vez que as chaves estejam espalhadas por toda a tabela, em vez de ficarem agrupadas em um local específico.

Função de espalhamento - exemplo

Suponha que temos uma tabela de espalhamento com 10 posições e desejamos armazenar palavras. A função de espalhamento poderia ser a seguinte:

- Converta a primeira letra da palavra em um valor numérico (por exemplo, "a" seria 1, "b" seria 2, "c" seria 3 e assim por diante).
- Some os valores numéricos das letras da palavra.

Função de espalhamento - exemplo

Aplicando essa função a algumas palavras, temos:

- "apple" => a(1) + p(16) + p(16) + l(12) + e(5) = 50.
- "banana" => b(2) + a(1) + n(14) + a(1) + n(14) + a(1) = 33.
- "cherry" => c(3) + h(8) + e(5) + r(18) + r(18) + y(25) = 77.

Agora, podemos usar esses códigos *hash* para mapear as palavras nas posições da tabela de espalhamento. O processo de armazenamento seria o seguinte:

- "apple" (código hash 50) é armazenado na posição 50 da tabela.
- "banana" (código hash 33) é armazenado na posição 33 da tabela.
- "cherry" (código hash 77) é armazenado na posição 77 da tabela.

Redimensionamento dinâmico

Conhecido como *rehashing* se refere à capacidade de ajustar o tamanho da tabela *hash* à medida com que ela cresce, mantendo o fator de carga razoável.

- O fator de carga é a relação entre o número de elementos e o tamanho da tabela.

Redimensionamento dinâmico - exemplo

Suponha que você tenha uma tabela de espalhamento inicial com 10 posições e insira 7 elementos. O fator de carga atual é $7/10 = 0,7$, o que excede o limite predeterminado de 0,7. É hora de redimensionar a tabela:

Cria-se uma nova tabela com 20 posições para efetuar a reorganização dos elementos. Isso acontece de que forma?

Redimensionamento dinâmico - exemplo

Para redimensionar os elementos e assim aumentar o **fator de carga**, é necessário:

- Aplicar a função *hash* para cada elemento.
- Calcular seu novo código *hash* para a tabela de 20 posições.
- Distribuir os elementos nas novas posições.

Redimensionamento dinâmico -

| | |
|----|----|
| 1 | |
| 2 | |
| 3 | 45 |
| 4 | |
| 5 | 32 |
| 6 | |
| 7 | 89 |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | 21 |
| 14 | 65 |
| 15 | |
| 16 | |
| 17 | 67 |
| 18 | |
| 19 | 99 |
| 20 | |

Após o redimensionamento, a nova tabela tem um fator de carga de

$$7/20 = 0,35$$

o que está dentro do limite aceitável.

Realidade Profissional



Proposta de resolução da atividade

Para resolver siga os passos:

- Identificar a reserva a ser excluída.
- Pesquisar na tabela de espalhamento.
- Lidar com colisões (lista ligada com encadeamento separado).
- Exclusão da reserva.

Revisando

Operações em tabela

1.

de espalhamento

Apresentar as operações básicas da *hash table*.

2.

Função de Espalhamento

Vamos abordar a função de espalhamento.

3.

Redimensionamento dinâmico

Ajuste do tamanho da tabela de espalhamento.

4.

Palavras-chave

Operações, função de espalhamento, dinâmico.