

Estrutura de Dados II

Prof. Me. Pietro M. de Oliveira

Tabelas de Dispersão

Hash Tables

Todos os métodos recém apresentados tem como base a comparação entre **chave** e os **elementos de um arranjo**

Hashtables

- Os elementos são indexados diretamente a partir de seu valor (geralmente dispensa comparações)
- Acesso direto (inserção e pesquisa com 1 iteração)
- Função *hash* – cálculo matemático
 - “Transforma” chave em índice
- Mais de um registro com mesmo índice (colisão)

Função de Dispersão (função *hash*)

- Processa uma chave x qualquer
- Retorna o índice no qual a chave deve estar armazenada

Considere um vetor de 10 posições (de 0 a 9)

- Exemplo de função hash:

$$h(x) = x^2 \bmod 10$$

- Para armazenar/pesquisar dados, basta substituir x pelo valor da chave em questão

Para armazenar/pesquisar dados, basta substituir x pelo valor da chave em questão

$$h(x) = x^2 \bmod 10$$

- Imagine que temos um vetor no qual precisamos armazenar o dado inteiro 72
 - O índice da posição na qual 72 será armazenado é:

Para armazenar/pesquisar dados, basta substituir x pelo valor da chave em questão

$$h(x) = x^2 \bmod 10$$

- Imagine que temos um vetor no qual precisamos armazenar o dado inteiro 72
- O índice da posição na qual 72 será armazenado é:

$$h(72) = 72^2 \bmod 10$$

Para armazenar/pesquisar dados, basta substituir x pelo valor da chave em questão

$$h(x) = x^2 \bmod 10$$

- Imagine que temos um vetor no qual precisamos armazenar o dado inteiro 72
- O índice da posição na qual 72 será armazenado é:

$$h(72) = 72^2 \bmod 10$$

$$h(72) = 5184 \bmod 10$$

Para armazenar/pesquisar dados, basta substituir x pelo valor da chave em questão

$$h(x) = x^2 \bmod 10$$

- Imagine que temos um vetor no qual precisamos armazenar o dado inteiro 72
- O índice da posição na qual 72 será armazenado é:

$$h(72) = 72^2 \bmod 10$$

$$h(72) = 5184 \bmod 10$$

$$\mathbf{h(72) = 4}$$

Para armazenar/pesquisar dados, basta substituir x pelo valor da chave em questão

$$h(x) = x^2 \bmod 10$$

- Imagine que temos um vetor no qual precisamos armazenar o dado inteiro 72
- O índice da posição na qual 72 será armazenado é:

$$h(72) = 72^2 \bmod 10$$

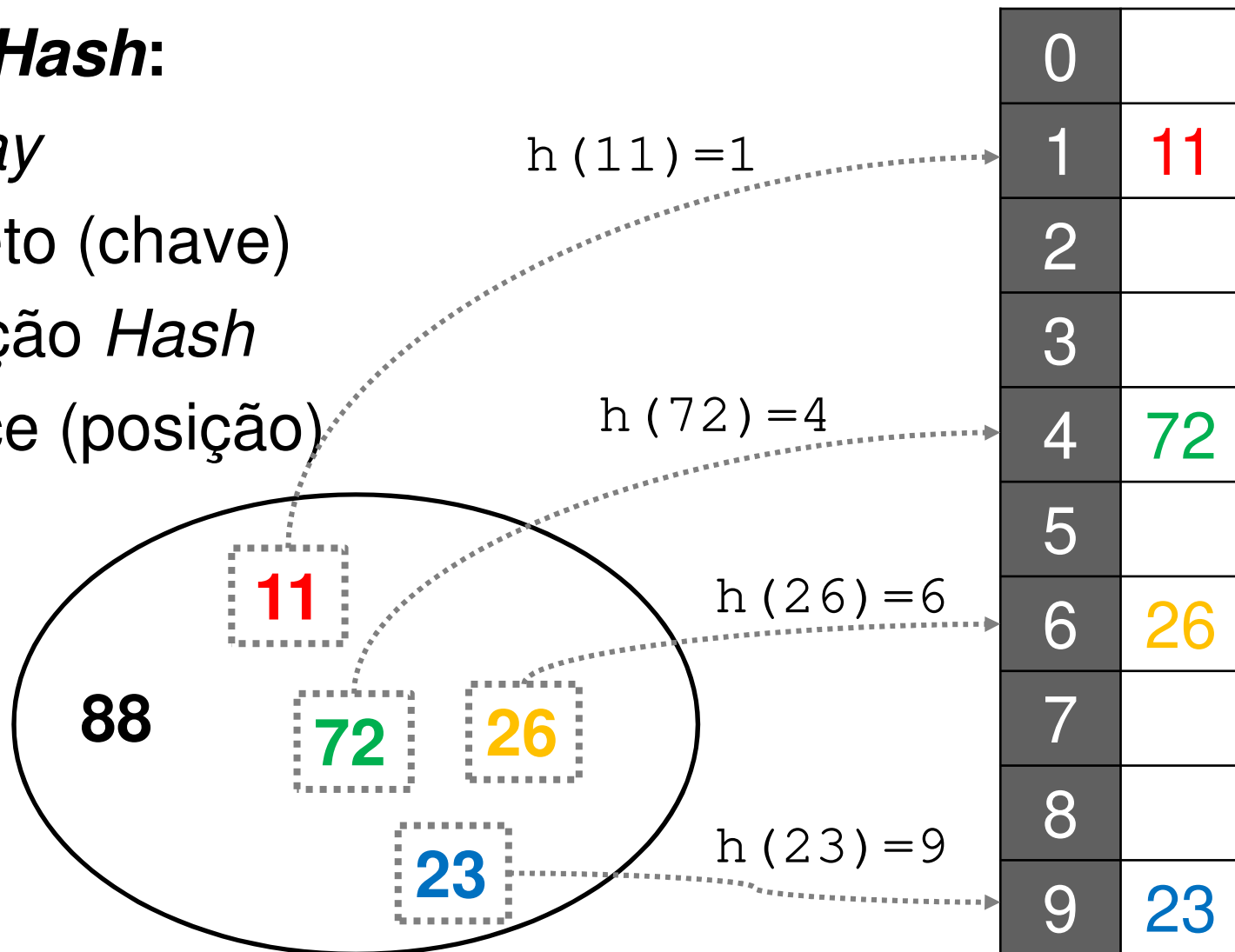
$$h(72) = 5184 \bmod 10$$

$$\mathbf{h(72) = 4}$$

3	...
4	72
5	...

Tabela Hash:

- I. *Array*
- II. Objeto (chave)
- III. Função *Hash*
- IV. Índice (posição)



Busca em Tabela Hash:

- Similar à inserção
- Processar a chave com **$h()$**

Exemplo:

- Buscar pelo elemento **23**
 - **$h(23) = 529 \bmod 10 = 9$**
- Buscar pelo elemento **35**
 - **$h(35) = 1225 \bmod 10 = 5$**

0	
1	11
2	
3	
4	72
5	
6	26
7	
8	
9	23

Existem situações nas quais dois ou mais elementos são mapeadas para uma mesma posição

- **Colisão!**

$$h(88) = 4$$

Exemplo de Colisão:

- Tentar inserir o **88**
 - $h(88) = 7744 \bmod 10 = 4$

0	
1	11
2	
3	
4	72
5	
6	26
7	
8	
9	23



Solução 1: Procurar pela próxima posição livre

- Inserção com colisão:
 - Armazenar na “próxima vaga”

Problema: e se algum elemento tiver sido removido?

- Cada posição a tabela possui uma *flag*
- **Vetor de structs:**
 - Armazenar dados
 - Sinalizar remoção

Exemplo - Solução 1

- Inserir o **88**

0		
1	11	
2		
3		
4	72	
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Inserir o **88**

$$\begin{aligned}h(88) &= \\88^2 \bmod 10 &= \\7744 \bmod 10 &= \\4\end{aligned}$$

0		
1	11	
2		
3		
4	72	
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Inserir o **88**

$$h(88) =$$

$$88^2 \bmod 10 =$$

$$7744 \bmod 10 =$$

4

0		
1	11	
2		
3		
4	72	
5		
6	26	
7		
8		
9	23	



Exemplo - Solução 1

- Inserir o **88**

$$h(88) =$$

$$88^2 \bmod 10 =$$

$$7744 \bmod 10 =$$

4

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	



Exemplo - Solução 1

- Buscar o 88

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Buscar o 88

$h(88)$

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	


Exemplo - Solução 1

- Buscar o 88

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Buscar o 88



0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Buscar o 88

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Inserir o 25

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	

Exemplo - Solução 1

- Inserir o 25

$$h(25) =$$

$$25^2 \bmod 10 =$$

$$625 \bmod 10 =$$

5

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7		
8		
9	23	

×

Exemplo - Solução 1

- Inserir o 25

$$\begin{aligned}h(25) &= \\25^2 \bmod 10 &= \\625 \bmod 10 &= \\5\end{aligned}$$

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7	25	
8		
9	23	



Exemplo - Solução 1

- Remover **72**

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Remover **72**

$$\begin{aligned}h(72) &= \\72^2 \bmod 10 &= \\5184 \bmod 10 &= \\4\end{aligned}$$

0		
1	11	
2		
3		
4	72	
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Remover **72**

$$\begin{aligned}h(72) &= \\72^2 \bmod 10 &= \\5184 \bmod 10 &= \\4\end{aligned}$$

0		
1	11	
2		
3		
4	72	R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Remover **72**

$$\begin{aligned}h(72) &= \\72^2 \bmod 10 &= \\5184 \bmod 10 &= \\4\end{aligned}$$

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o **12**

$h(12)$

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	


Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12



0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o **12**

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o 12

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

Exemplo - Solução 1

- Buscar o **12**

0		
1	11	
2		
3		
4		R
5	88	
6	26	
7	25	
8		
9	23	

×

Solução 2: Criar listas encadeadas durante colisões











- Inserção com colisão:
 - “Pendurar” o elemento na lista

Alocação dinâmica:

- Cada posição da tabela possui uma lista
- **Vetor de structs:**
 - Armazenar dados
 - Ponteiro para o próximo elemento colidido

Exemplo - Solução 2











- Inserir o **88**

0		
1	11	
2		
3		
4	72	
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Inserir o **88**

$$\begin{aligned}h(88) &= \\88^2 \bmod 10 &= \\7744 \bmod 10 &= \\4\end{aligned}$$

0		
1	11	
2		
3		
4	72	
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Inserir o **88**

$$h(88) =$$

$$88^2 \bmod 10 =$$

$$7744 \bmod 10 =$$

4



0		
1	11	
2		
3		
4	72	
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Inserir o **88**

$$\begin{aligned}
 h(88) &= \\
 88^2 \bmod 10 &= \\
 7744 \bmod 10 &= \\
 \mathbf{4}
 \end{aligned}$$

0		
1	11	
2		
3		
4	72	→
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Inserir o **88**

$$h(88) =$$

$$88^2 \bmod 10 =$$

$$7744 \bmod 10 =$$

4

0		
1	11	
2		
3		
4	72	88
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Buscar o 88

0		
1	11	
2		
3		
4	72	88
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Buscar o 88

$h(88)$

0		
1	11	
2		
3		
4	72	88
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Buscar o **88**

0		
1	11	
2		
3		
4	72	88
5		
6	26	
7		
8		
9	23	

Exemplo - Solução 2

- Buscar o 88

0		
1	11	
2		
3		
4	72	88
5		
6	26	
7		
8		
9	23	

- Buscas em vetores estáticos
 - Sequencial
 - Sequencial indexada
 - Binária
 - Por interpolação
- Tabelas *Hash*
 - Funções *hash*
 - Colisões

Estrutura de Dados II

Prof. Me. Pietro M. de Oliveira