

ESTRUTURA DE DADOS E ALGORITMOS

Hashing (Tabela de Dispersão)

Cristina Boeres

Motivação



- Os métodos de pesquisa vistos até agora buscam informações armazenadas com base na comparação de suas chaves
- Para obtermos algoritmos eficientes, armazenamos os elementos ordenados e tiramos proveito dessa ordenação
- hashing (tabela de dispersão) ou método de cálculo de endereço
 - No caso médio é possível encontrar a chave em tempo constante

Conceitos Básicos

- Índices em vetores ou listas sequenciais são utilizados para acessar informações
- No entanto, se quisermos acessar uma informação de um determinado conteúdo (e não posição)?
 - temos que procurá-lo

Família	1	2	3	4	5	6
	José Maria	Leila	Artur	Jolinda	Gisela	Alciene

`Família[1] = "José Maria"`

`Família[3] = "Artur"`

`Família[2] = "Leila"`

Em qual posição está "Alciene" ?

Conceitos Básicos

- Ideal

- Parte da informação poderia ser utilizada na recuperação ou busca(consulta)

Definição de Hash (1/3)

- **Hash** é uma generalização da noção mais simples de um arranjo comum, sendo uma estrutura de dados do tipo dicionário
- Dicionários são estruturas especializadas em prover as operações de inserir, pesquisar e remover.
- A idéia central do **Hash** é utilizar uma função, aplicada sobre parte da informação (**chave**), para retornar o índice onde a informação deve ou deveria estar armazenada.

Definição de Hash (2/3)

- Esta função que mapeia a chave para um índice de um arranjo é chamada de **Função de Hashing**
- A estrutura de dados Hash é comumente chamada de Tabela Hash.

Definição de Hash (3/3)

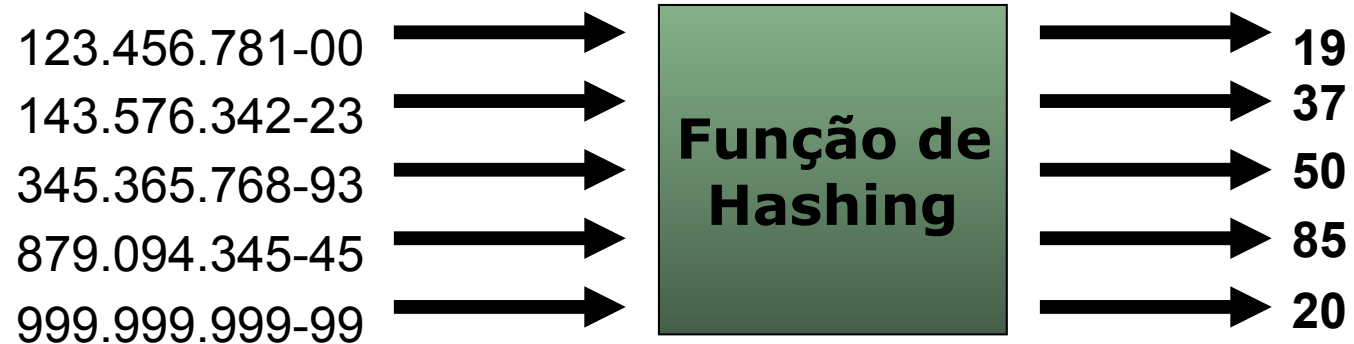


Tabela Hash

19	123.456.781-00; Fausto Silva; Av. Canal. Nº 45.
20	
...	
37	143.576.342-23; Carla Perez; Rua Celso Oliva. Nº 27.
...	
50	345.365.768-93; Gugu Liberato; Av. Atlântica. S/N.
...	
85	879.094.345-45 ; Hebe Camargo; Rua B. Nº 100.
...	

Tabela Hash



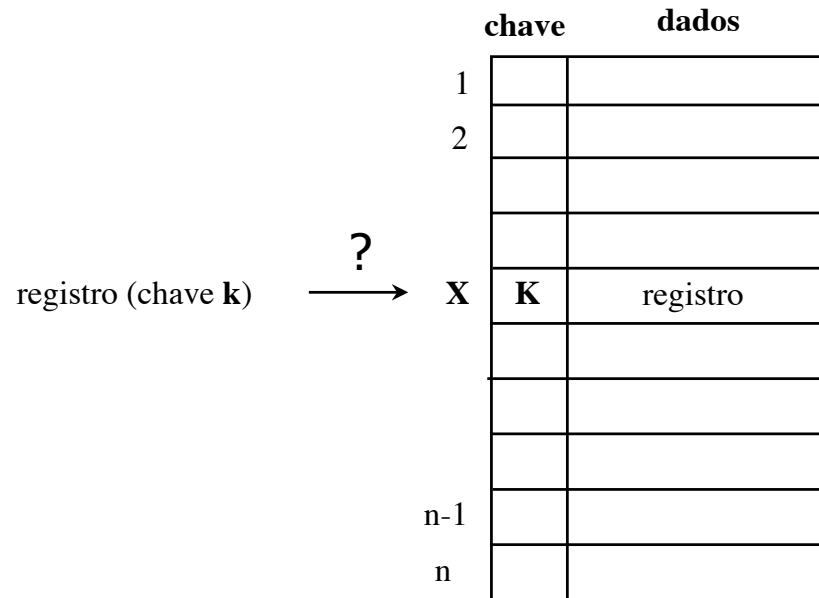
□ Tabela Hash

- é uma estrutura de dados especial
- armazena as informações desejadas associando chaves

□ Objetivo

- a partir de uma chave, fazer uma busca rápida e obter o valor desejado.

Ilustração de uma Tabela Hash



Como o registro (com chave **K**) foi armazenado na posição **X** na Tabela Hash ao lado?

Resp: Através de uma **Função de Hashing**.

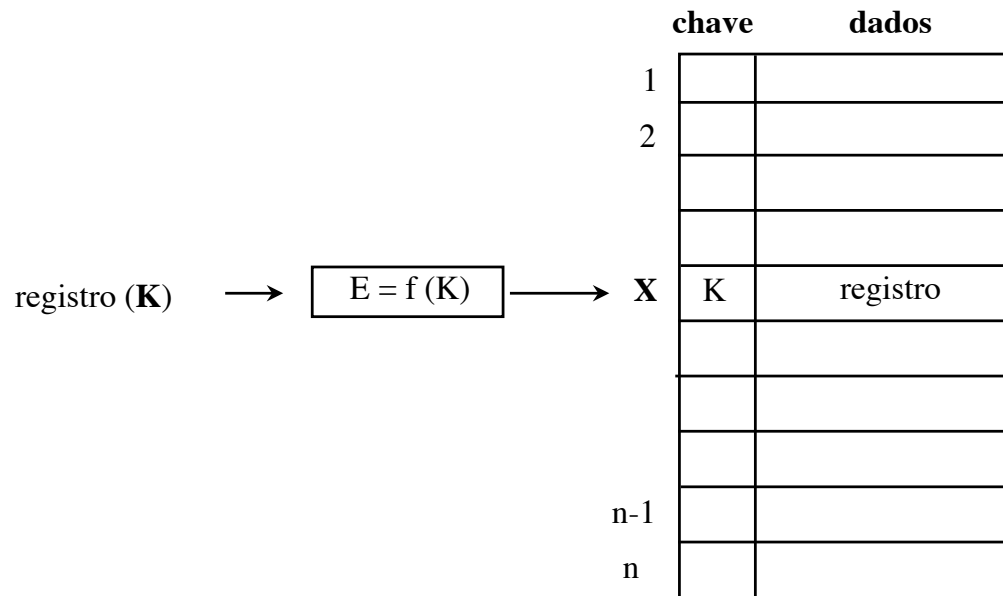
Como representar Tabelas Hash?

- Vetor ou lista sequencial
 - cada posição do vetor guarda uma informação.
 - Se a função de hashing aplicada a um conjunto de elementos determinar as informações i_1, i_2, \dots, i_n
 - então o vetor $V[1 \dots n]$ é usado para representar a tabela hash
- mas somente vetor?

Função de Hashing (ou dispersão)

- A Função de Hashing é a responsável por gerar um índice a partir de uma determinada chave
 - O ideal é que a função forneça índices únicos para o conjunto das chaves de entrada possíveis
 - sem colisões
 - fácil de computar
 - uniforme (todos os locais da tabela sejam igualmente utilizados)
 - extremamente importante, pois ela é responsável por distribuir as informações pela Tabela Hash

Ilustração da Função de Hashing



- Os valores da chave podem ser **numéricos**, **alfabéticos** ou **alfa-numéricos**.

- Executam a transformação do valor de uma chave em um endereço, pela aplicação de operações aritméticas e/ou lógicas

$$f: C \rightarrow E$$

f: função de cálculo de endereço

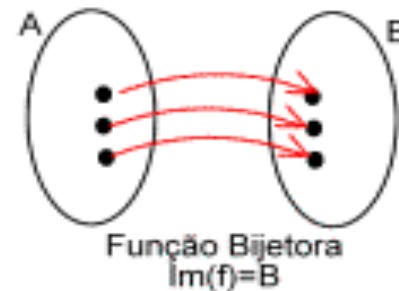
C: espaço de valores da chave (domínio de f)

E: espaço de endereçamento (contradomínio de f)

Hashing Perfeito

□ Característica:

- Para quaisquer chaves x e y diferentes e pertencentes a A , a função utilizada fornece saídas diferentes;



Exemplo de Hashing Perfeito

- Armazenamento de alunos de uma determinada turma de um curso específico
- Cada aluno é identificado unicamente pela sua matrícula.

```
struct aluno {  
    int mat;           // 4 bytes      = 4 bytes  
    char nome[81];     // 1 byte * 81 = 81 bytes  
    char email[41];    // 1 byte * 41 = 41 bytes  
};                    // Total = 126 bytes
```

```
typedef struct aluno Aluno;
```

Exemplo de Hashing Perfeito

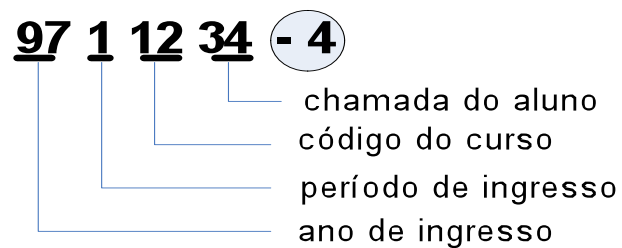
- O número de dígitos efetivos na matrícula são 7
- Para permitir um acesso a qualquer aluno em ordem constante, podemos usar o número de matrícula do aluno como índice de um vetor
- Um problema é que pagamos um preço alto para ter esse acesso rápido. Porque?

Exemplo de Hashing Perfeito

- O número de dígitos efetivos na matrícula são 7
- Para permitir um acesso a qualquer aluno em ordem constante, podemos usar o número de matrícula do aluno como índice de um vetor
- Um problema é que pagamos um preço alto para ter esse acesso rápido. Porque?
 - Visto que a matrícula é composta de 7 dígitos, então podemos esperar uma matrícula variando de 0000000 a 9999999. Portanto, precisaríamos dimensionar um vetor com DEZ Milhões de elementos

Exemplo de Hashing Perfeito

- Como economizar em espaço, mas ainda usando *hashing perfeito*
 - Identificando as partes **significativas** da chave:



Exemplo de Hashing Perfeito

- mostra a dimensão que a Tabela Hash deverá ter

- Por exemplo, dimensionando com apenas 100 elementos, ou seja,

`aluno* tabAlunos[100];`

- Função que aplicada sobre matrículas de alunos retorna os índices únicos da tabela

Exemplo de Hashing Perfeito (6/6)

- Supondo que a turma seja do 2º semestre de 2005 (código 052) e do curso de Sistemas de Informação (código 35).

Qual seria a função de hashing **perfeito** !?

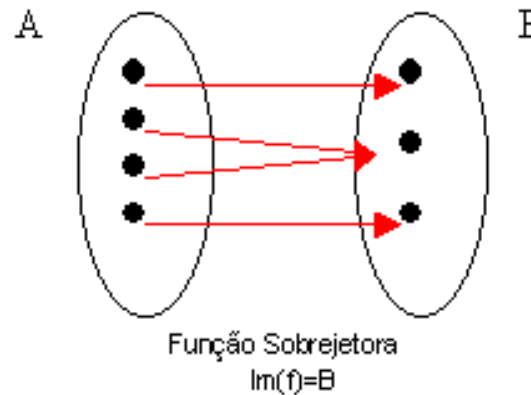
```
int funcao_hash(int matricula) {  
    int valor = matricula - 0523500;  
    if((valor >= 0) & (valor <=99)) then  
        return valor;  
    else  
        return -1;  
}
```

Acesso: dada **mat** → tabAlunos[funcao_hash(mat)]

Hashing Imperfeito

□ Características:

- Existe chaves x e y diferentes e pertencentes a A , onde a função Hash utilizada fornece saídas iguais;



Exemplo de Hashing Imperfeito

- Suponha que queiramos armazenar as seguintes chaves: C, H, A, V, E e S em um vetor de $P = 7$ posições (0..6) conforme a seguinte

- função $f(k) = k(\text{código ASCII}) \% P$.

- Exemplo

símbolo	ASCII
C	67
H	72
A	65
V	86
E	69
S	83

Exemplo de Hashing Imperfeito

- Suponha que queiramos armazenar as seguintes chaves: C, H, A, V, E e S em um vetor de $P = 7$ posições (0..6) conforme a seguinte

- função $f(k) = k(\text{código ASCII}) \% P$.

- Exemplo

símbolo	ASCII	$f(k)$
C	67	4
H	72	2
A	65	2
V	86	2
E	69	6
S	83	6

Colisões

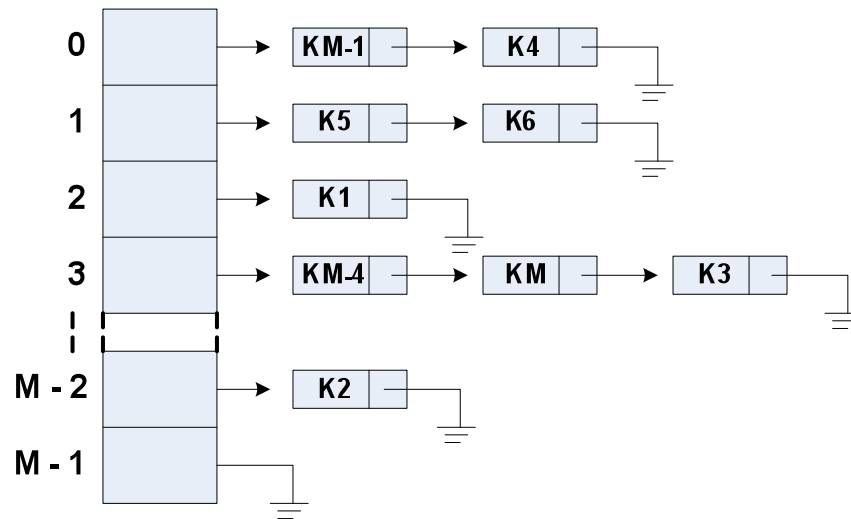
- Quando duas ou mais chaves geram o mesmo endereço da Tabela Hash
- É comum ocorrer colisões.
- Principais causas:
 - em geral o número N de chaves possíveis é muito maior que o número m disponíveis na tabela
 - não se pode garantir que as funções de hashing possuam um bom potencial de distribuição (espalhamento)

Colisões: Encadeamento

- Cada entrada na tabela aponta para uma lista encadeada
 - colisões geram uma nova entrada em uma lista
- A função utilizada deve ser uniforme para evitar uma grande lista encadeada em poucas posições da tabela
- Cada busca só será constante se o número de elementos em cada lista encadeada for pequeno

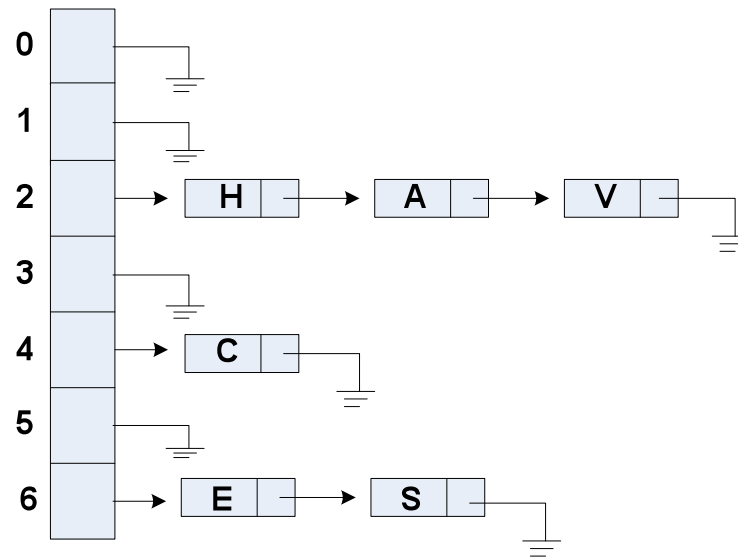
Colisões: Encadeamento

- A informação é armazenada em estruturas encadeadas



Colisões: Encadeamento

- $P = 7$ posições (0..6) e a **função** $f(k) = k(\text{código ASCII}) \% P$.



Colisões: Encadeamento

0 (A)

1 (B)

2 (C)

3 (D)

4 (E)

5 (F)

⋮

⋮

⋮

⋮

⋮

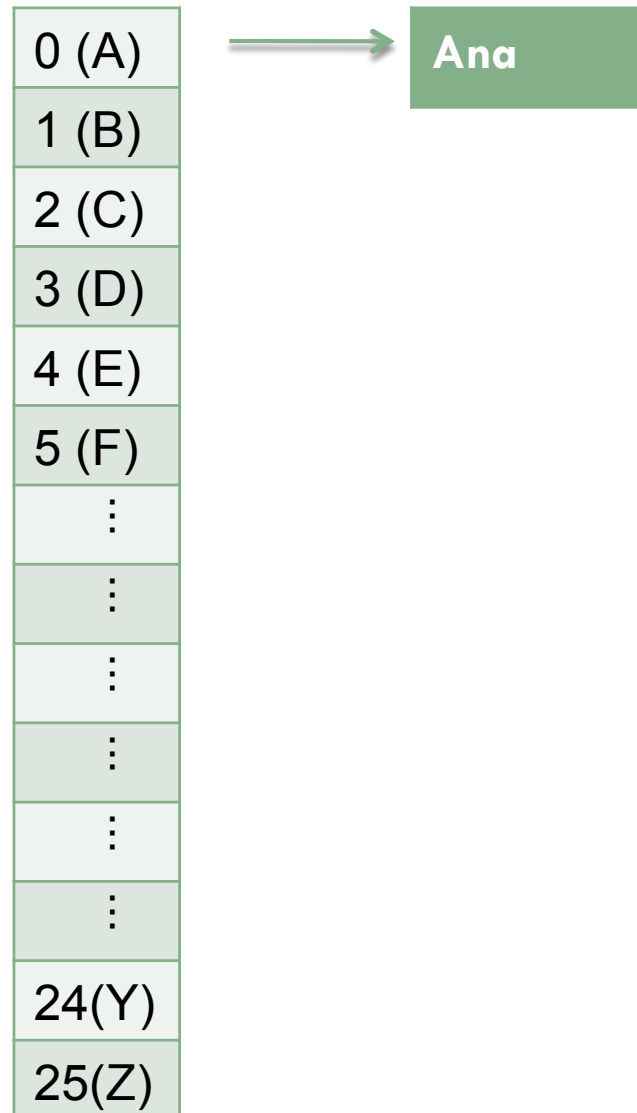
⋮

24(Y)

25(Z)

insere "Ana"

Colisões: Encadeamento



Colisões: Encadeamento

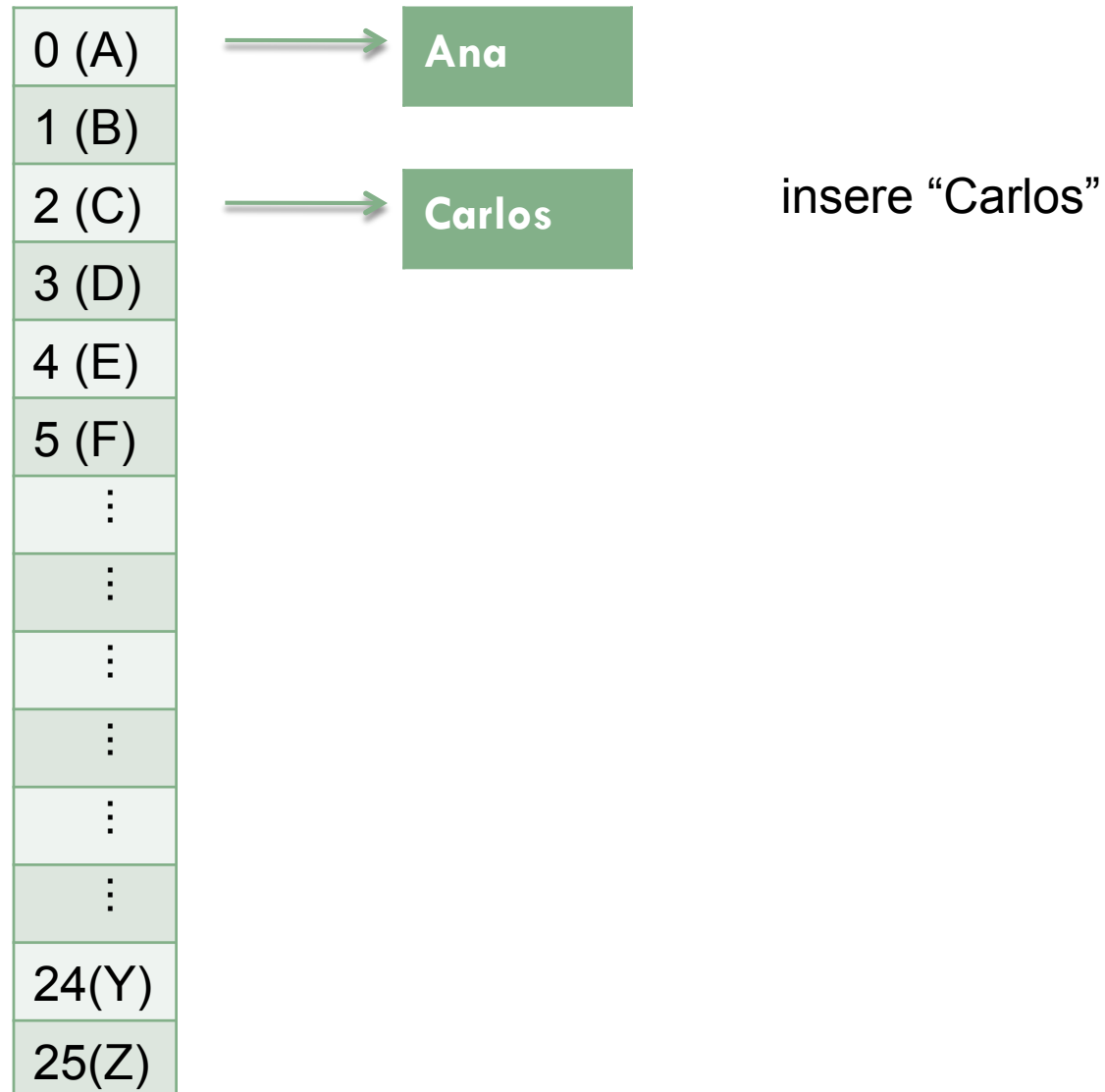
0 (A)
1 (B)
2 (C)
3 (D)
4 (E)
5 (F)
⋮
⋮
⋮
⋮
⋮
⋮
24(Y)
25(Z)



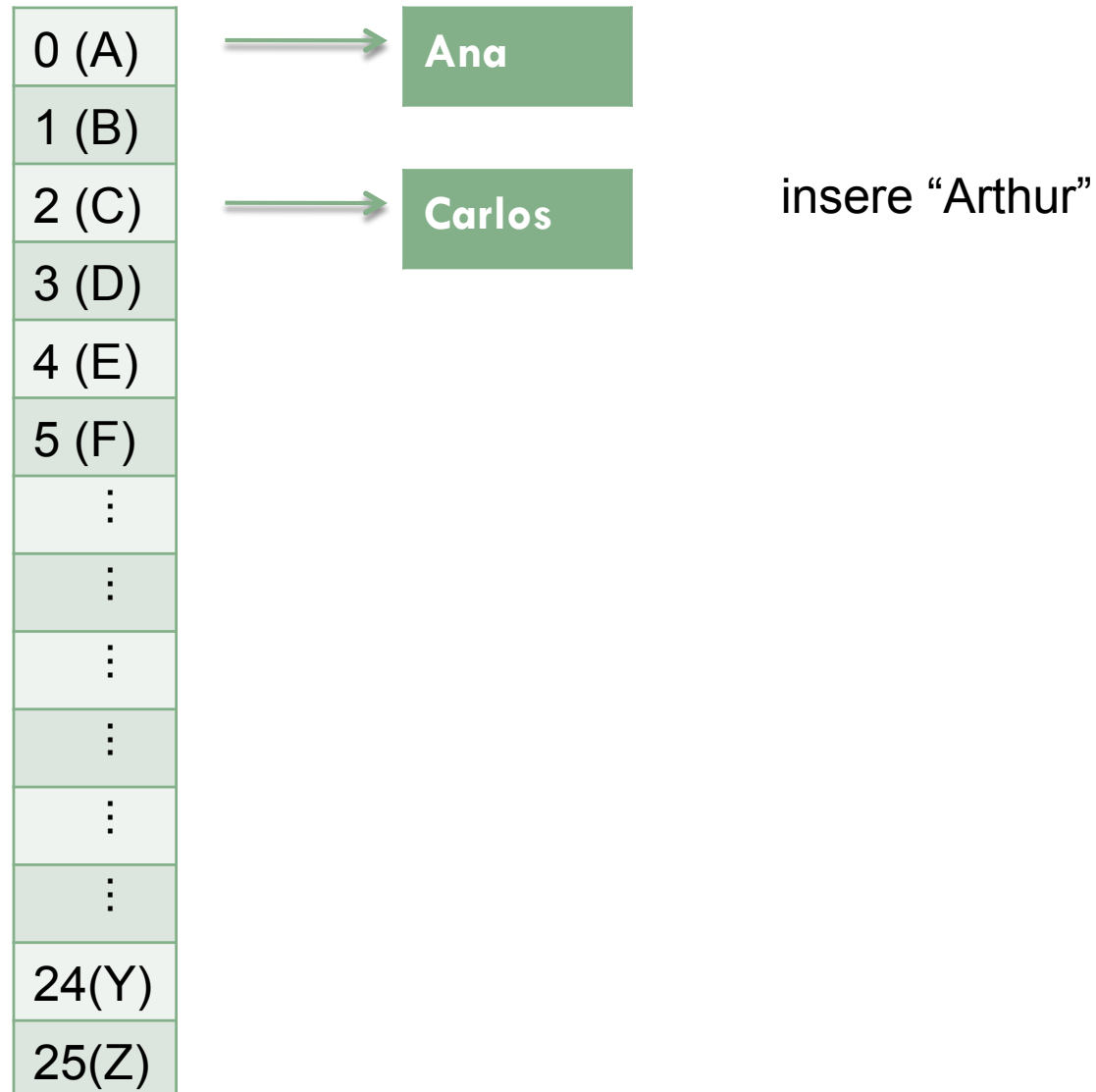
Ana

insere "Carlos"

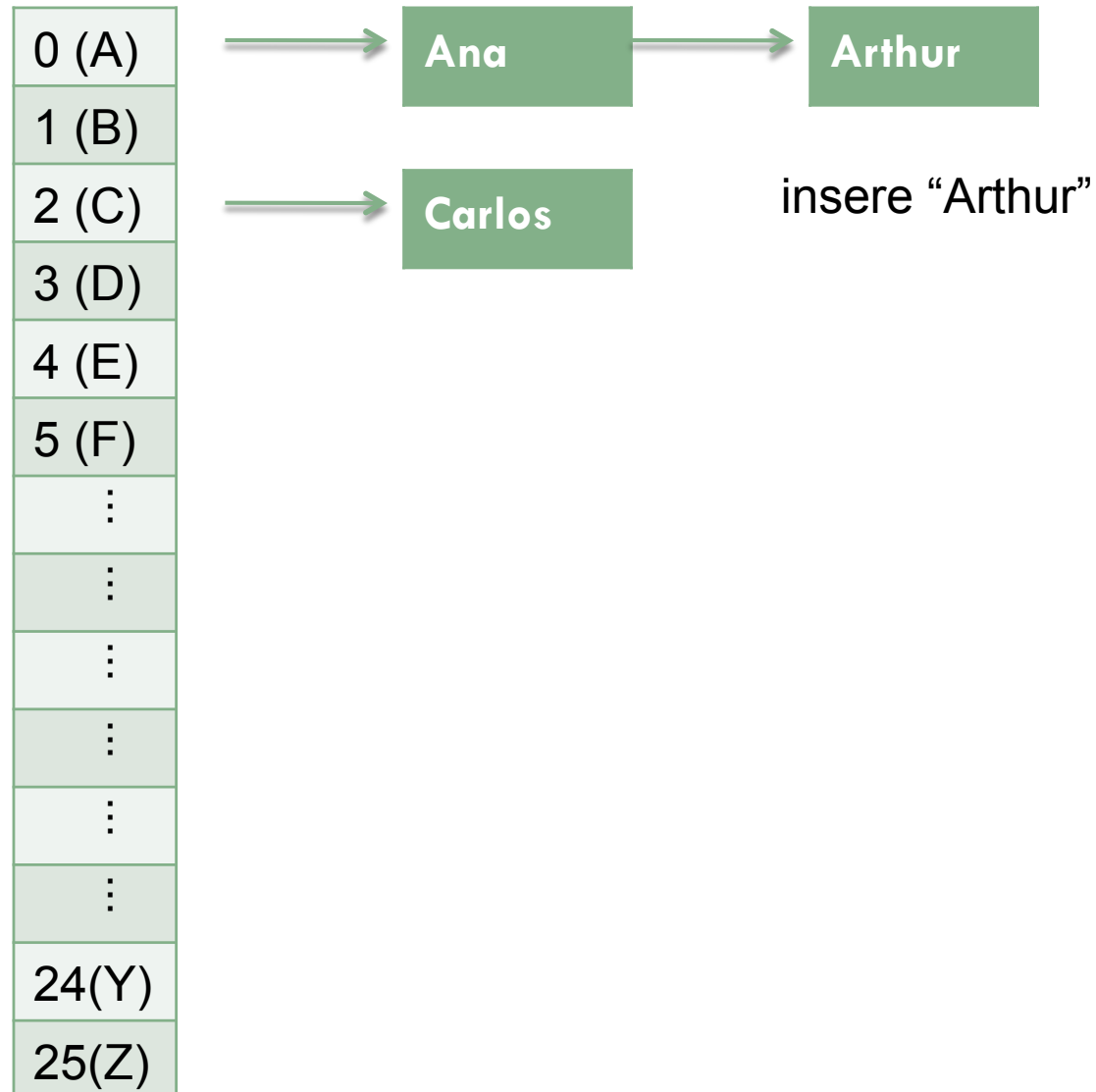
Colisões: Encadeamento



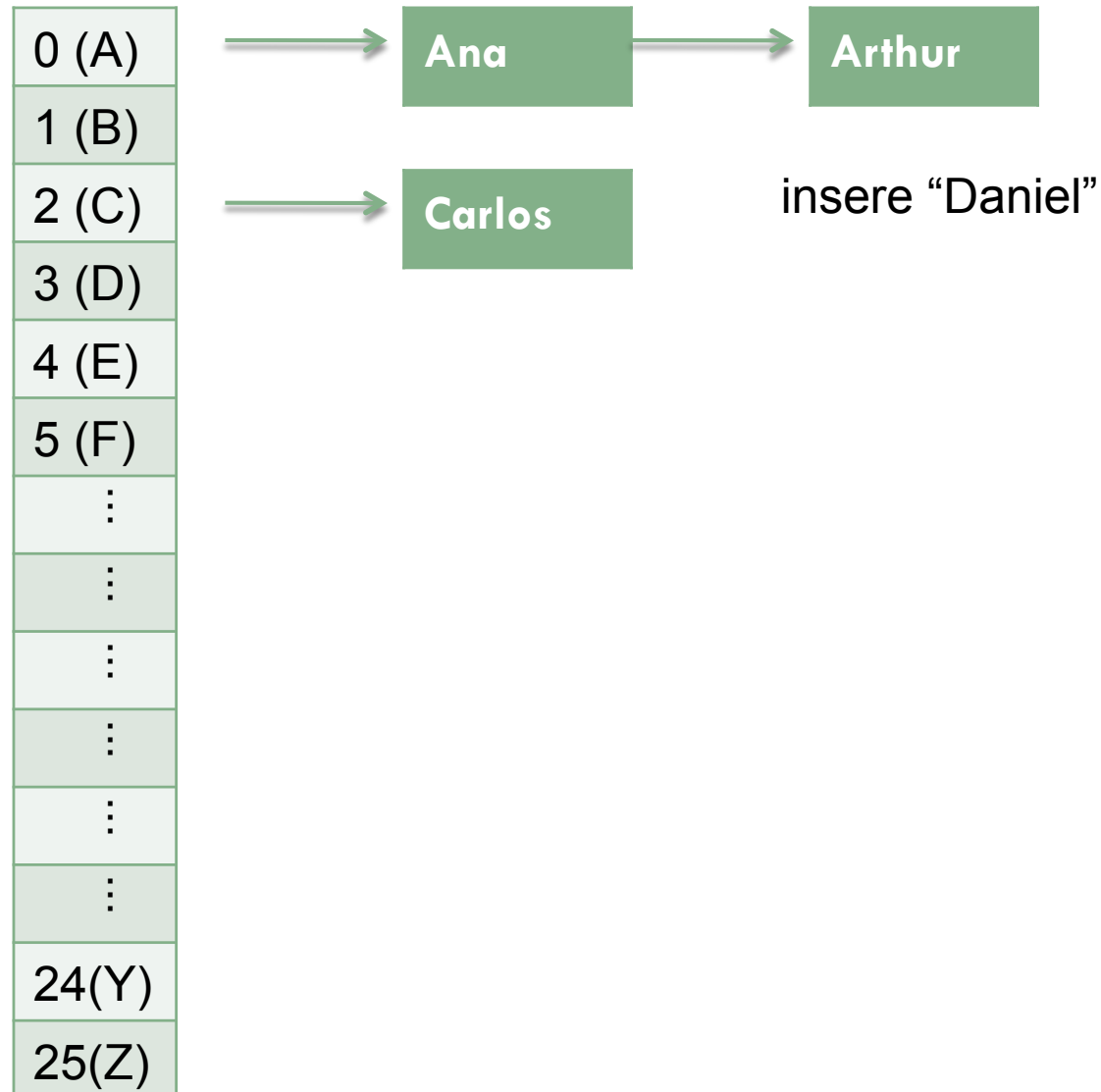
Colisões: Encadeamento



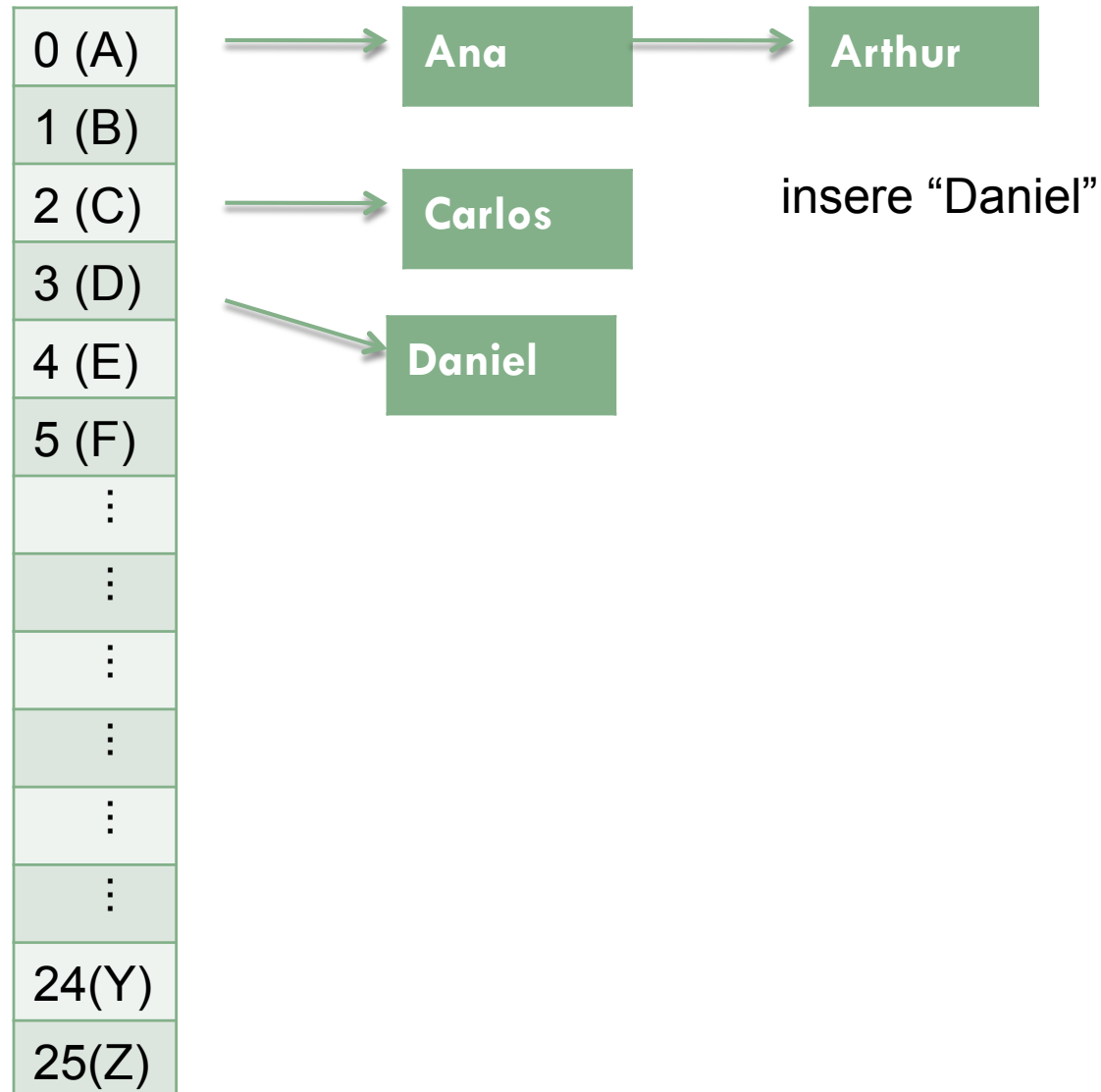
Colisões: Encadeamento



Colisões: Encadeamento



Colisões: Encadeamento



Colisões: endereçamento aberto

- sem listas encadeadas
- sem informação adicional
- quando houver colisões – através de um cálculo qual o próximo local a ser examinado
- sucesso: vai calculando até achar uma posição livre/ encontra a chave
- sem sucesso: a tabela está cheia ou não se encontra a chave

Colisões: endereçamento aberto

- supõe-se que exista uma função hash $h()$ que um endereço base $h(x,k)$ distinto para cada k de 0 a m
- inicialmente, veremos endereçamento aberto
 - tentativa/sondagem linear
 - tentativa/sondagem quadrática

Colisões: sondagem linear

- Ao verificar que uma posição $h(k)$ da tabela está ocupada, tenta adicionar o elemento na primeira posição livre seguinte:
 - $h(k) + 1, h(k) + 2, h(k) + 3, \dots$ até uma posição vazia, considerando a tabela circular
- função hash

$$h(x,k) = (h'(x) + k) \bmod m \quad 0 \leq k \leq m-1$$

Colisões: sondagem linear



0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
21	
22	

- $m = 23$

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

- Ao inserir

- $44 \rightarrow k = 0, 44 \bmod 23 = 21$

- $46 \rightarrow k = 0, 46 \bmod 23 = 0$

- $49 \rightarrow k = 0, 49 \bmod 23 = 3$

- $68 \rightarrow k = 0, 68 \bmod 23 = 22$

- $71 \rightarrow k = 0, 71 \bmod 23 = 3$

- $97 \rightarrow k = 0, 44 \bmod 23 = 4$

Colisões: sondagem linear

0	46
1	
2	71
3	49
4	
5	97
6	
7	
8	
9	
21	44
22	68

- $m = 23$

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

- Ao inserir

- $44 \rightarrow k = 0, 44 \bmod 23 = 21$

- $46 \rightarrow k = 0, 46 \bmod 23 = 0$

- $49 \rightarrow k = 0, 49 \bmod 23 = 3$

- $68 \rightarrow k = 0, 68 \bmod 23 = 22$

- $71 \rightarrow k = 0, 71 \bmod 23 = 3$

- $97 \rightarrow k = 0, 97 \bmod 23 = 4$

Colisões: sondagem linear

0	46
1	
2	71
3	49
4	
5	97
6	
7	
8	
9	
21	44
22	68

- $m = 23$

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

- inserir 26
 - $k = 0, 26 \bmod 23 = 3$
 - $k = 1, 26+1 \bmod 23 = 4$

Colisões: sondagem linear

0	46
1	
2	71
3	49
4	26
5	97
6	
7	
8	
9	
21	44
22	68

- $m = 23$

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

- inserir 26
 - $k = 0, 26 \bmod 23 = 3$
 - $k = 1, 26+1 \bmod 23 = 4$

Colisões: sondagem linear

0	46
1	
2	71
3	49
4	26
5	97
6	
7	
8	
9	
21	44
22	68

- $m = 23$

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

- inserir 72
 - $k = 0, 72 \bmod 23 = 3$
 - $k = 1, 72+1 \bmod 23 = 4$
 - $k = 2, 72+2 \bmod 23 = 5$
 - $k = 3, 72+3 \bmod 23 = 6$

Colisões: sondagem linear

0	46
1	
2	71
3	49
4	26
5	97
6	72
7	
8	
9	
21	44
22	68

- $m = 23$

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

- Ao inserir
- inserir 72
 - $k = 0, 72 \bmod 23 = 3$
 - $k = 1, 72+1 \bmod 23 = 4$
 - $k = 2, 72+2 \bmod 23 = 5$
 - $k = 3, 72+3 \bmod 23 = 6$

Colisões: sondagem linear



□ Consequências:

- Cria grandes blocos de dados numa mesma região da tabela
- Dificulta a remoção de dados
- Aumenta a complexidade para a busca nos casos de colisão
- limitado pelo tamanho da tabela

Colisões: sondagem quadrática

- Tentativa de se espalhar mais os elementos
- função

$$h(x,k) = (h'(x) + c_1k + c_2k^2) \bmod m$$

- onde c_1 e c_2 são constantes, $c_2 \neq 0$ e $k = 0, \dots, m-1$

Colisões: sondagem quadrática

- Tentativa de se espalhar mais os elementos
- função

$$h(x,k) = (h'(x) + c_1k + c_2k^2) \bmod m$$

- onde c_1 e c_2 são constantes, $c_2 \neq 0$ e $k = 0, \dots, m-1$
- exemplo:

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

$$h(x,k) = (h(x, k-1) + k) \bmod m \quad 0 < k < m-1$$

- tentar no exemplo anterior: as tentativas são as mesmas? O número de tentativas

Colisões: sondagem quadrática

- Como ficaria o algoritmo de
 - busca
 - inserção
 - remoção
- de uma chave v para endereçamento aberto com
 - sondagem linear
 - sondagem quadrática