Tabela Hash

Tratamento de Colisão

No nosso último encontramos vimos as definições das tabelas de dispersão

Que para armazenar n elementos, não necessitaríamos de indexação direta, que poderíamos armazena-los em uma estrutura de m espaços, tal que $n \leq m$

Compreendemos também, que correspondem a uma estrutura indexada no intervalo [0, m-1], tendo então m partições (espaços). Cada partição pode armazenar 1 elemento ou r elementos distintos.

E quem em tal estrutura, podemos transformar as n chaves em um valor entre [0, m-1], acessando-as para consulta e inserção em tempo O(1)

A transformação de cada chave x em um valor neste intervalo é realizada por meio de uma função de dispersão também conhecida como função hash h(x)

Ao calcularmos o endereço com uso da função, caso a posição/endereço h(x) esteja vazio/desocupado poderemos armazenar um novo elemento neste espaço ou capturar a informação contida nele

Mas vimos que há alguns detalhes no processo de inserção. Há a probabilidade de uma chave y diferente de x, “apontar” pro mesmo endereço. Ou seja h(x) = h(y)

E que para tratar tal situação utilizamos um procedimento para tratamento de colisões

Dessa forma, sugere-se, que para qualquer tipo de tabela hash, tenhamos o acompanhamento do seu fator de carga/taxa de ocupação que é dado por $\alpha = \frac{n}{m'}$, sendo n o número de chaves indexadas na estrutura.

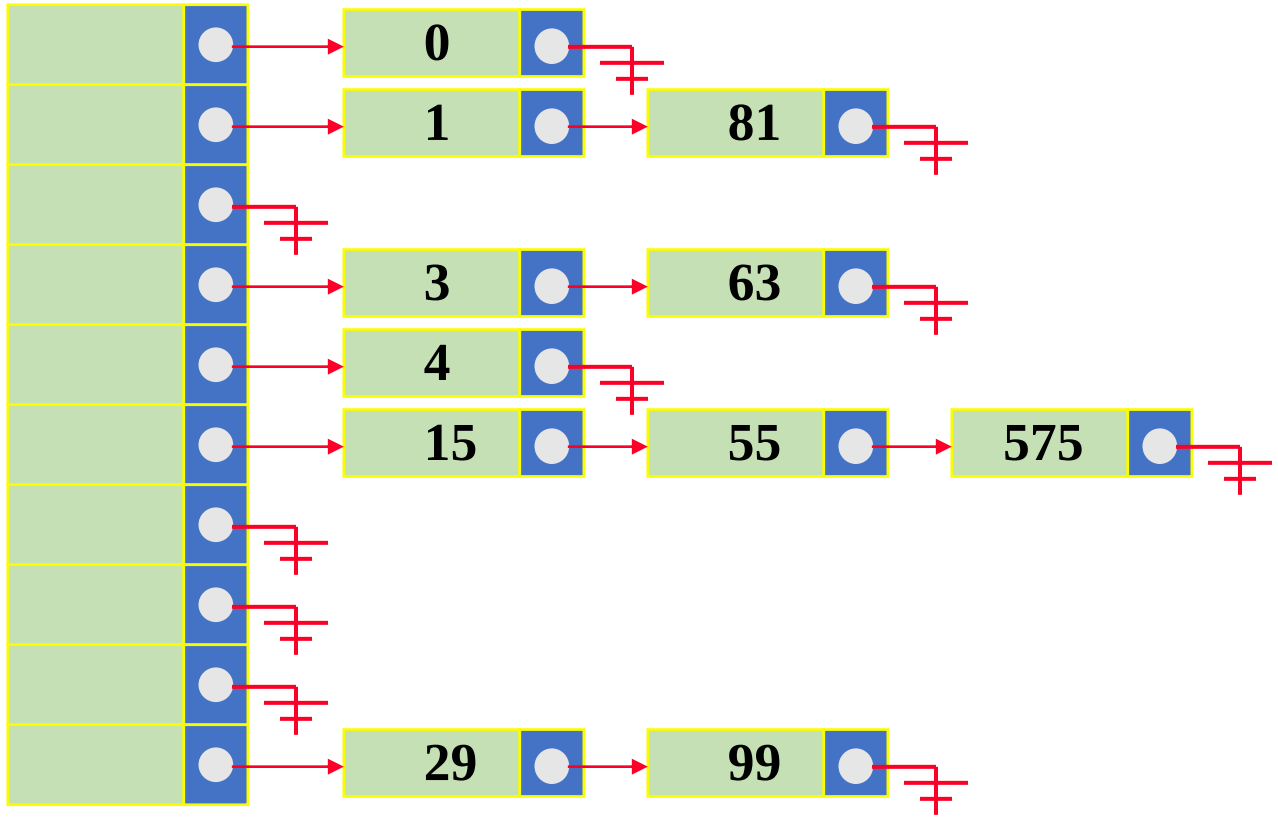
Tal indicador, pode revelar, a medida que seu valor aumenta o risco de ocorrência de colisões

Justificando-se assim, a adoção de estratégias para tratamento de colisões, ao mesmo passo que são formuladas as tabelas hash.

No último encontro, vimos uma dessas estratégias: O tratamento de colisões por encadeamento

Mais especificamente, o encadeamento externo ou exterior (separado) é uma solução fácil de ser implementada

Os nós que representam o endereçamento, correspondem à nós que iniciam listas de elementos com mesmo endereçamento hash



Fonte:https://www.inf.ufsc.br/~aldo.vw/estruturas/Hashing/

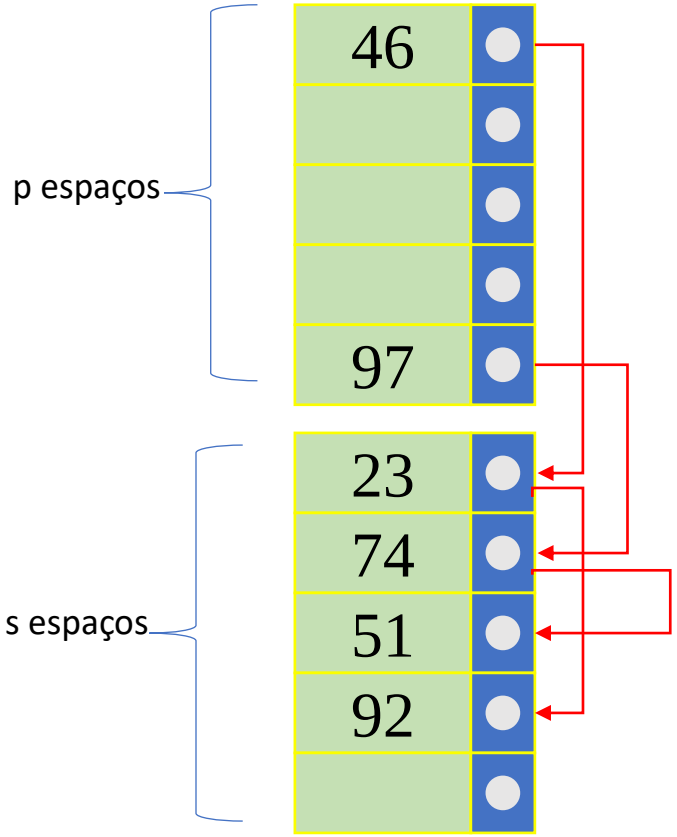
Contudo, podemos observar que a complexidade de pior caso, é referente ao possível comprimento de cada lista de endereçamento externo, sendo O(n)

Em algumas situações tal representação em listas externas não são desejadas, então tem-se como alternativa similar, porém pontualmente diferente, o encadeamento interno

Neste procedimento a tabela é dividida em duas seções: uma correspondendo os endereços de indexações bases, com um certo tamanho p e outra parte, com tamanho s, para os elementos que colidiram com algum contido no endereço base.

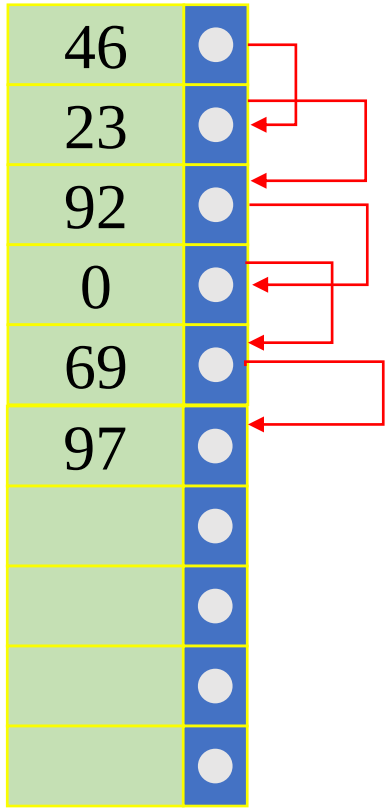
Importante destacar que p e s, possuem tamanhos fixos, e que p+s = m. E que a função hash realiza o calculo do endereçamento no intervalo [0, p-1]

Em termos de estrutura, a tabela hash de encadeamento interno é a mesma que a do encadeamento externo.



Adaptado da Fonte:https://www.inf.ufsc.br/~aldo.vw/estruturas/Hashing/

O que diferencia é que o “apontamento” entre os elementos ocorre por meio de um elemento pertencente ao endereçamento base (p espaços), que detém o início da lista de encadeamento interno aos elementos contidos nos endereços de colisão (s espaços)



Há propostas de não ter-se a separação dos endereços em p e s partes e fazer uso do próximo espaço livre na tabela.

Contudo, é notório que tal estratégia poderá promover colisões secundárias com espaços de endereços hash não vinculados ao endereço hash de origem

Outro problema existente, para além da fusão das listas de encadeamento interno, é o processo de exclusão. Pois ao excluir um elemento, pode-se perder a referência da lista interna.

Uma estratégia para contornar esta situação é a inclusão um rótulo para cada endereçamento da tabela hash: ocupado, vazio ou livre. Estando no estado livre sua ocupação poderá substituída, mas nunca poderá ocorrer a exclusão do elemento

Bom, até o momento vimos uma continuidade dos conceitos obtidos no último encontro da disciplina

Agora passaremos a conhecer algumas estratégias para o tratamento de colisões por endereçamento aberto

Colisões: Tratamento por Endereçamento Aberto

O tratamento de colisões por encadeamento aberto não faz uso de ponteiros e lista para a indexação dos elementos com mesmo endereço hash inicial

Uma das vantagens dessa estratégia é o não uso de ponteiros e o não consumo desse espaço de memória para esse operador

Nesta estratégia, ao ser observado o choque no endereçamento, um novo processo para o encontro de um novo endereço é realizado

Neste caso é possível verificar que há m! possíveis endereços para uma chave

Vamos conhecer algumas estratégias para endereçamento aberto?

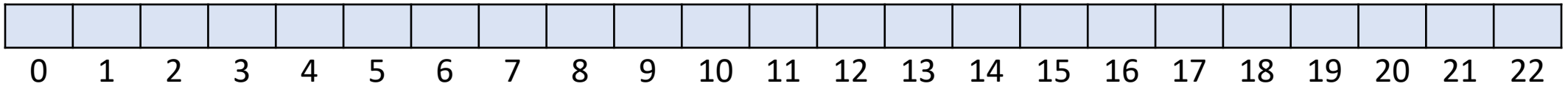
Tentativa Linear (ou Re-hasing Progressivo)

Uma estratégia simples que corresponde a alternativa de buscar, consecutivamente, um endereçamento livre, por meio do recalculo da função hash através de:

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

Promovendo um comportamento de agrupamento dos elementos indexados. Vamos observar um exemplo?

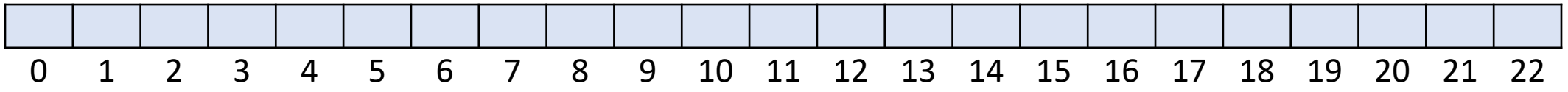
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23



SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de Dados e seus Algoritmos**. Livros Tecnicos e Cientificos, 1994.

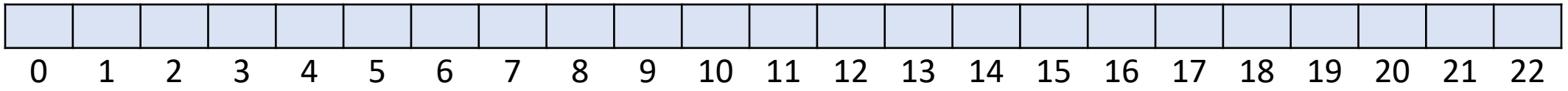
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

44 | 23



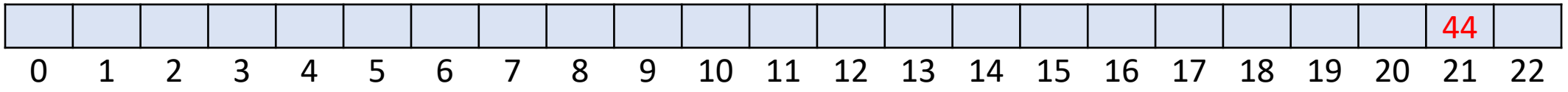
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

$$44 \mid 23$$
$$\quad\; 1$$



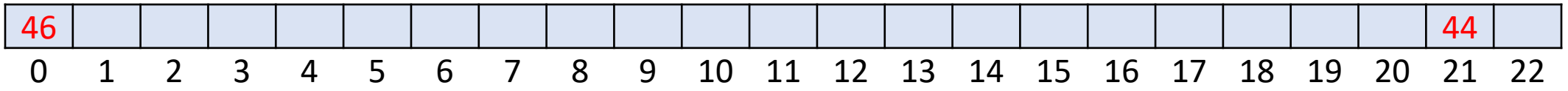
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

44	23
(21)	1



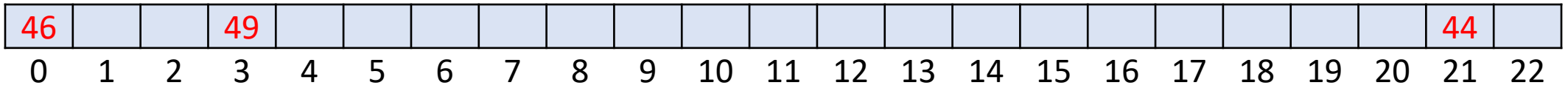
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

46	23
(0)	2



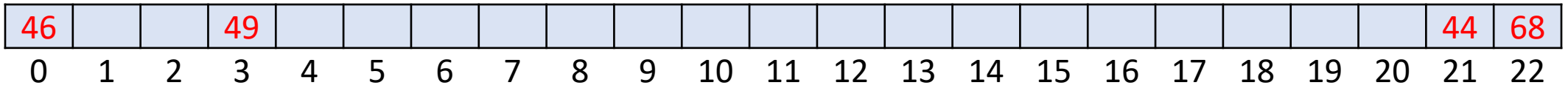
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

49	23
(3)	2



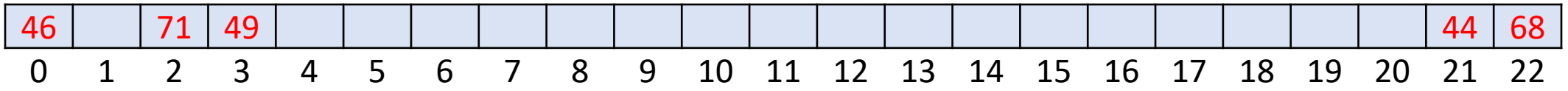
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

68	23
(22)	2



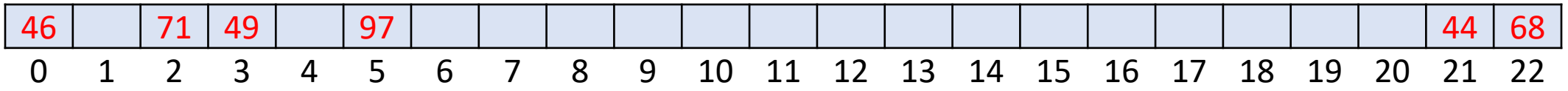
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

71	23
(2)	3

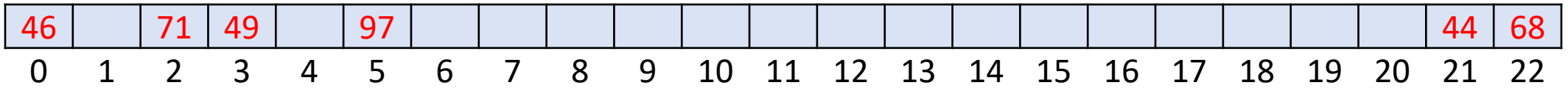


Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

97	23
(5)	4



Dando continuidade, vamos tentar inserir os elementos: 26, 72, 27?



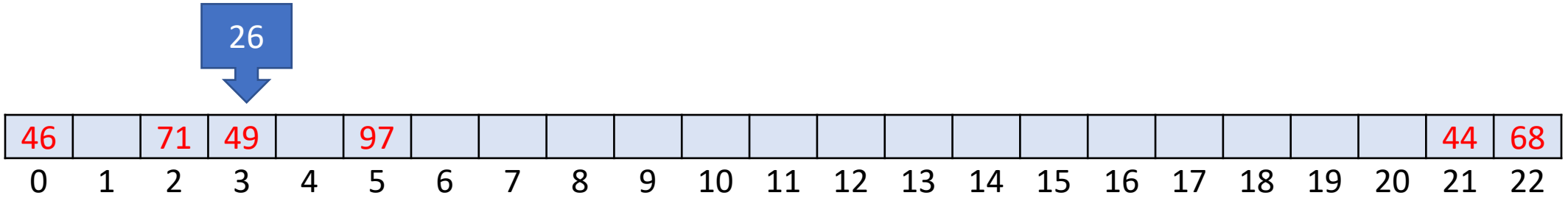
Inserir os elementos: 26, 72, 27

26	23
(3)	1

46		71	49		97															44	68
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Inserir os elementos: 26, 72, 27

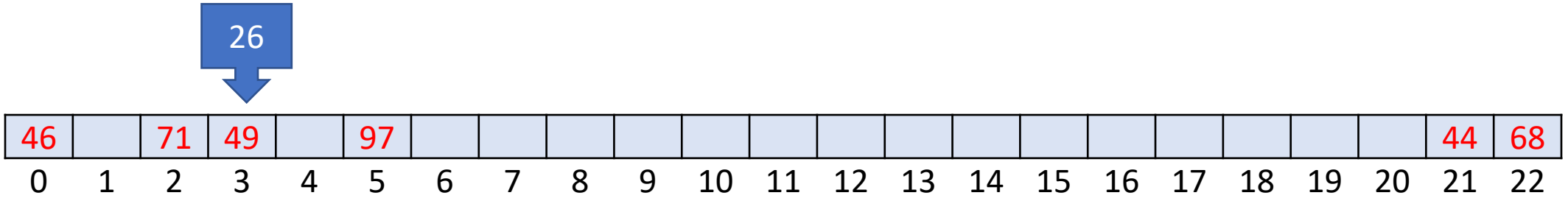
26	23
(3)	1



Inserir os elementos: 26, 72, 27

4	23
(4)	0

26	23
(3)	1

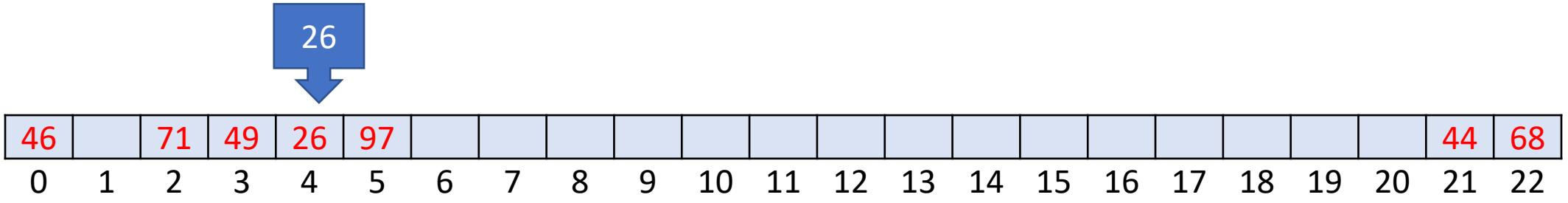


$h(x, k) = (h'(x) + k) \bmod m,\ 0 \leq k \leq m - 1$

$h(26, 0) = (26 + 0) \bmod 23 = 3$

$h(3, 1) = (3 + 1) \bmod 23 = 4$

Inserir os elementos: 26, 72, 27



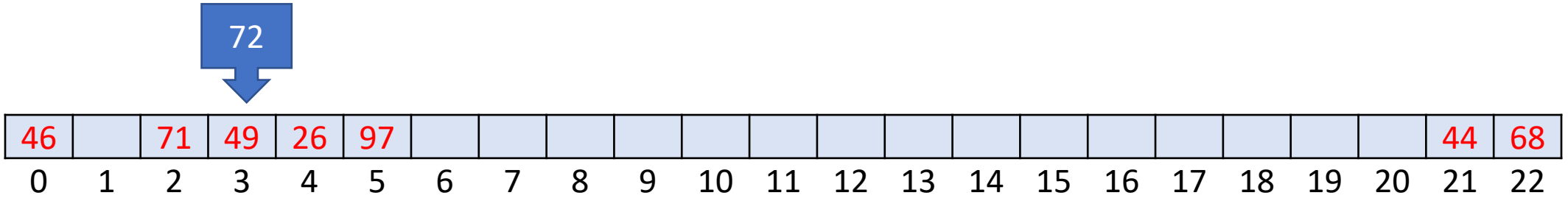
Inserir os elementos: 26, 72, 27

6	23
(6)	0

5	23
(5)	0

4	23
(4)	0

72	23
(3)	3



$h(x, k) = (h'(x) + k) \bmod m,\ 0 \leq k \leq m - 1$
$h(72, 0) = (72 + 0) \bmod 23 = 3$
$h(3, 1) = (3 + 1) \bmod 23 = 4$
$h(4, 1) = (4 + 1) \bmod 23 = 5$
$h(5, 1) = (5 + 1) \bmod 23 = 6$

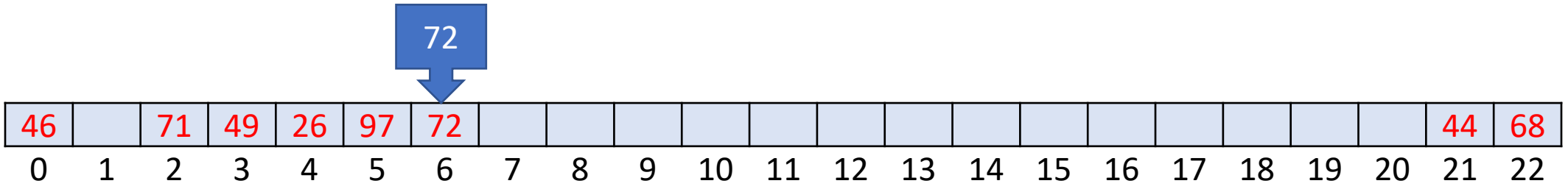
Inserir os elementos: 26, 72, 27

6	23
(6)	0

5	23
(5)	0

4	23
(4)	0

72	23
(3)	3



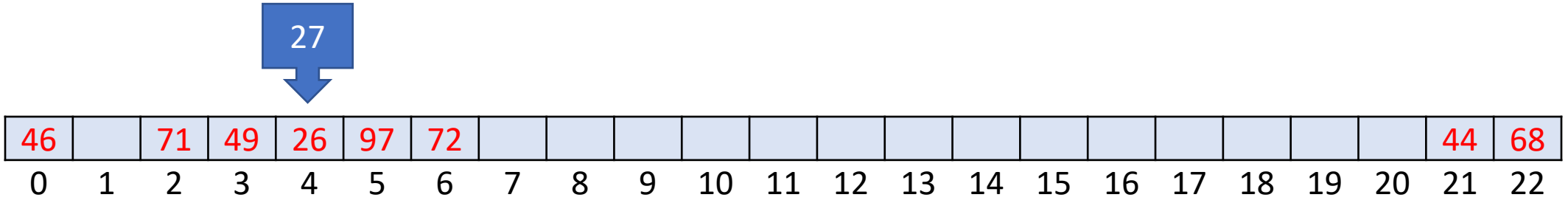
$h(x, k) = (h'(x) + k) \bmod m,\ 0 \leq k \leq m - 1$
$h(72, 0) = (72 + 0) \bmod 23 = 3$
$h(3, 1) = (3 + 1) \bmod 23 = 4$
$h(5, 1) = (5 + 2) \bmod 23 = 7$

Inserir os elementos: 26, 72, 27

7	23
(7)	0

4	23
(4)	0

27	23
(4)	1



$h(x, k) = (h'(x) + k) \bmod m,\ 0 \leq k \leq m - 1$
$h(27, 0) = (27 + 0) \bmod 23 = 4$
$h(4, 1) = (4 + 1) \bmod 23 = 5$
$h(5, 2) = (5 + 2) \bmod 23 = 7$

Inserir os elementos: 26, 72, 27

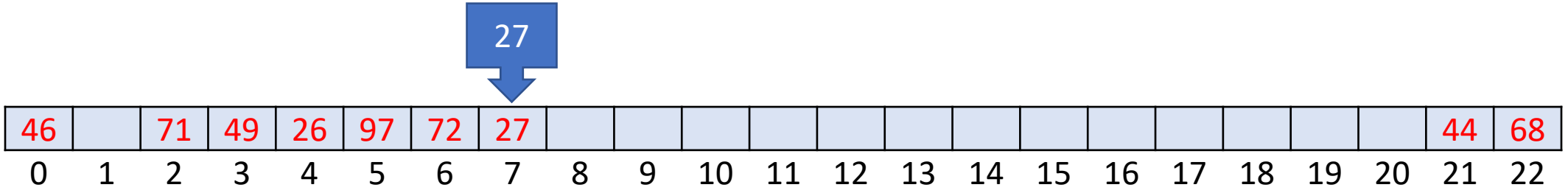
7	23
(7)	0

6	23
(6)	0

5	23
(5)	0

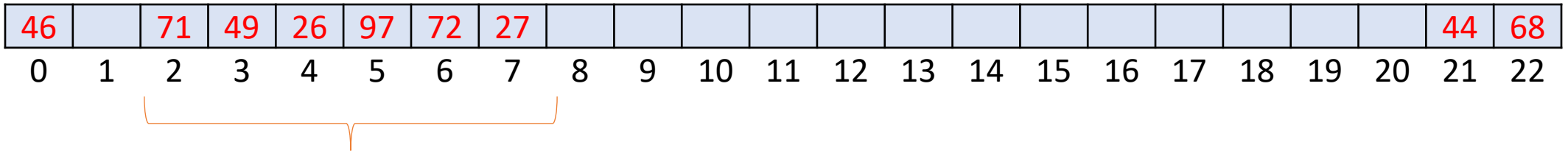
4	23
(4)	0

27	23
(4)	1



$h(x, k) = (h'(x) + k) \bmod m,\ 0 \leq k \leq m - 1$
$h(27, 0) = (27 + 0) \bmod 23 = 4$
$h(4, 1) = (4 + 1) \bmod 23 = 5$
$h(5, 1) = (5 + 1) \bmod 23 = 6$
$h(6, 1) = (6 + 1) \bmod 23 = 7$

Elementos indexados: 44, 46, 49, 68, 71, 97, 26, 72 e 27 na tabela hash cujo m = 23



Este método promove um agrupamento de trechos consecutivos de memória ocupados. Denominado também de agrupamento primário

Double Hash

Ao invés de fazer os incrementos de 1 invariavelmente, os incrementos são feitos por um valor que depende da chave x

Então calculamos o valor do hash:

$$h(x) = x \bmod m = i$$

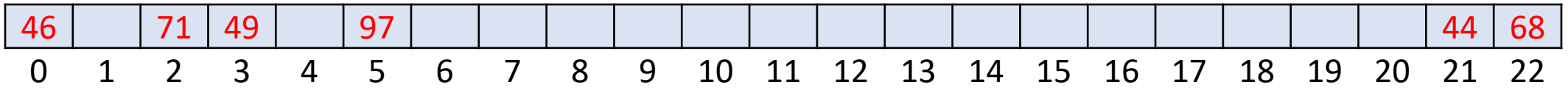
Caso haja colisão, inicialmente calculamos $h_2(K)$, que pode ser definida como:

$$h_2(x) = 1 + (k \bmod (m-1)) = w$$

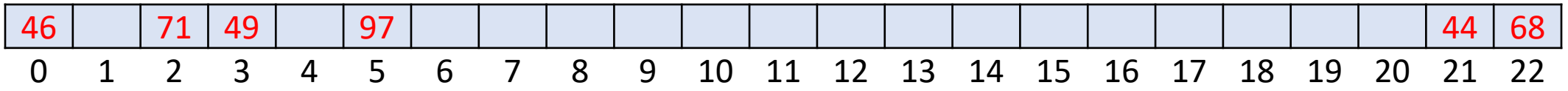
Em seguida calculamos a função re-hashing como sendo:

$$rh(i,w) = (i + w) \bmod m$$

Mesmo exemplo anterior: 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

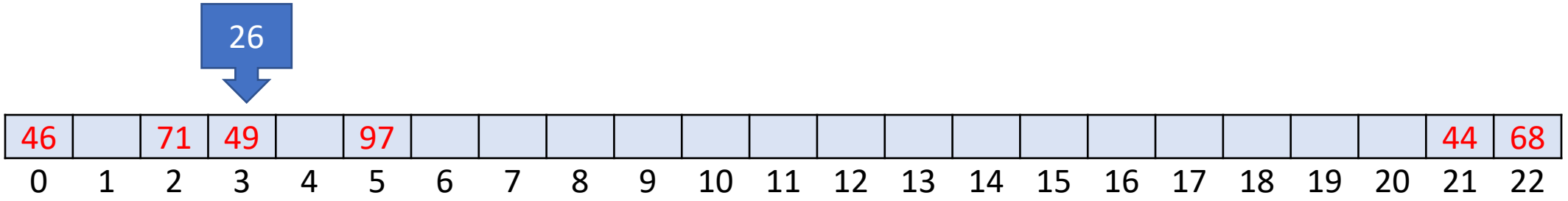


Dando continuidade, vamos tentar inserir os elementos com double hash: 26, 72, 27?



Inserir os elementos: 26, 72, 27

26	23
(3)	1

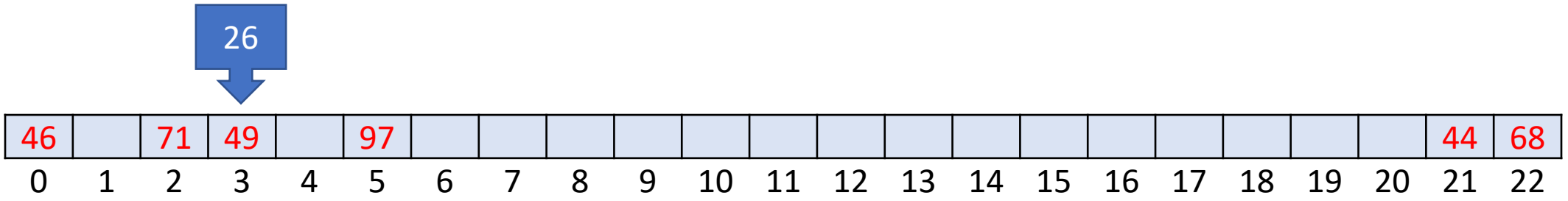


Inserir os elementos: 26, 72, 27

8	23
(8)	0

26	22
(4)	1

26	23
(3)	1



$h(x) = x \bmod m = i$

$h(26) = 26 \bmod 23 = 3 = i$

$h_2(x) = 1 + (x \bmod (m-1)) = w$

$h_2(26) = 1 + (26 \bmod 22) = 1+4 = 5 = w$

$rh(i,w) = (i + w) \bmod m$

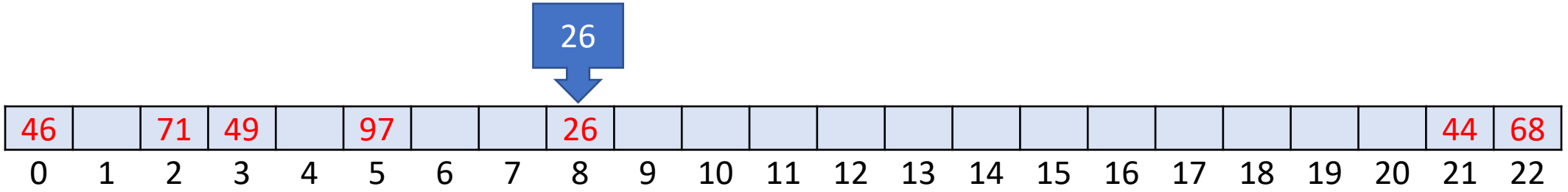
$rh(3,5) = (3 + 5) \bmod 23 = 8$

Inserir os elementos: 26, 72, 27

8	23
(8)	0

26	22
(4)	1

26	23
(3)	1



$h(x) = x \bmod m = i$

$h(26) = 26 \bmod 23 = 3 = i$

$h_2(x) = 1 + (x \bmod (m-1)) = w$

$h_2(26) = 1 + (26 \bmod 22) = 1+4 = 5 = w$

$rh(i,w) = (i + w) \bmod m$

$rh(3,5) = (3 + 5) \bmod 23 = 8$

Inserir os elementos: 26, 72, 27

10	23
(10)	0

72	22
(6)	3

72	23
(3)	3

46		71	49		97			26													44	68
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

$h(x) = x \bmod m = i$

$h(72) = 72 \bmod 23 = 3 = i$

$h_2(x) = 1 + (x \bmod (m-1)) = w$

$h_2(72) = 1 + (72 \bmod 22) = 1+6 = 7 = w$

$rh(i,w) = (i + w) \bmod m$

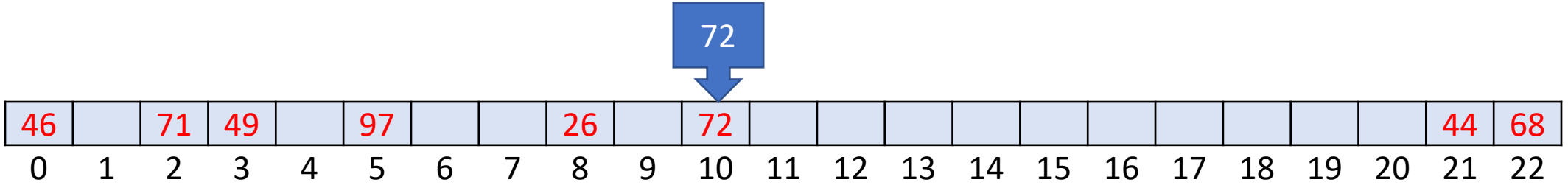
$rh(3,7) = (3+ 7) \bmod 23 = 10$

Inserir os elementos: 26, 72, 27

10	23
(10)	0

72	22
(6)	3

72	23
(3)	3



$h(x) = x \bmod m = i$

$h(72) = 72 \bmod 23 = 3 = i$

$h_2(x) = 1 + (x \bmod (m-1)) = w$

$h_2(72) = 1 + (72 \bmod 22) = 1+6 = 7 = w$

$rh(i,w) = (i + w) \bmod m$

$rh(3,7) = (3+ 7) \bmod 23 = 10$

Inserir os elementos: 26, 72, 27

16	23
(16)	0

10	23
(10)	0

27	22
(5)	1

27	23
(4)	1

46		71	49		97			26		72											44	68
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

h(x) = x mod m = i

h(27) = 27 mod 23 = 4 = i

$h_2(x) = 1 + (x \bmod (m-1)) = w$

$h_2(27) = 1 + (27 \bmod 22) = 1+5 = 6 = w$

rh(i,w) = (i + w) mod m

rh(4,6) = (4+6) mod 23 = 10 → COLISÃO, então 10 será w e 6 será i

rh(6,10) = (6+10) mod 23 = 16

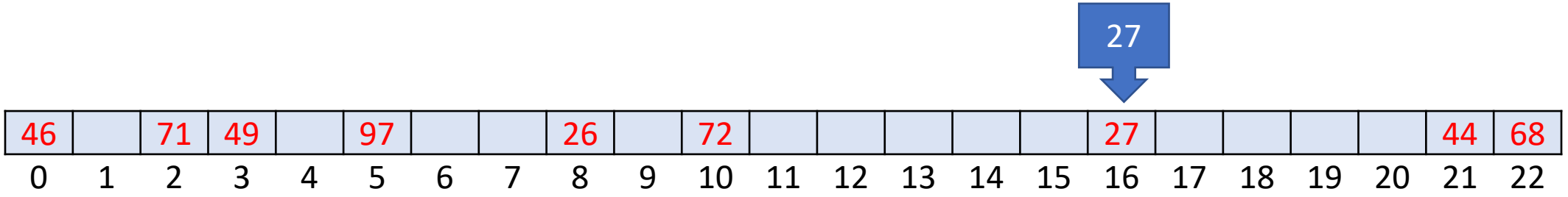
Inserir os elementos: 26, 72, 27

16	23
(16)	0

10	23
(10)	0

27	22
(5)	1

27	23
(4)	1



$h(x) = x \bmod m = i$

$h(27) = 27 \bmod 23 = 4 = i$

$h_2(x) = 1 + (x \bmod (m-1)) = w$

$h_2(27) = 1 + (27 \bmod 22) = 1+5 = 6 = w$

$rh(i,w) = (i + w) \bmod m$

$rh(4,6) = (4+6) \bmod 23 = 10 \rightarrow$ COLISÃO, então 10 será w e 6 será i

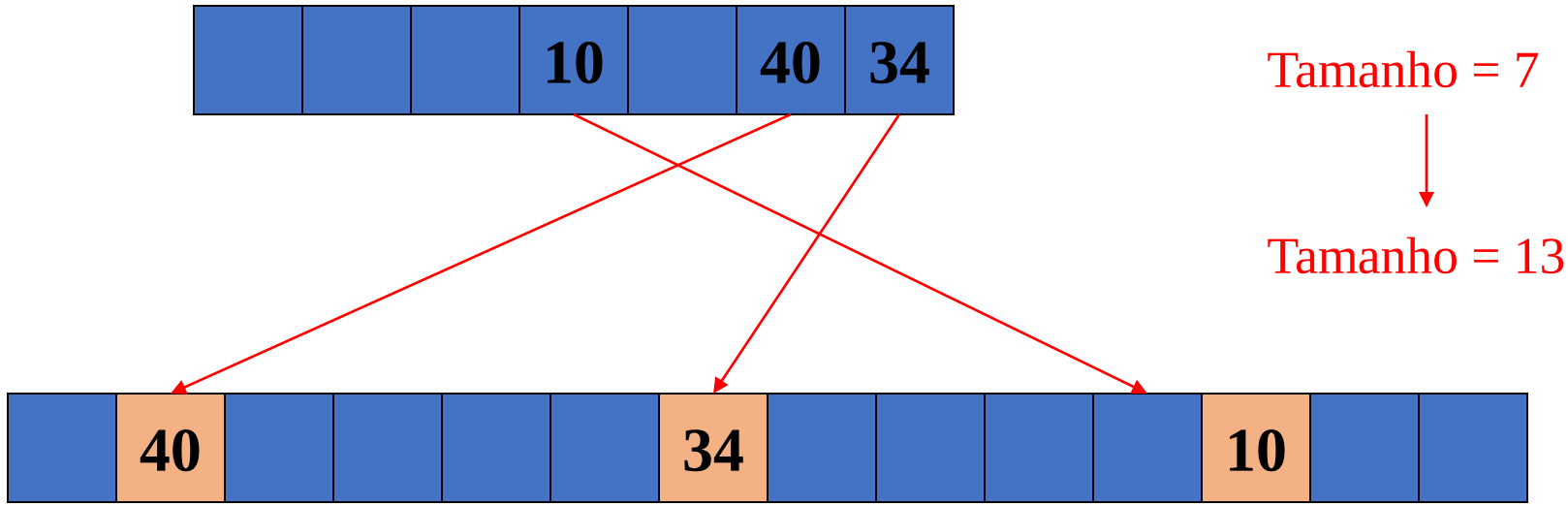
$rh(6,10) = (6+10) \bmod 23 = 16$

Lembrando que:

Para qualquer estratégia adotada em tabela hash é de grande importância o acompanhamento da taxa de ocupação da estrutura, ou para redimensionamentos automáticos ou revisão dos procedimentos adotados.

Visto que, ao passo que temos mais de elementos, mais tratamentos de colisões vamos realizar e com isso o processo de alocação ou recuperação dos elementos demandará mais tempo.

Quando isto ocorre, uma estratégia é redimensionar e transferir os elementos para uma nova tabela.
Necessitando então o recalculo do hash para a nota tabela de indexação



Próxima semana
Trabalharemos na Disciplina:

Tries

Fontes e Referências:

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de Dados e seus Algoritmos**. Livros Tecnicos e Cientificos, 1994.