Tabelas Hash

Mais uma nova estrutura arbórea?

Não. Mas também não deixa de ser uma estrutura de indexação

Isso mesmo, os elementos serão indexados/armazenados, mas não por uma ordem, um hierarquia como vínhamos trabalhando desde o início do semestre

Os elementos, aqui neste tópico que vamos iniciar, serão indexados também por suas chaves

Semelhante a um dicionário, em que o agrupamento dos elementos são realizados por meio de uma característica em comum

Não necessitando ter uma relação entre esses elementos

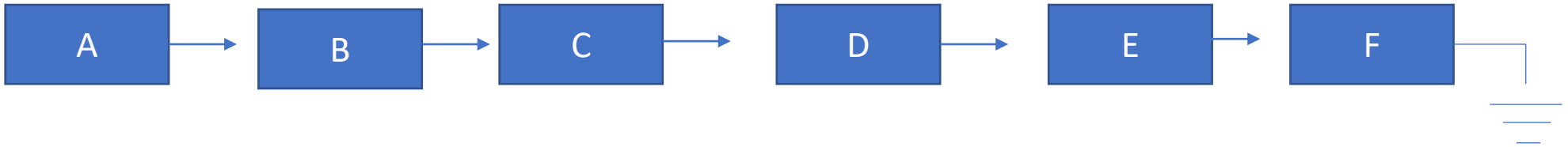
Nessa estrutura há um mapeamento entre chave e valor

Suas contribuições estão no processo de armazenar e acessar os dados indexados por suas chaves na memória, em complexidade média de O(1), contudo sendo o pior caso O(n)

Vamos imaginar algumas situações: até agora se pensarmos em armazenar um conjunto de elementos por uma chave, ou tentaremos organizar por certa prioridade ou realizar uso de alguma outra estrutura que faz uso de estratégias de comparação de chaves, como as listas, não é mesmo?

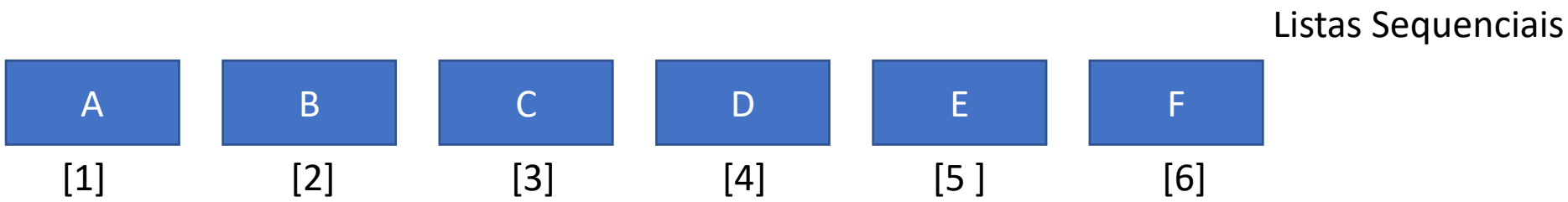
Listas Sequenciais

Listas Encadeadas



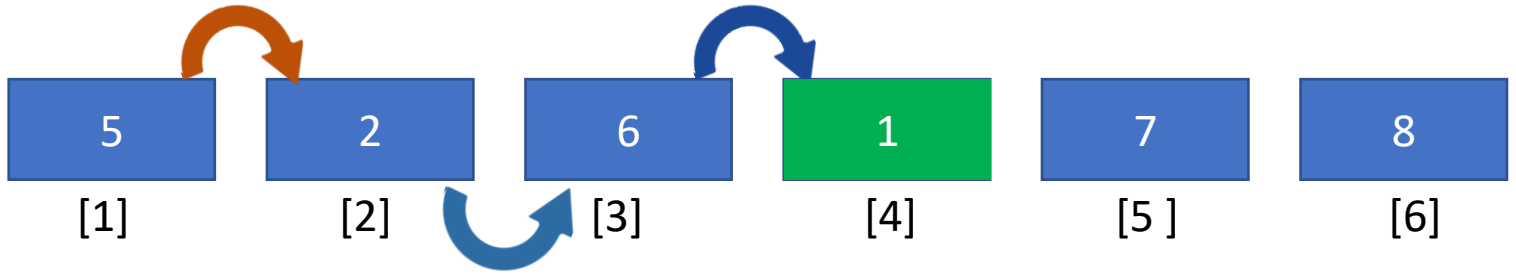
Não estamos errados com essa perspectiva, mas há um porém

Para esse caso, pensando em uma lista sequencial, a estrutura deve ter em si, para n chaves, n espaços par uma indexação direta, não é mesmo?



Vamos fazer uma busca?

Buscar o elemento 1?



Qual o tempo de acesso?

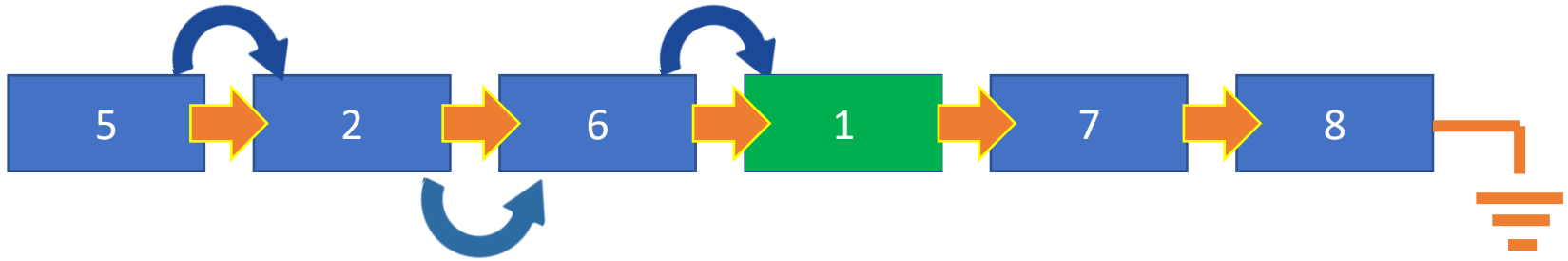
O(1)

Qual o tempo busca?

Bom, mas você pode pensar nas listas encadeadas, talvez. Será que é diferente?

Vamos fazer uma busca?

Buscar o elemento 1?

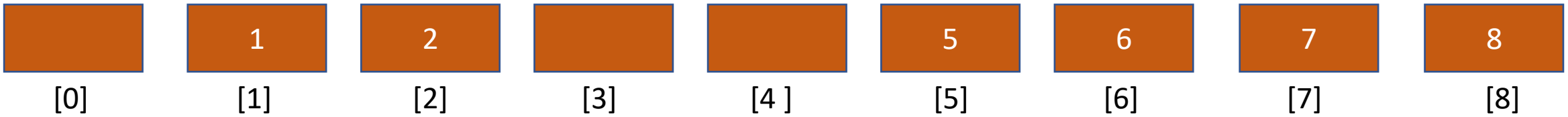


Qual o tempo de acesso?

O(1)

Qual o tempo busca?

Bom, mas podemos pensar em uma solução para ser O(1), não é mesmo?



Um exemplo bem extremo de SZWARCFITER; MARKENZON (1994): seja um conjunto de duas chaves apenas, de valores 0 e 999.999, respectivamente. A aplicação da técnica de acesso direto conduziria a uma tabela com 1.000.000 compartimentos

Nas tabelas hashs não são necessários n espaços para armazenar n elementos, podemos armazena-los em uma estrutura de m espaços, tal que $n \leq m$

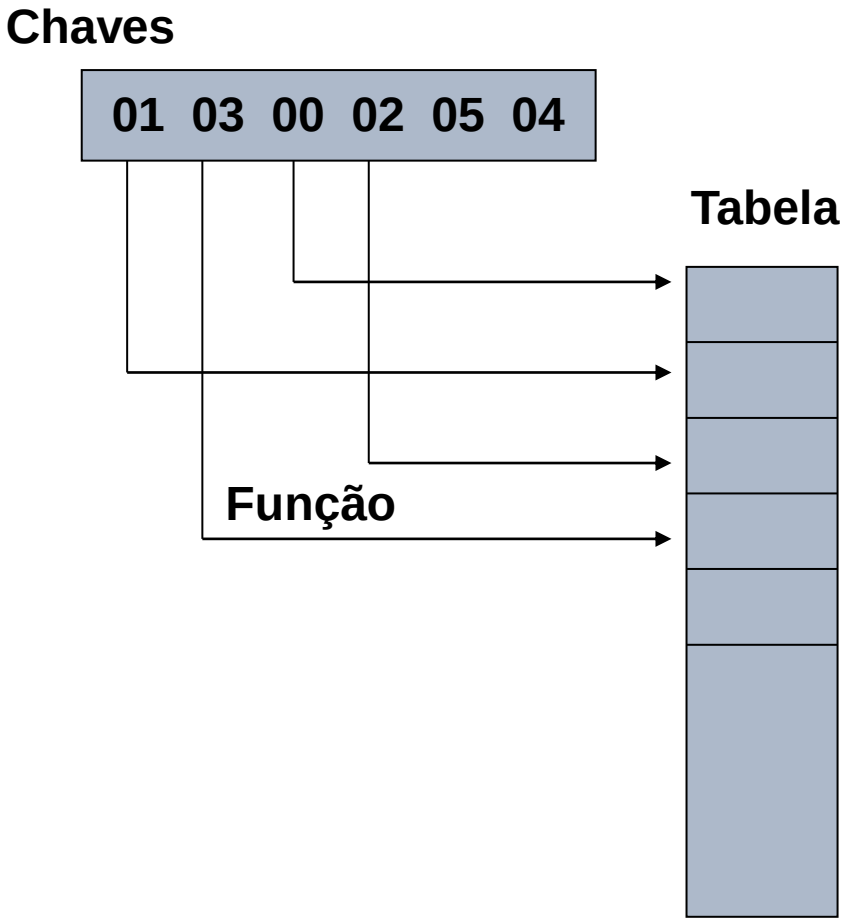
E como são essas tabelas?

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.

"Quando a tabela se encontra na memória é comum que cada compartimento armazene apenas um nó. Compartimentos com diversos nós são usados geralmente por tabelas de dispersão armazenadas em unidades de disco, quando o número de acessos é mais importante do que o uso eficiente de memória. Nesse caso, cada compartimento ocupa, tipicamente, um bloco do disco. O objetivo é armazenar cada chave no bloco referente a seu endereço. A busca, assim, requer somente um acesso a um bloco do disco. "

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

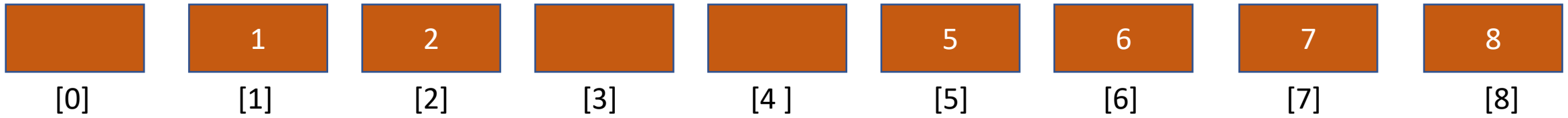
Bom, aqui na nossa disciplina, na maioria das vezes vamos considerar que em cada espaço teremos um elemento único, combinado?



"Para o tratamento do caso geral, um obstáculo aparente é o fato de que as chaves nem sempre são valores numéricos. Por exemplo, as chaves podem consistir em nomes de pessoas. Na realidade, essa questão pode ser facilmente contornável, pois a todo dado não numérico corresponde uma representação numérica no computador."

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

Bom, mas como não indexar n chaves naquelas n posições?



Se tentarmos transformar as n chaves em um valor entre [0, m-1] é possível acessá-las em tempo O(1)

Mas para transformar cada chave x em um valor neste intervalo, é necessário 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

Bom, mas 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)

Esta situação denominamos de colisão, uma situação de “choque” que pode ser contornada com uso de “tratamentos de colisões”, que busca encontrar um novo espaço de armazenamento para a chave que identificou a colisão

Bom, mas antes de falarmos sobre alguns tratamentos de colisões. Vamos conhecer algumas formas de descrever as funções hashs?

Na literatura é dito que uma boa função hash deve possuir algumas características:

- produzir um número baixo de colisões – é interessante conhecer a composição/característica das chaves para determinar uma função hash para este conjunto.
- ser facilmente computável – tempo de consumo para o calculo da função
- ser uniforme – a probabilidade de distribuição dos elementos é a mesma para qualquer valor de chave

Método da Divisão

Um dos métodos mais aplicado, por ser acessível e de fácil entendimento.

Consiste em obter o resto da divisão da chave x pela dimensão da tabela m.

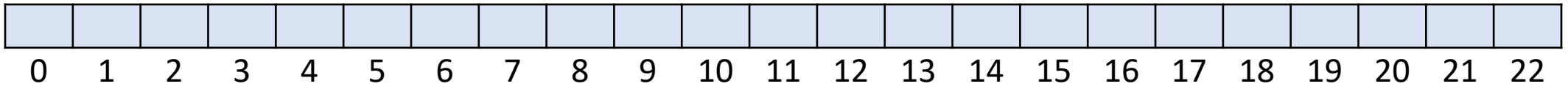
$$h(x) = x \bmod m$$

Tal procedimento revela uma característica:

- Quando m for par h(x) será par quando x for par
- Quando m for ímpar h(x) será ímpar quando x for ímpar

Contudo, na literatura é considerado como uma boa estratégia o uso de um valor primo para m, ou não próximo da potência de 2, ou que não possua divisores primos menores que 20. Tal como: 13, 23, por exemplo.

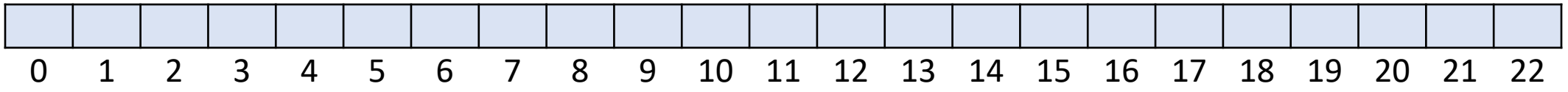
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23



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

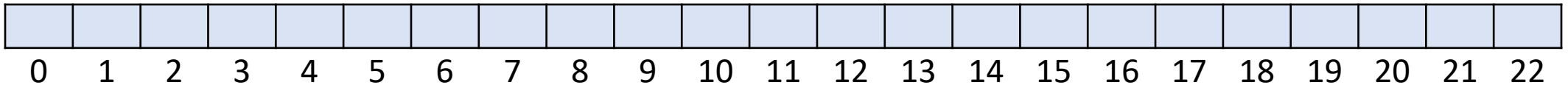
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

44 | 23



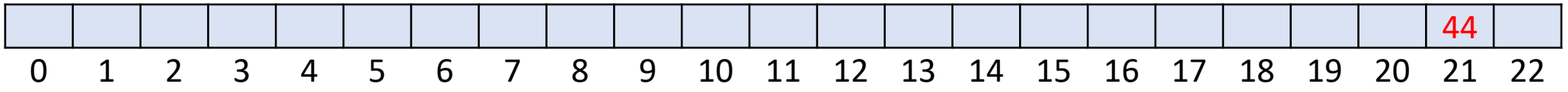
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

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



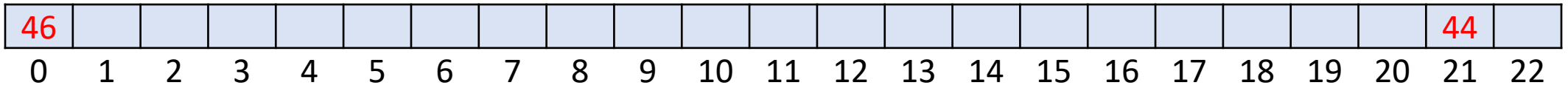
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

44	23
(21)	1



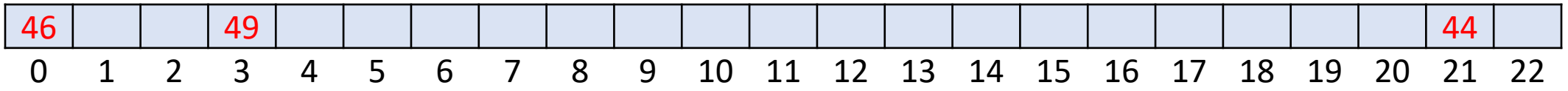
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

46	23
(0)	2



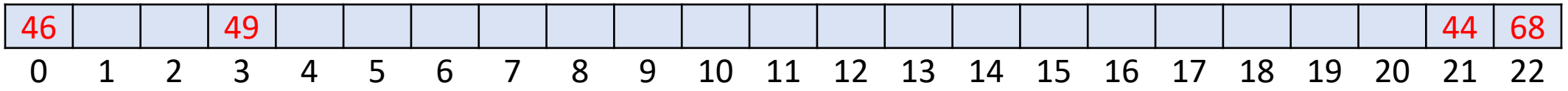
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

49	23
(3)	2



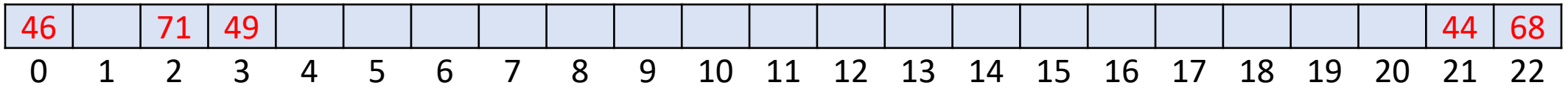
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

68	23
(22)	2



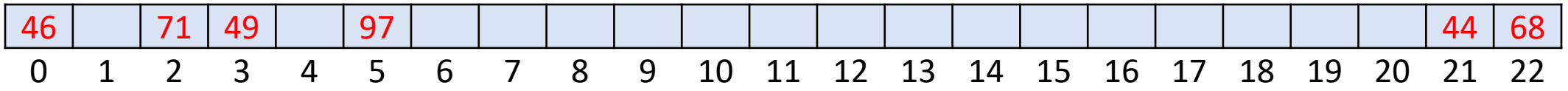
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

71	23
(2)	3



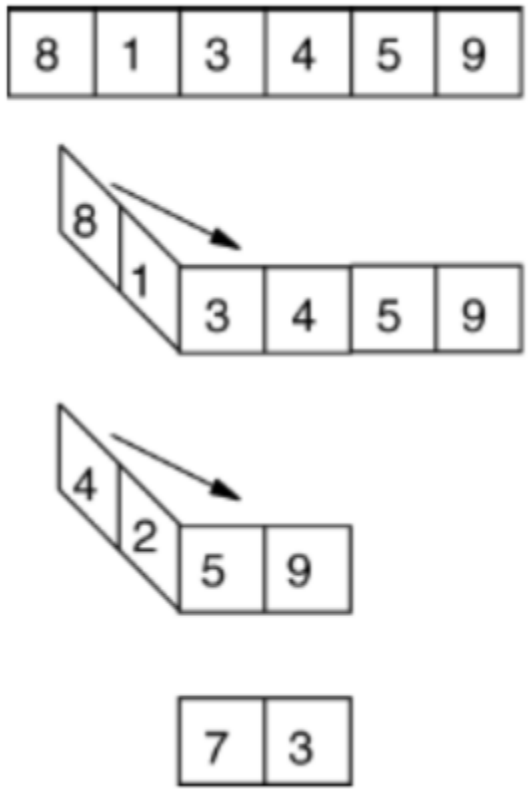
Exemplo: Incluir as chaves 44, 46, 49, 68, 71 e 97 na tabela hash cujo m = 23

97	23
(5)	4



Método da Dobra

O processo consiste em dobrar e somar os dígitos de x para obter sua posição



8+4=~~1~~2 ⬅ Quando somados não é considerado o "vai um"

1+3=4

4+9=~~1~~3

2+5=7

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

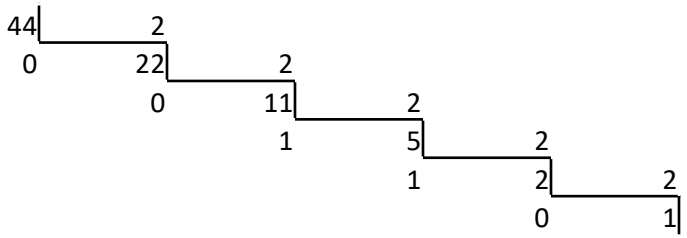
O mesmo procedimento pode ser aplicado tomando como base o valor de x em bits

Para isso basta obter a equivalência do número em bits, por exemplo: 71 é equivalente a 00010 00111

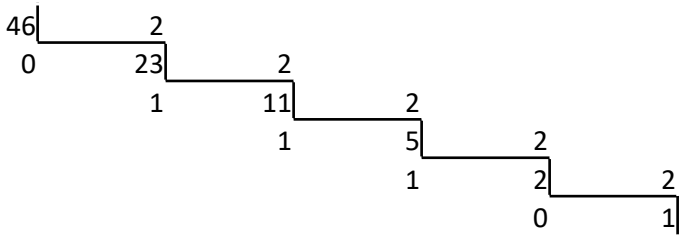
Usando a operação do ou exclusivo:

ou exclusivo

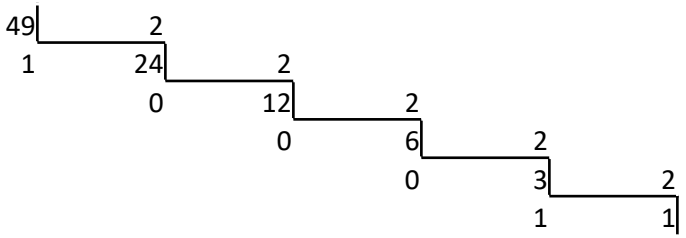
00010
00111
_____
00101 = 5



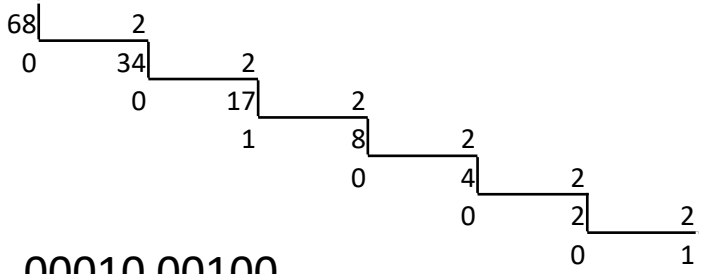
00001 01100



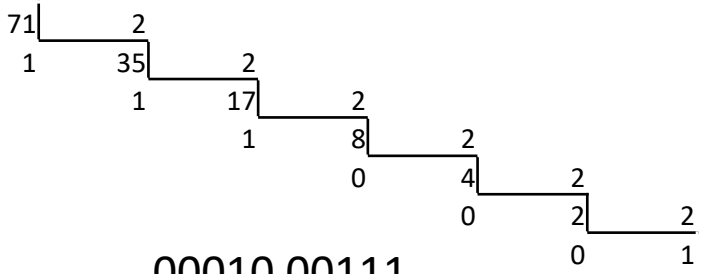
00001 01110



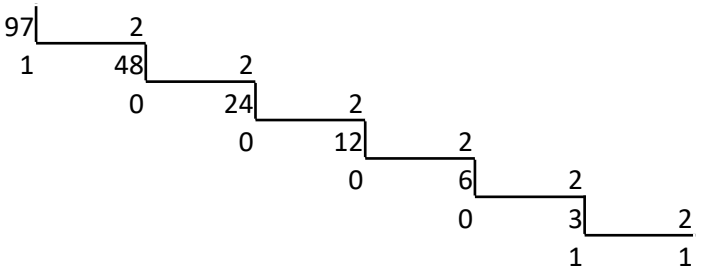
00001 10001



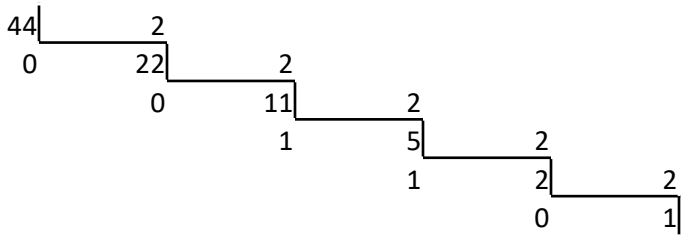
00010 00100



00010 00111

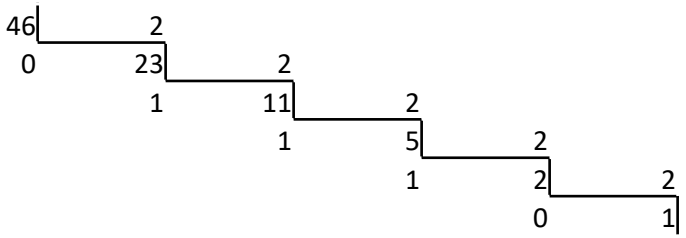


00011 00001



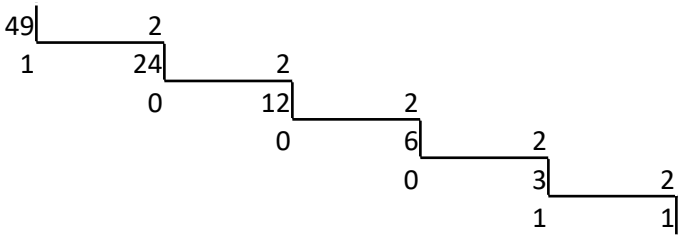
00001
01100
01101 = 13

ou exclusivo



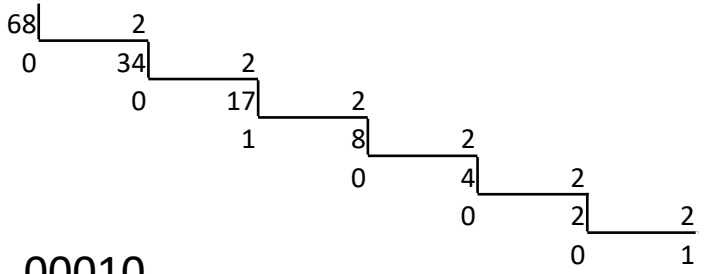
00001
01110
01111 = 15

ou exclusivo



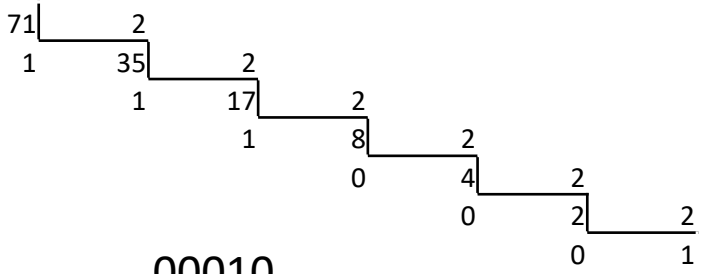
00001
10001
10000 = 16

ou exclusivo



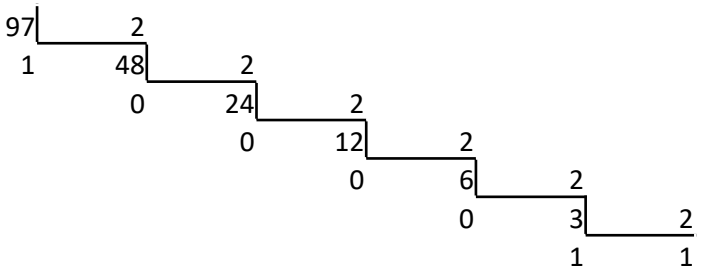
00010
00100
00110 = 6

ou exclusivo



00010
00111
00101 = 5

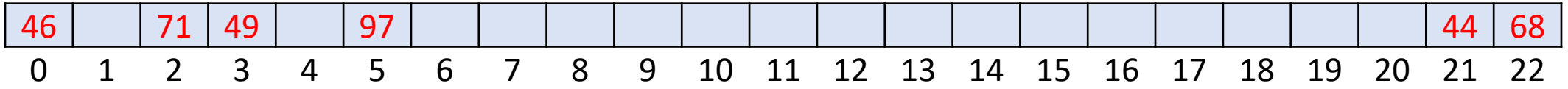
ou exclusivo



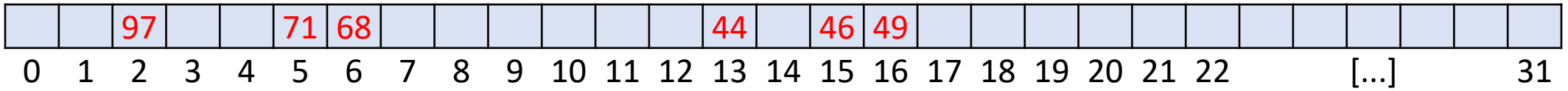
00011
00001
00010 = 2

ou exclusivo

Método da Divisão$_{m = 23}$



Método da Dobra$_{m = 32, \text{pois } 2^5}$



Método da Multiplicação

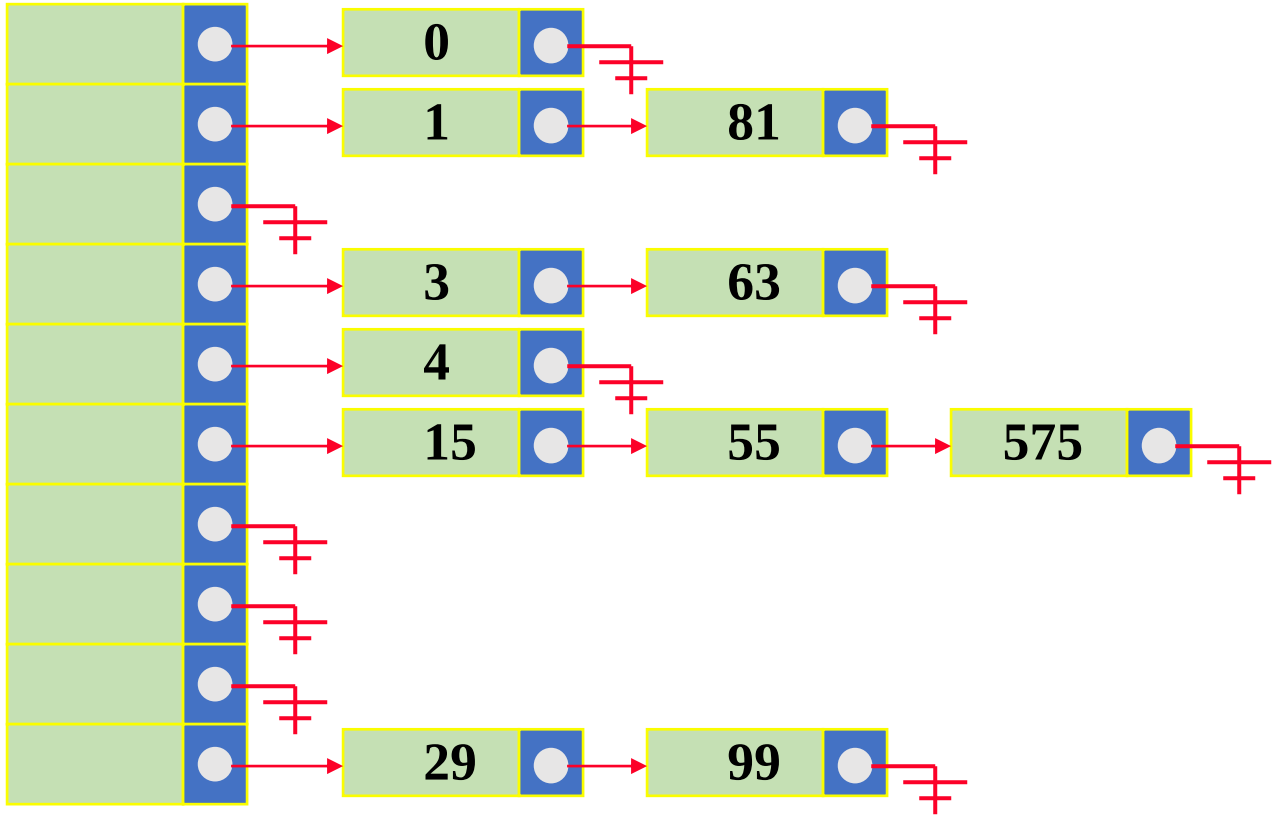
Neste método, há diversas abordagem. Uma delas é utilizarmos o tamanho da palavra do computado (w) e o número inteiro qualquer(b). Considerando que a tabela hash possui m espaços = 2^b

Obtendo-se assim um valor dado por : $\frac{b}{w}$ de modo que b e w sejam primos

$$h(x) = x * b/w$$
$$h(x) = h(x) \bmod m$$

E como implementar uma tabela hash?

Vamos ver uma abordagem que traz essa característica apresentada na imagem, conhecida por Hashing Encadeamento Separado (externo)



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

Fontes e Referências:

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