Índices: Hash

ALGORITMOS E ESTRUTURAS DE DADOS III

Prof. Marcos André S. Kutova

Tabela de dispersão

- As tabelas de dispersão (*hash*) em disco também podem ser usadas como índices, ao invés das árvores.
- Nessas tabelas, o curso de acesso é O(1).
- A posição do registro é determinada por uma função de dispersão (ou função *hash*).

Função de dispersão

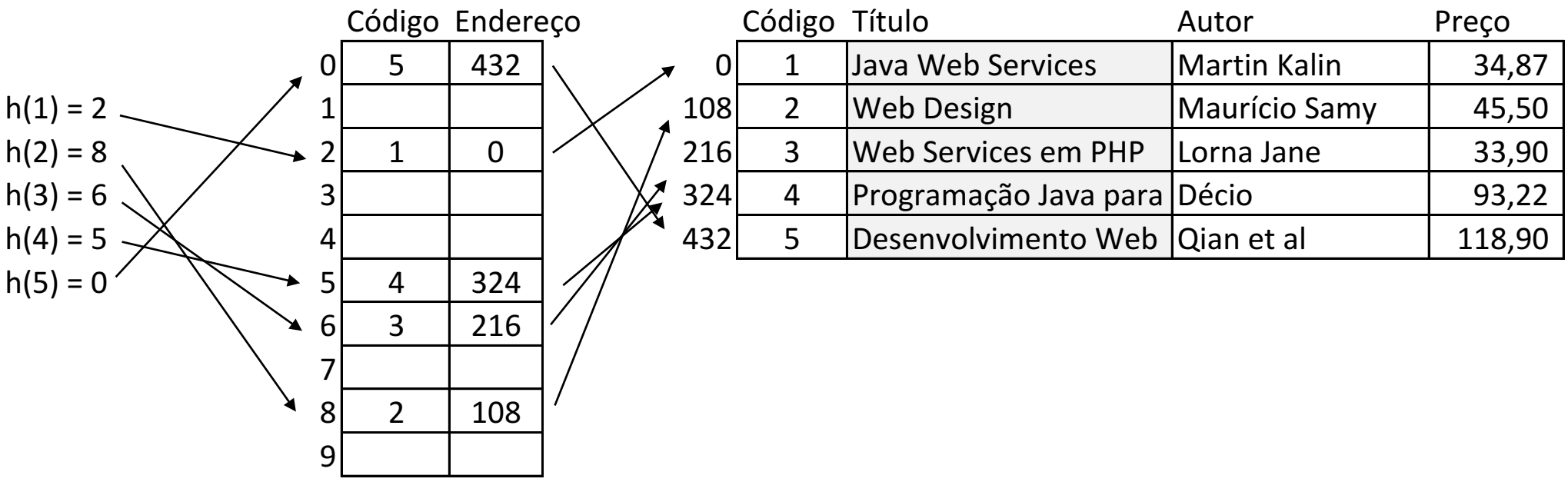
- h(chave) → endereço

h(3204) = 504

chave endereço

- Depende do número de endereços e da natureza da chave.
- Registros do índice devem ser de tamanho fixo.
- Quantidade fixa de endereços
  (depende do tratamento de colisões)

Tabela de dispersão



	Código	Endereço
0	5	432
1		
2	1	0
3		
4		
5	4	324
6	3	216
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,50
216	3	Web Services em PHP	Lorna Jane	33,90
324	4	Programação Java para	Décio	93,22
432	5	Desenvolvimento Web	Qian et al	118,90

Exemplos de função de dispersão

- Elevar a chave ao quadrado e
  pegar um grupo de dígitos do meio:

  $A = h(453) \rightarrow 453^2 = 205209 \rightarrow A = 52$

  (dois dígitos foram escolhidos pois o arquivo possui apenas 100 endereços)

Exemplos de função de dispersão

- Mudar a chave para outra base:

  A = h(453) → $453_{10} = 382_{11}$ →

  382 mod 99 = 85 → A = 85

  (99 é a quantidade de endereços no arquivo)

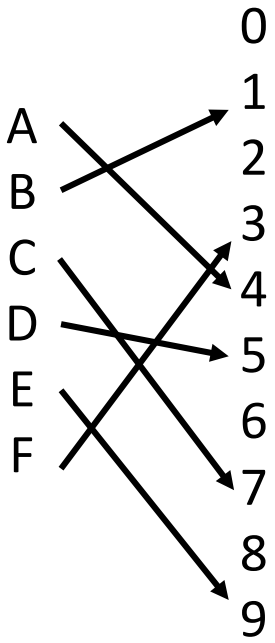
Exemplos de função de dispersão

- Multiplicar o valor ASCII das letras e usar o resto da divisão pelo número de endereços

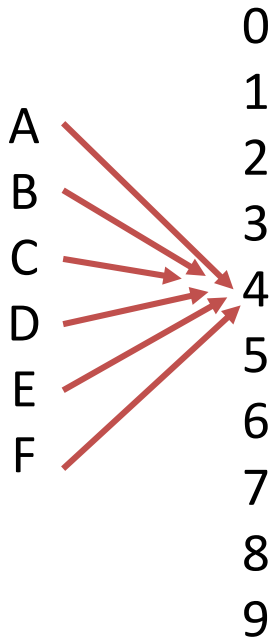
Chave	Cálculo	Endereço
JOÃO	74 x 79 = 5846	846
CARLOS	67 x 65 = 4355	355
GILBERTO	71 x 73 = 5183	183

Colisões

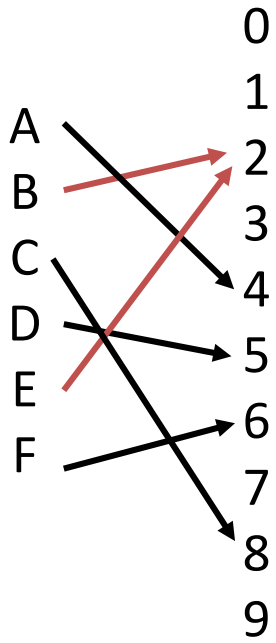
Colisões



DESEJÁVEL



PIOR CASO



ACEITÁVEL

Tratamento de colisões

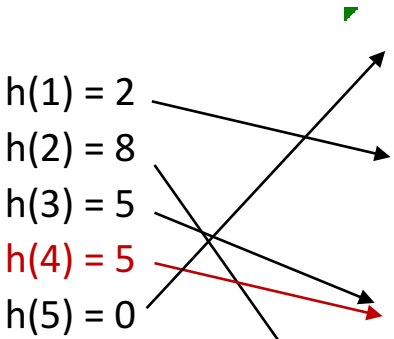
- Alternativas:
  - **Encadeamento interno** – usa outras posições vazias dentro da própria da tabela *hash*
  - **Encadeamento externo** – usa uma área extra, além da tabela *hash*, como uma área de extensão, ou um segundo arquivo.

Encadeamento interno

- Endereçamento aberto - uma nova posição **dentro da área da tabela** será procurada
  - Sondagem linear
  - Sondagem quadrática
  - Duplo *hash* (*double hashing*)

Encadeamento interno

- Sondagem linear – as próximas posições são sondadas (circularmente), até que uma posição livre seja encontrada.



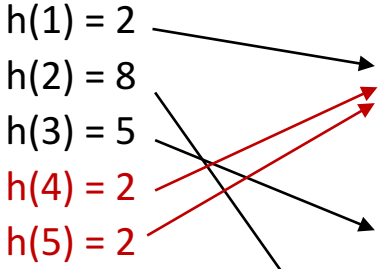
	Código	Endereço
0	5	432
1		
2	1	0
3		
4		
5	3	216
6	4	324
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design Responsivo	Maurício Samy Silva	45,50
216	3	Web Services em PHP	Lorna Jane Mitchell	33,90
324	4	Programação Java para a Web	Décio Heinzelmann	93,22
432	5	Desenvolvimento Web Java	Qian et al	118,90

Regra: $h(k,i) = [h(k) + i] mod\ n$

Encadeamento interno

- Sondagem quadrática – a distância até a próxima posição a ser sondada é determinada pelo quadrado da tentativa



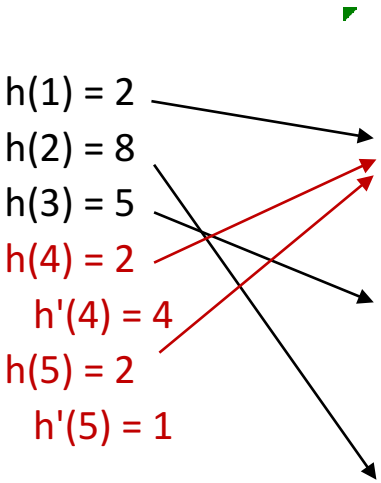
	Código	Endereço
0		
1		
2	1	0
3	4	324
4		
5	3	216
6	5	432
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design Responsivo	Maurício Samy Silva	45,50
216	3	Web Services em PHP	Lorna Jane Mitchell	33,90
324	4	Programação Java para a Web	Décio Heinzelmann	93,22
432	5	Desenvolvimento Web Java	Qian et al	118,90

Regra: $h(k,i) = [h(k) + i^2] \; mod \; n$

Encadeamento interno

- Duplo *hash* – a distância até a próxima posição a ser sondada é determinada por uma segunda função *hash*



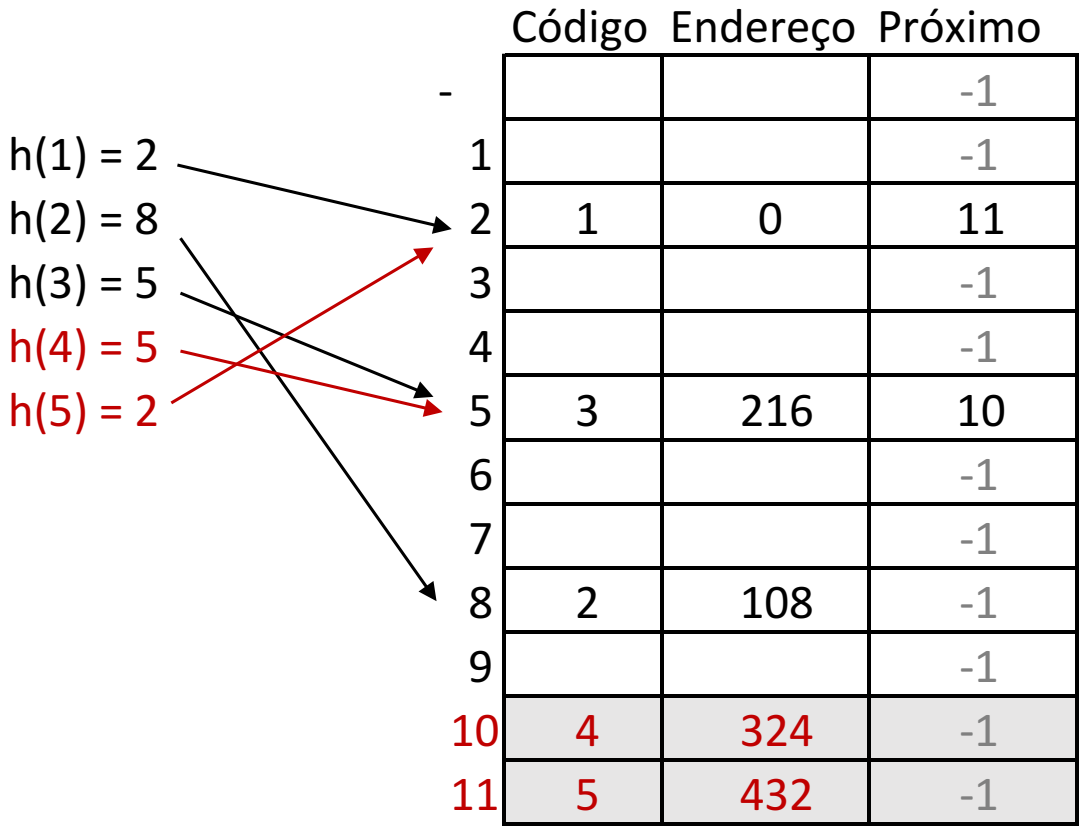
	Código	Endereço
0		
1		
2	1	0
3	5	432
4		
5	3	216
6	4	324
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design Responsivo	Maurício Samy Silva	45,50
216	3	Web Services em PHP	Lorna Jane Mitchell	33,90
324	4	Programação Java para a Web	Décio Heinzelmann	93,22
432	5	Desenvolvimento Web Java	Qian et al	118,90

Regra: $h(k,i) = [h(k) + i * h'(k)] \, mod \, n$

Encadeamento externo

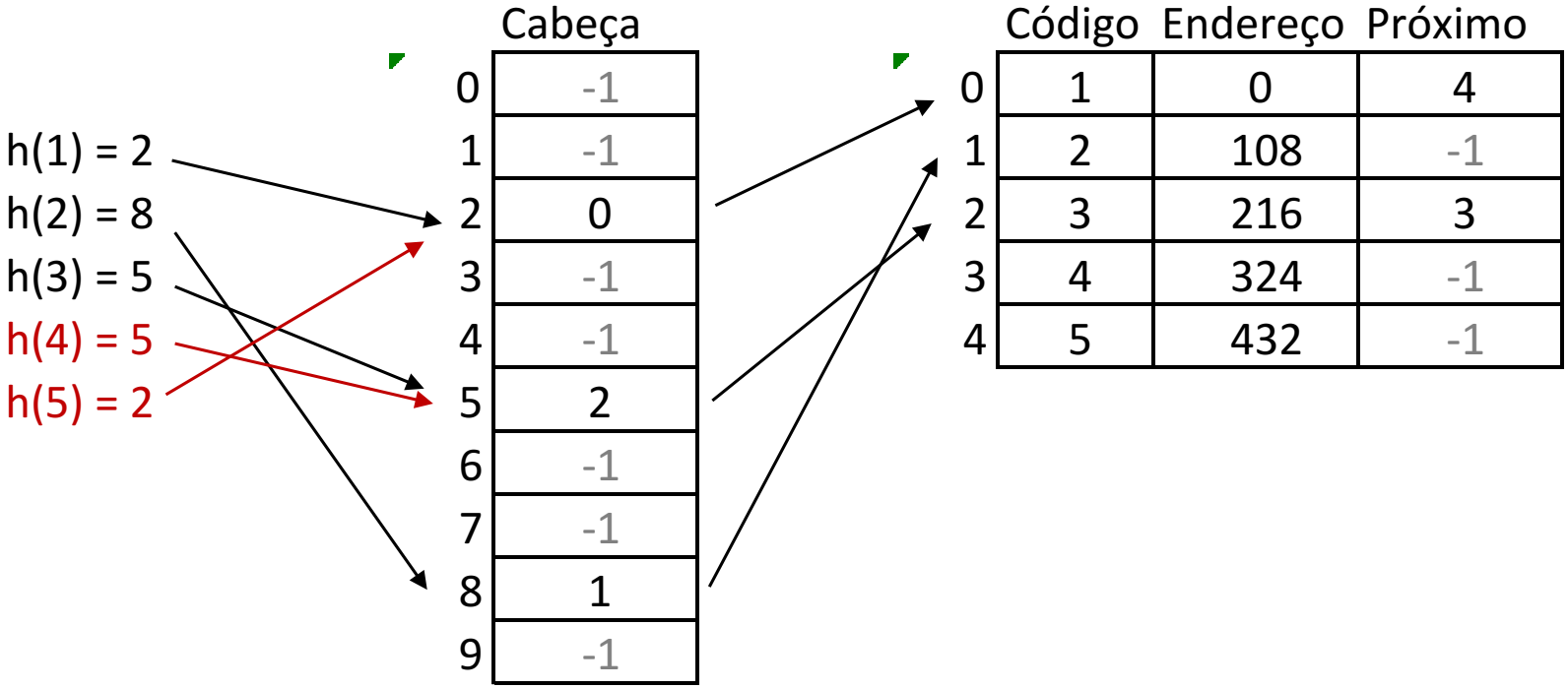
- Área de extensão – os registros colididos são armazenados em uma área de extensão



	Código	Endereço	Próximo
-			-1
1			-1
2	1	0	11
3			-1
4			-1
5	3	216	10
6			-1
7			-1
8	2	108	-1
9			-1
10	4	324	-1
11	5	432	-1

Encadeamento externo

- Lista encadeada – todos os registros são armazenados em uma lista encadeada (outro arquivo)

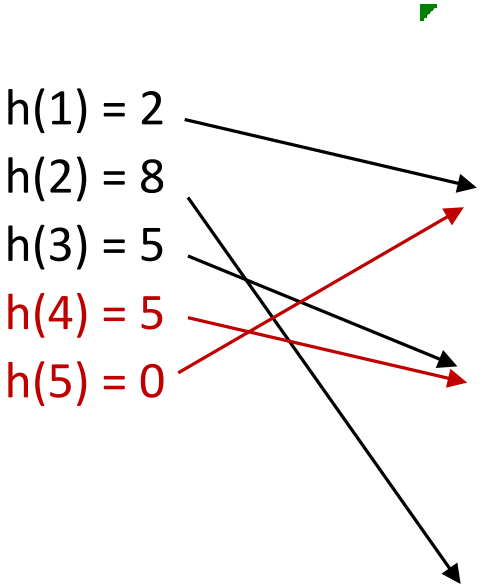


Buckets

Buckets (cesto)

- Da mesma forma que no caso da árvore B, é importante otimizar o acesso ao disco.
- Assim, cada *posição* no índice, pode conter mais de uma entrada (ou registro)
- Exemplo:
  - Registro no índice = 12 bytes
  - Setor do HD = 4096 bytes = 341,33 registros

Buckets (cesto)



	Código	End.	Código	End.	Código	End.	Código	End.
0								
1								
2	1	0	5	432				
3								
4								
5	3	216	4	324				
6								
7								
8	2	108						
9								

Buckets (cesto)

- Tratamento de colisões (quando o cesto está cheio)
  - Alocar o registro no próximo cesto em que houver espaço disponível (usando endereçamento aberto)
  - Usar uma das técnicas anteriores (considerando que cada posição equivale a um novo cesto)

Tabelas *hash* dinâmicas

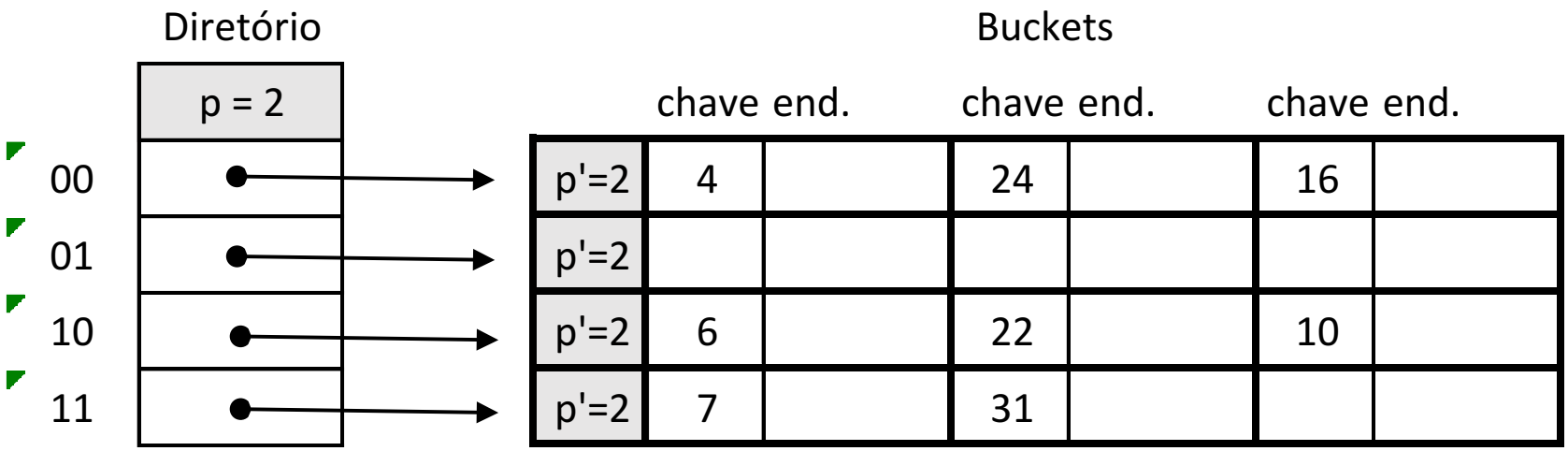
Tabela *hash* dinâmica

- Quando o arquivo de dados cresce ou diminui com frequência (muitas inclusões e exclusões), o índice também precisará ser ajustado.
- Uma tabela *hash* estática, para crescer, precisa reposicionar todos os registros.

Tabela *hash* dinâmica

- Uma tabela *hash* dinâmica é uma tabela *hash* em que apenas alguns registros afetados (aqueles do *bucket*) precisam ser reposicionados.

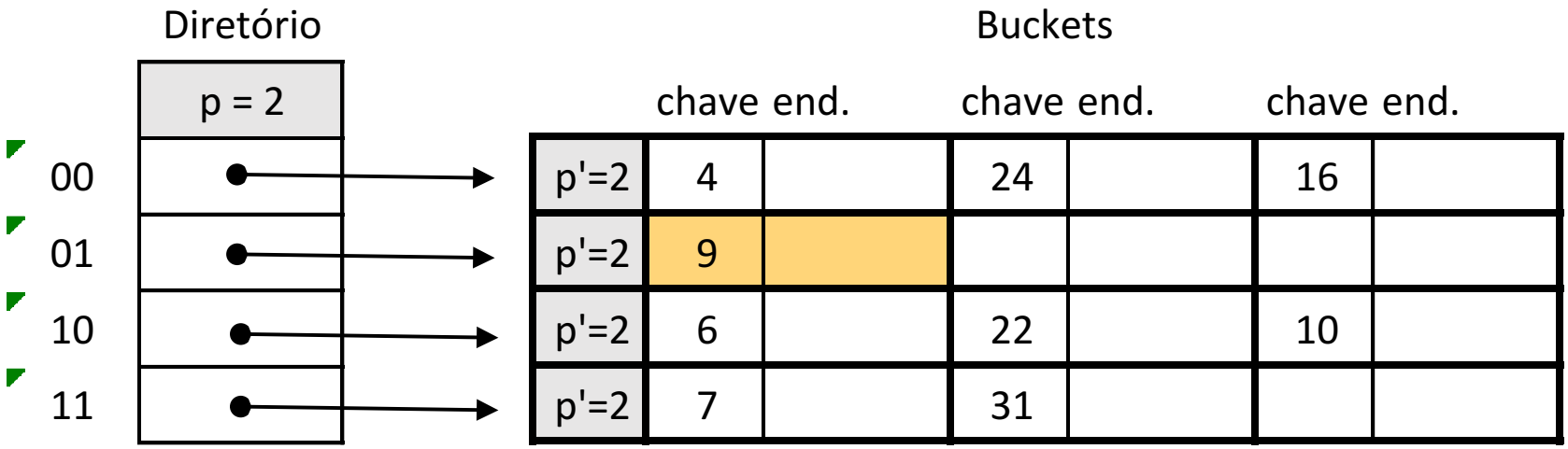
Hash extensível



$h(k) = k \; mod \; 2^p$

Hash extensível

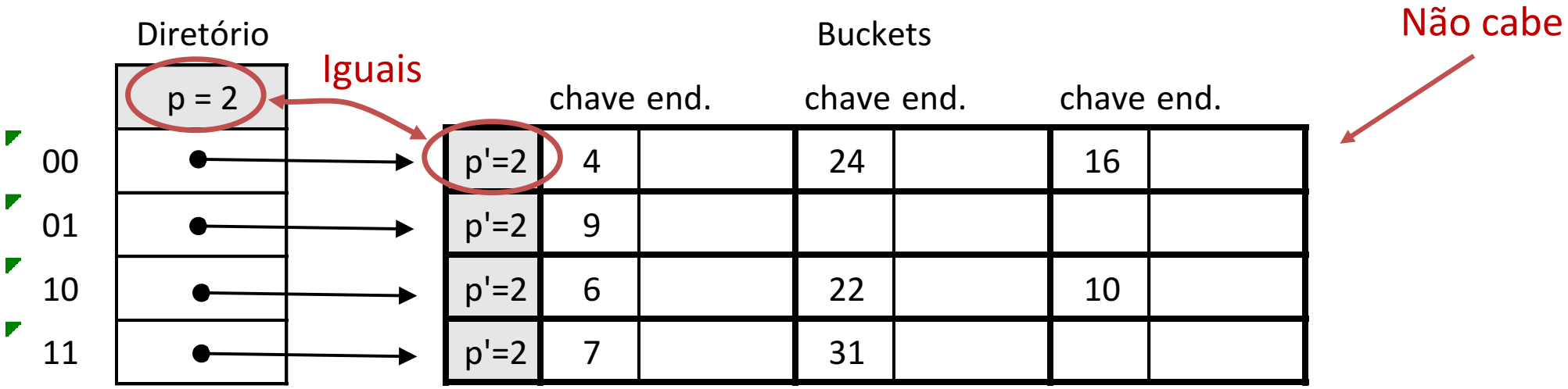
Adicionar chave 9:



$h(k) = k \; mod \; 2^p$

Hash extensível

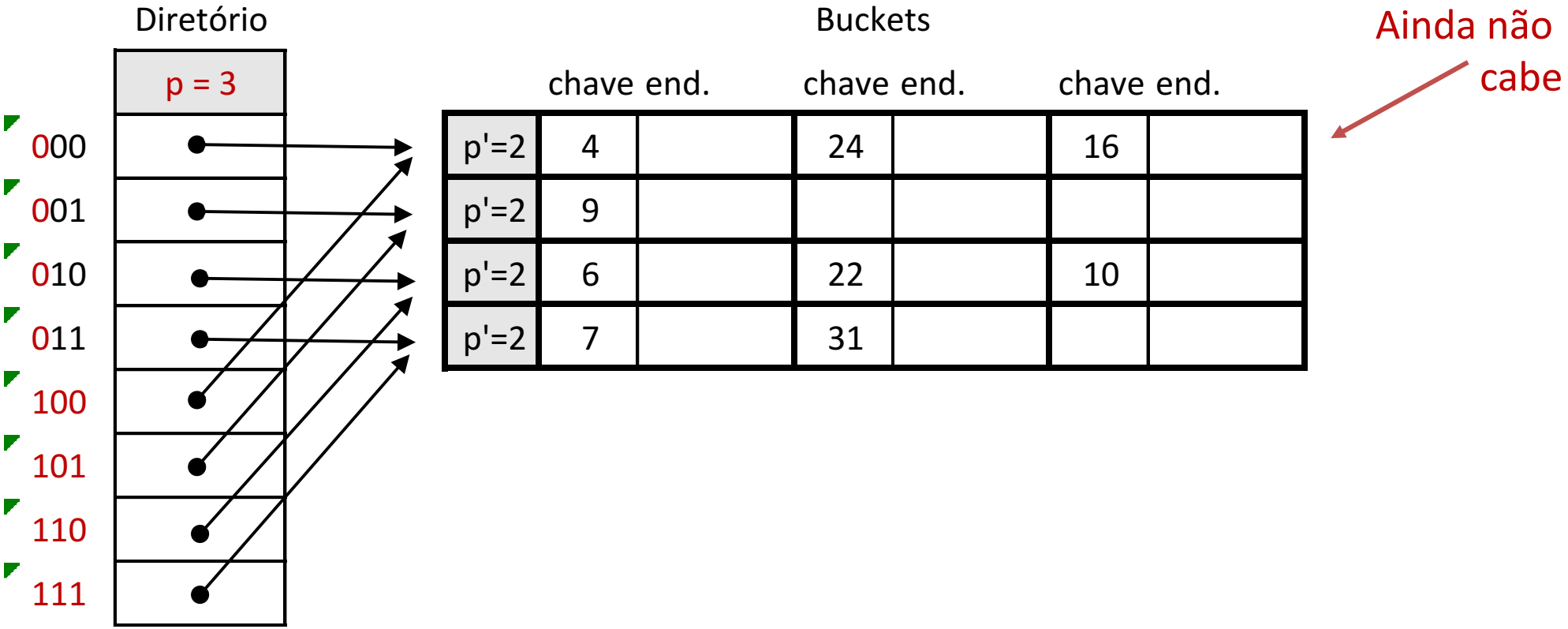
Adicionar chave 20:



$h(k) = k \; mod \; 2^p$

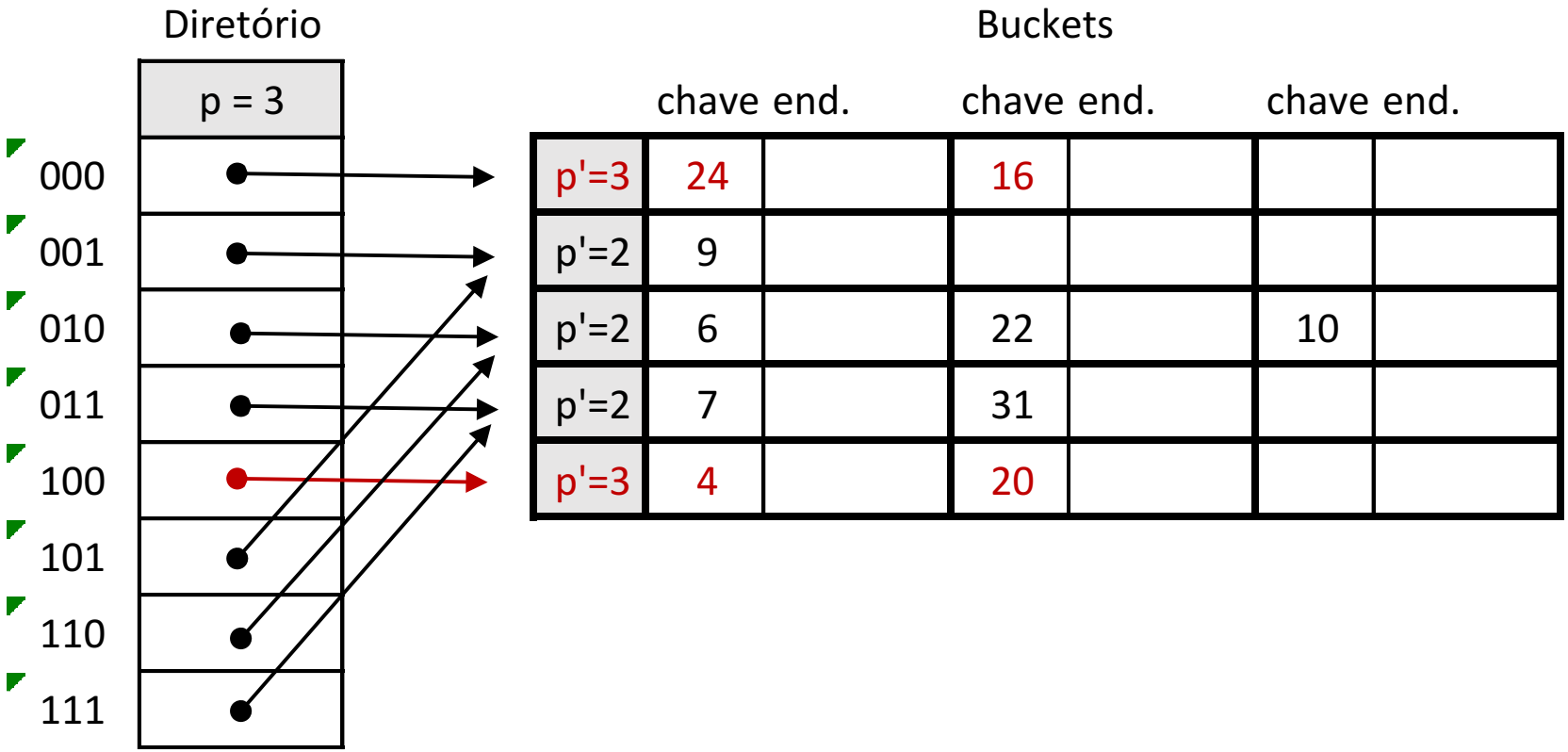
Hash extensível

Adicionar chave 20:



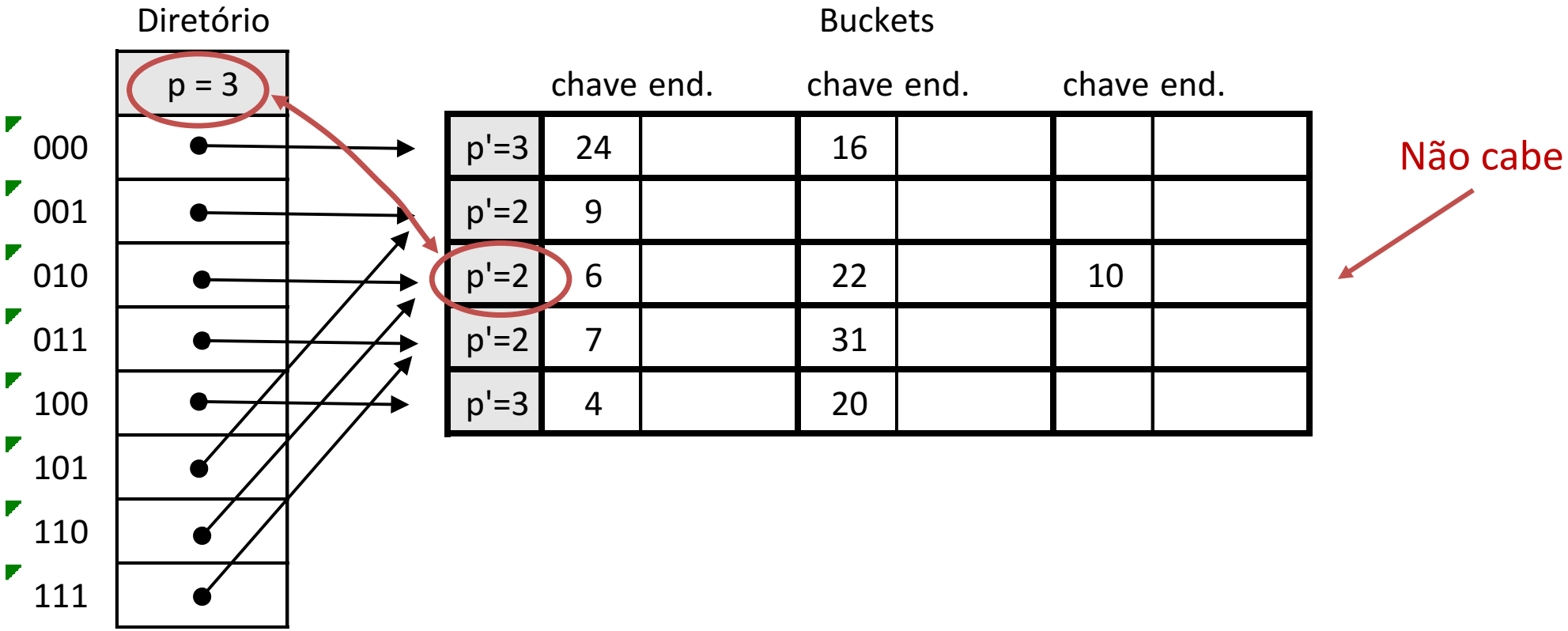
Hash extensível

Adicionar chave 20:



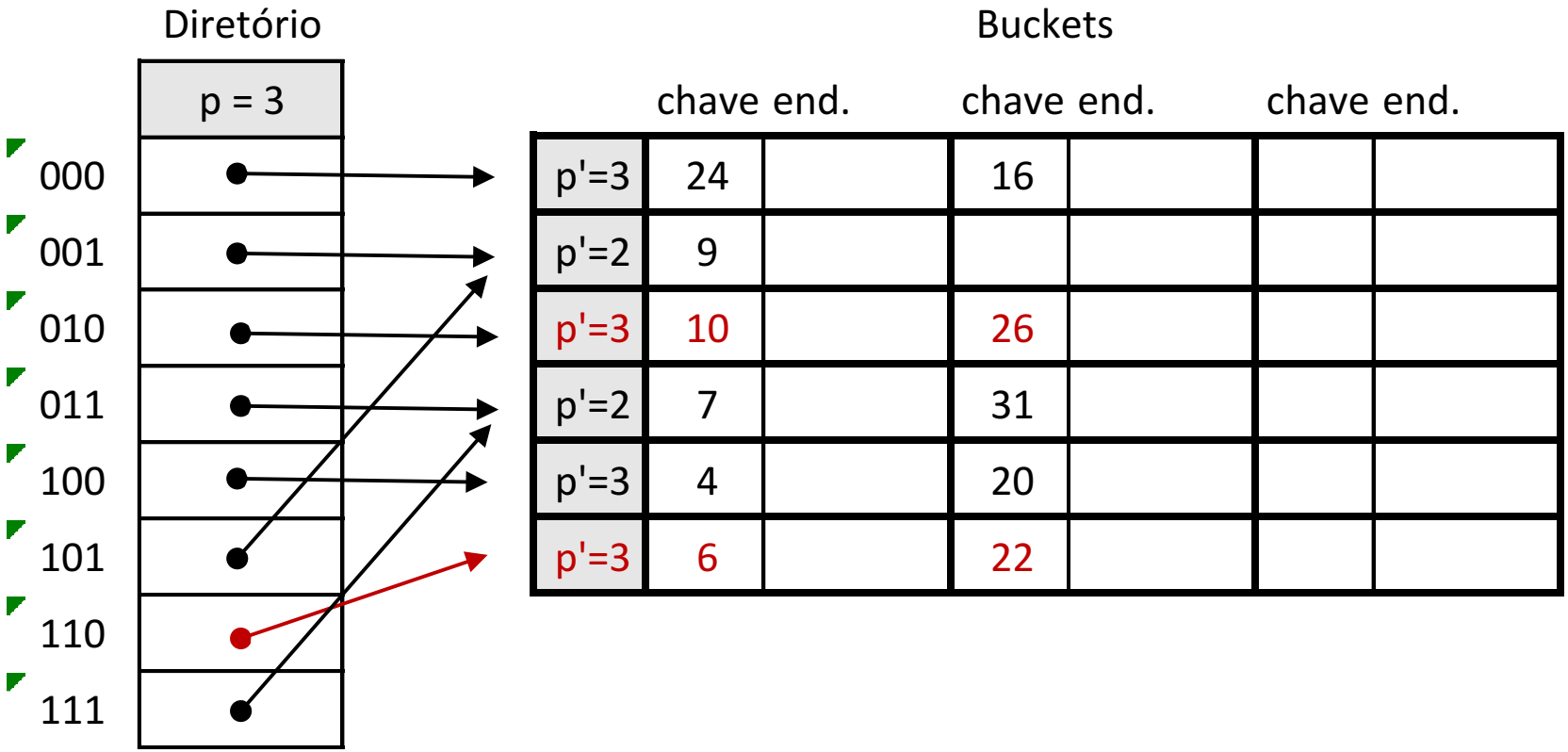
Hash extensível

Adicionar chave 26:



Hash extensível

Adicionar chave 26:



Vantagens do *hash* extensível

- O diretório cresce, sem precisarmos reposicionar todos os registros (do índice)
- O índice (lista de *buckets*) cresce de acordo com a necessidade
- Como não há encadeamento dos *buckets*, não há perda de eficiência