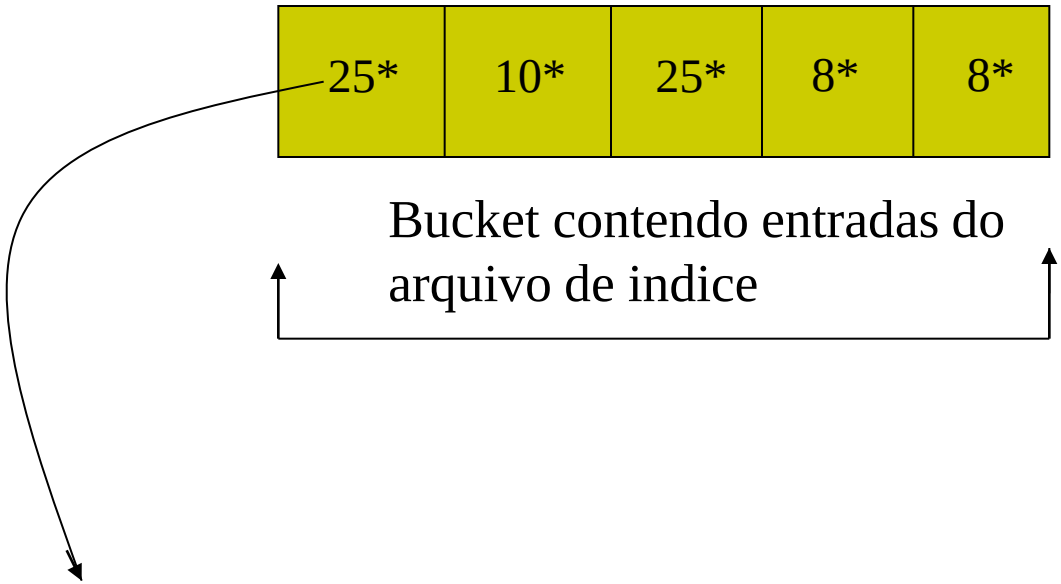
Métodos de Acesso

Hash Extensível x Hash Linear

Hash Extensível

- **Solução 1** : quando algum bucket ficar cheio
  - Dobrar o número de buckets
  - Distribuir as entradas nos novos buckets
  - Defeito : o arquivo todo deve ser lido e reorganizado e o dobro de páginas devem ser escritas.
- **Solução 2** : utilizar um diretório de ponteiros para os buckets.
  - Dobrar o número de entradas no diretório
  - Separar **somente** os buckets que ficaram cheios.

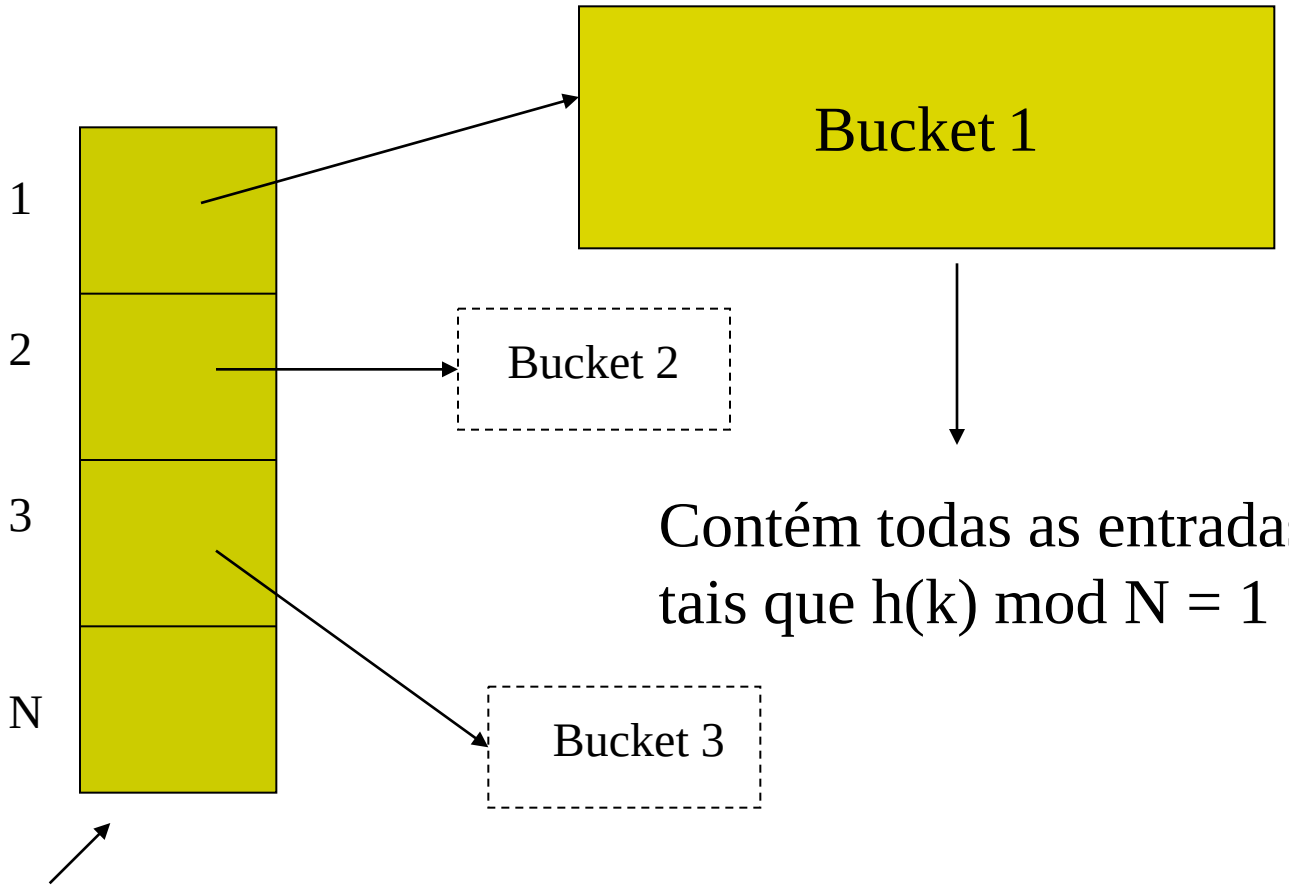
Convenção de Notação



Representa a entrada <k,*>, onde h(k) = 25

h = função hash **fixa**

Diretorio de buckets

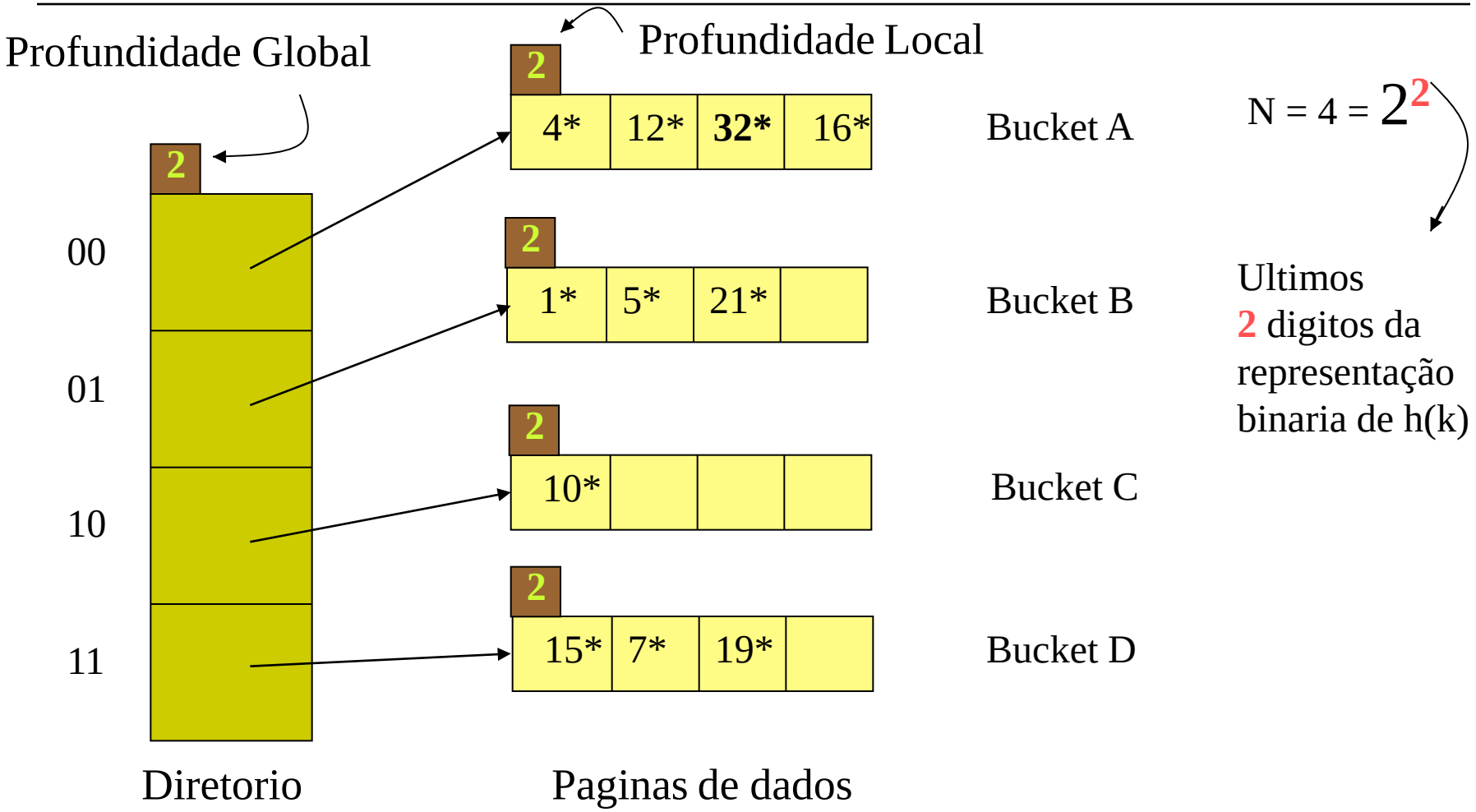


Diretorio : so armazena ponteiros para os buckets
numero de registros = numero de buckets

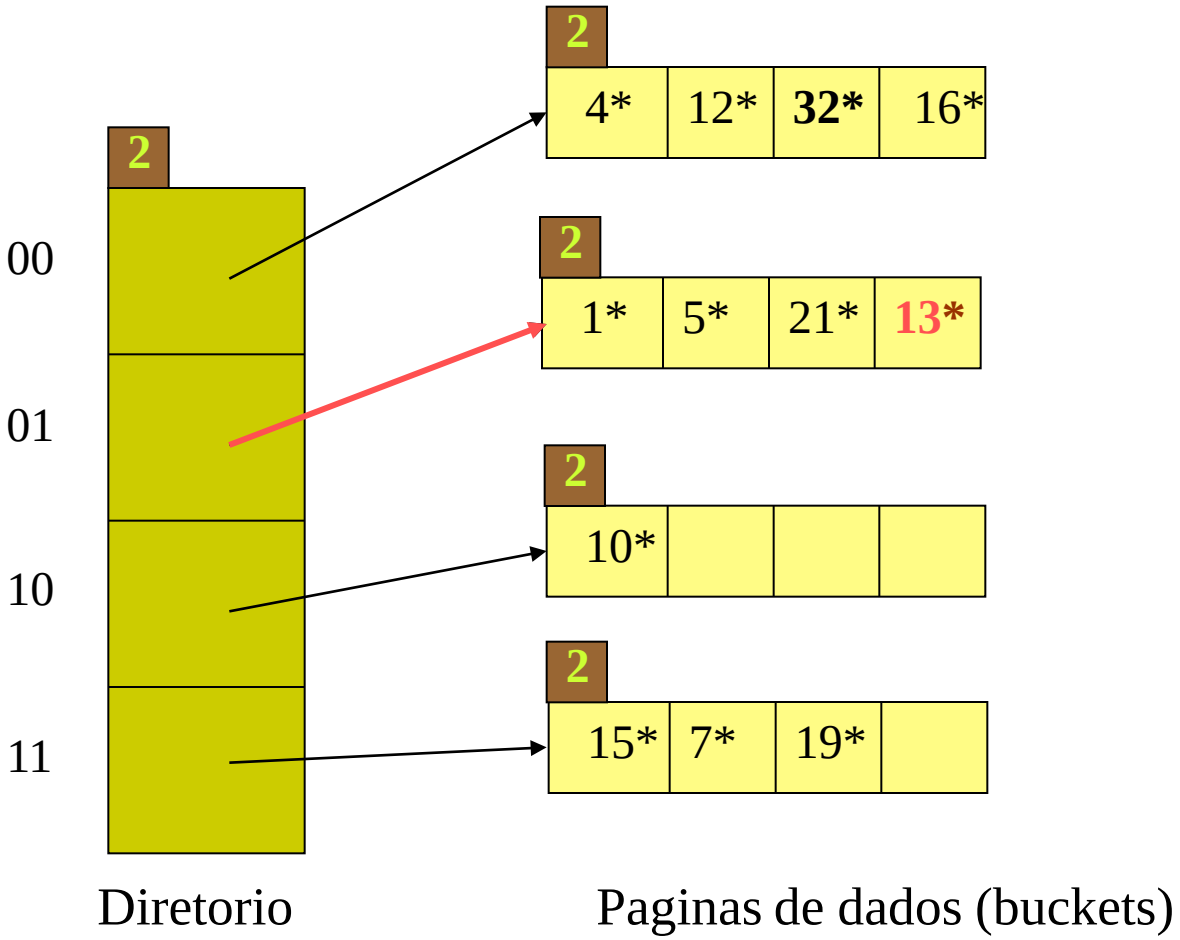
O que varia ?

- Função hash não varia
- O numero N de buckets varia
- A medida que os buckets se enchem, estes se duplicam, e o diretorio de buckets duplica
- Resultado :
  - se um único bucket duplica, o diretório todo de buckets duplica
  - Dois ponteiros do diretório podem apontar para o mesmo bucket
  - So duplicam os buckets que ficam cheios.
  - Ao contrário do hash estático, registros em buckets duplicados (decorrentes de um overflow) podem ser facilmente localizados através do novo ponteiro no diretório de buckets.

Exemplo

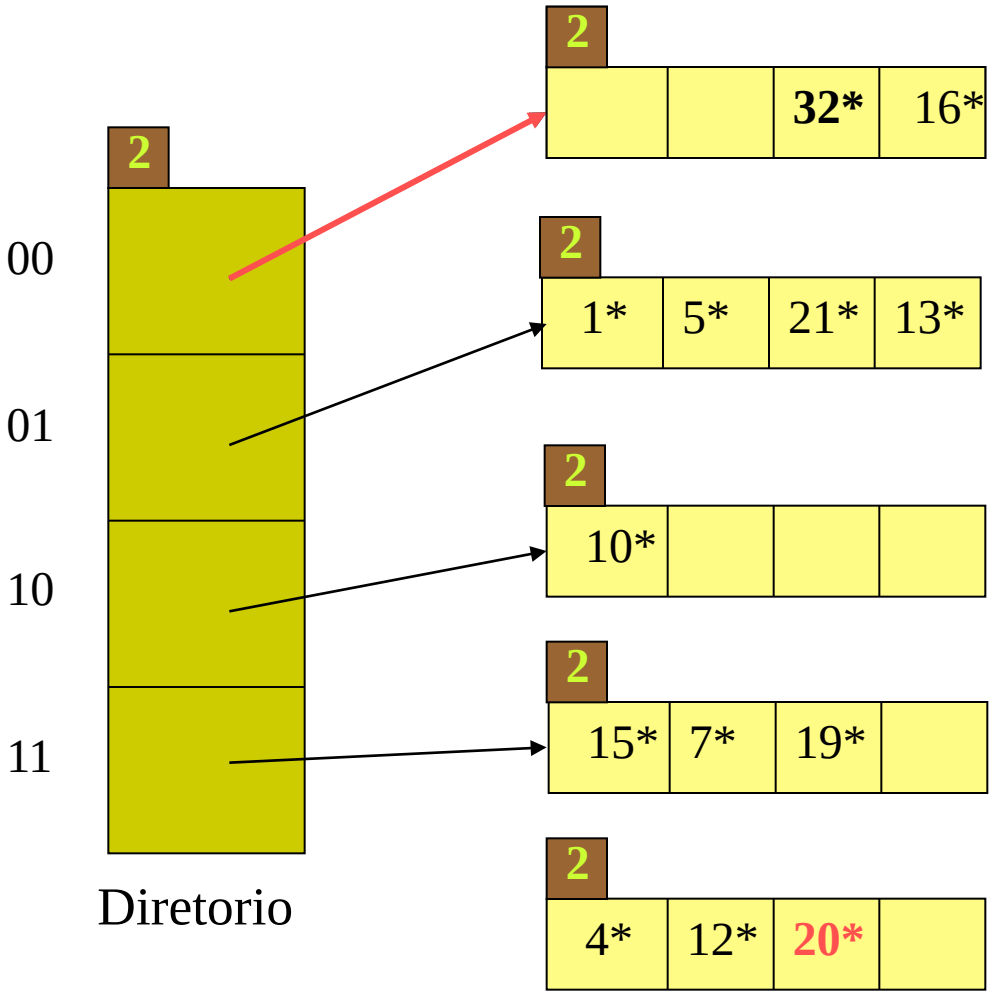

Profundidade Global
Profundidade Local
2
00
01
10
11
4*
12*
32*
16*
Bucket A
1*
5*
21*
Bucket B
10*
Bucket C
15*
7*
19*
Bucket D
N = 4 = 2^2
Ultimos
2 digitos da
representação
binaria de h(k)
Diretorio
Paginas de dados

Exemplo – inserção



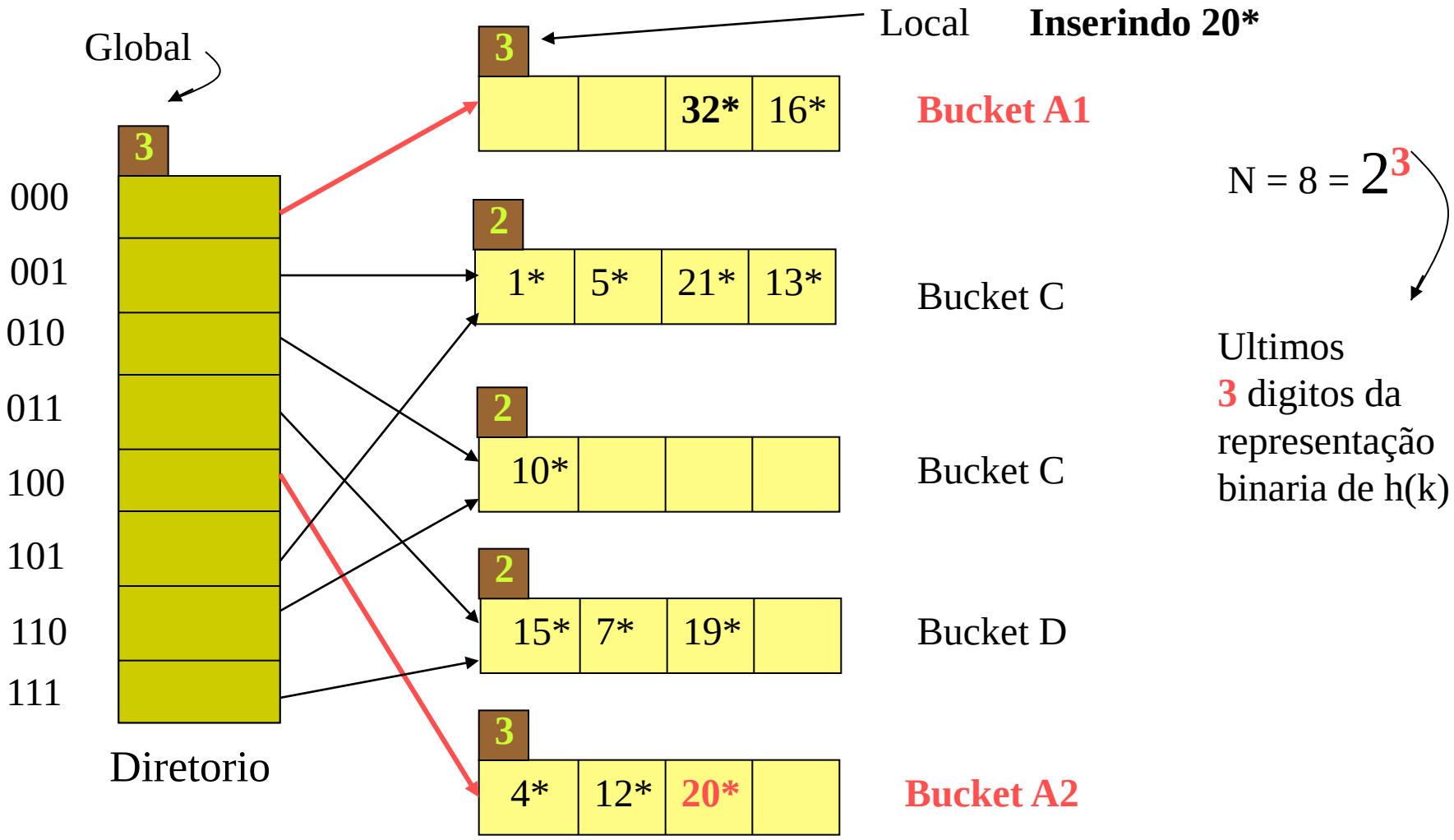
Inserindo 13*

Exemplo – inserção

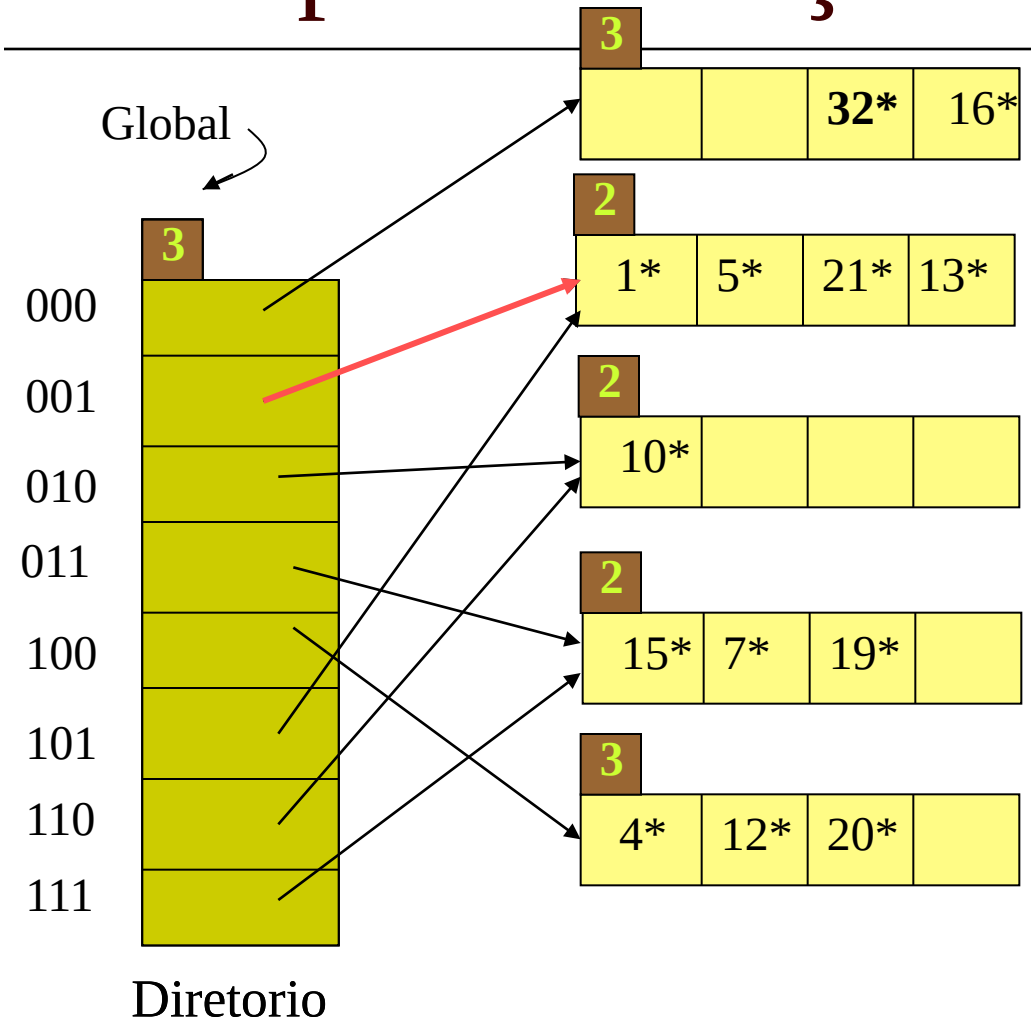


Inserindo 20*

Exemplo – inserção

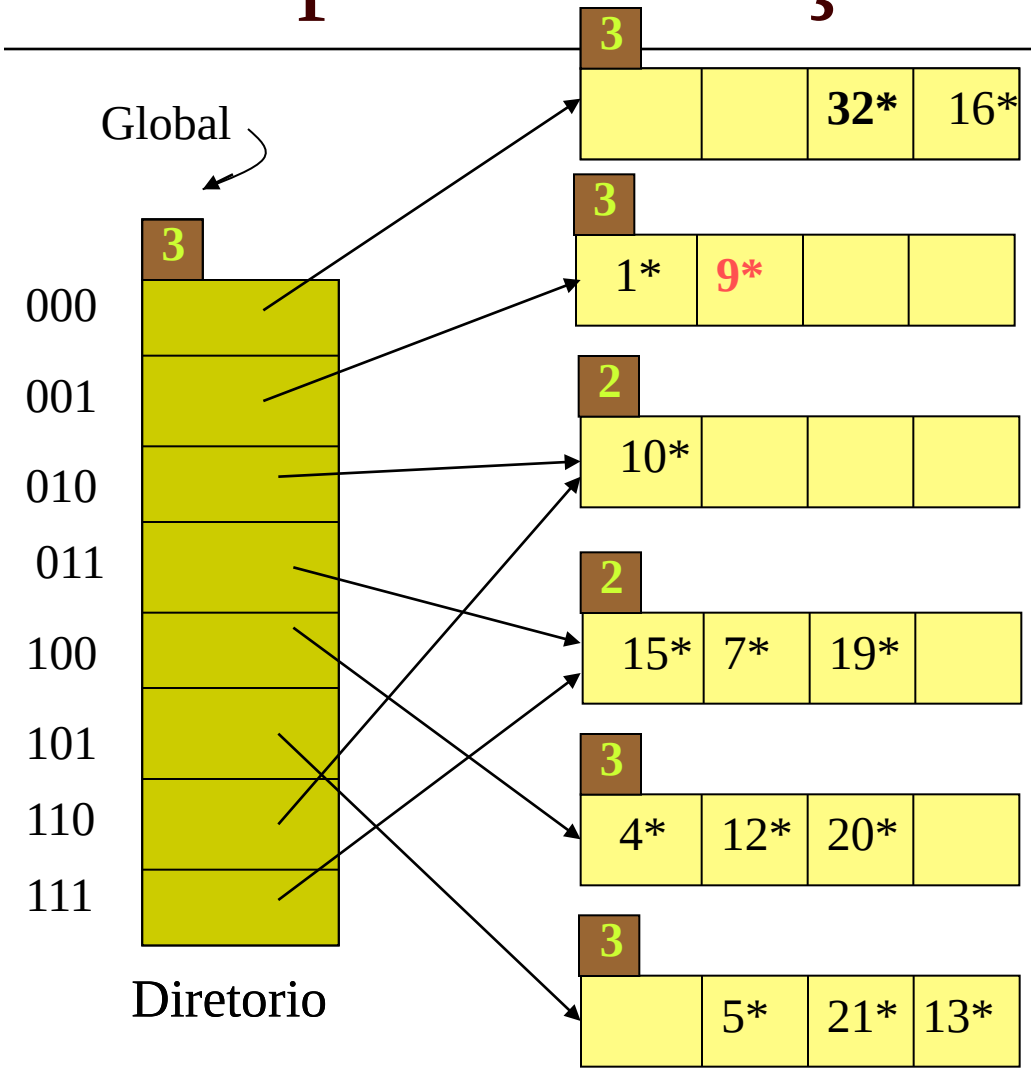


Exemplo – inserção



Inserindo 9*

Exemplo – inserção



Inserindo 9*

Regra geral para inserção de k*

Se Nivel global = **d**

- Calcula h(k)
- Considera a entrada m do diretório, onde

  m = número correspondente aos **d** últimos dígitos da representação binária de h(k)
- Dirige-se para o bucket indicado
- Se o bucket estiver cheio e **nível local = d**
  - Divide o bucket e duplica o diretório de buckets
- Se o bucket estiver cheio e **nível local < d**
  - Divide o bucket, mas **não** duplica diretório

Análise

- Se o diretório couber na memória
  - Seleção com igualdade : **1** I/0
- Se o diretório tiver que ser armazenado em disco
  - Seleção com igualdade : **2** I/0 (Por que ?)
- Comparação com hash estático:
  - Sem overflow : hash estático tem custo 1 I/0
  - Com overflow : hash estático tem custo dependente do numero de páginas de overflow
  - Hash extensível : no máximo 2 I/0

Possiveis problemas

- Distribuição tendenciosa dos valores h(k) : muitos num único bucket !
  - Este é um problema que pode ser resolvido no momento da criação do índice : basta ajustar a função h de modo a ter uma distribuição uniforme.

- Colisão : quando existem muitas entradas <k,*> com mesmo h(k), que não cabem numa página
  - Este é um problema que só aparece à medida que o arquivo cresce.
  - Neste caso, páginas de overflow são utilizadas.

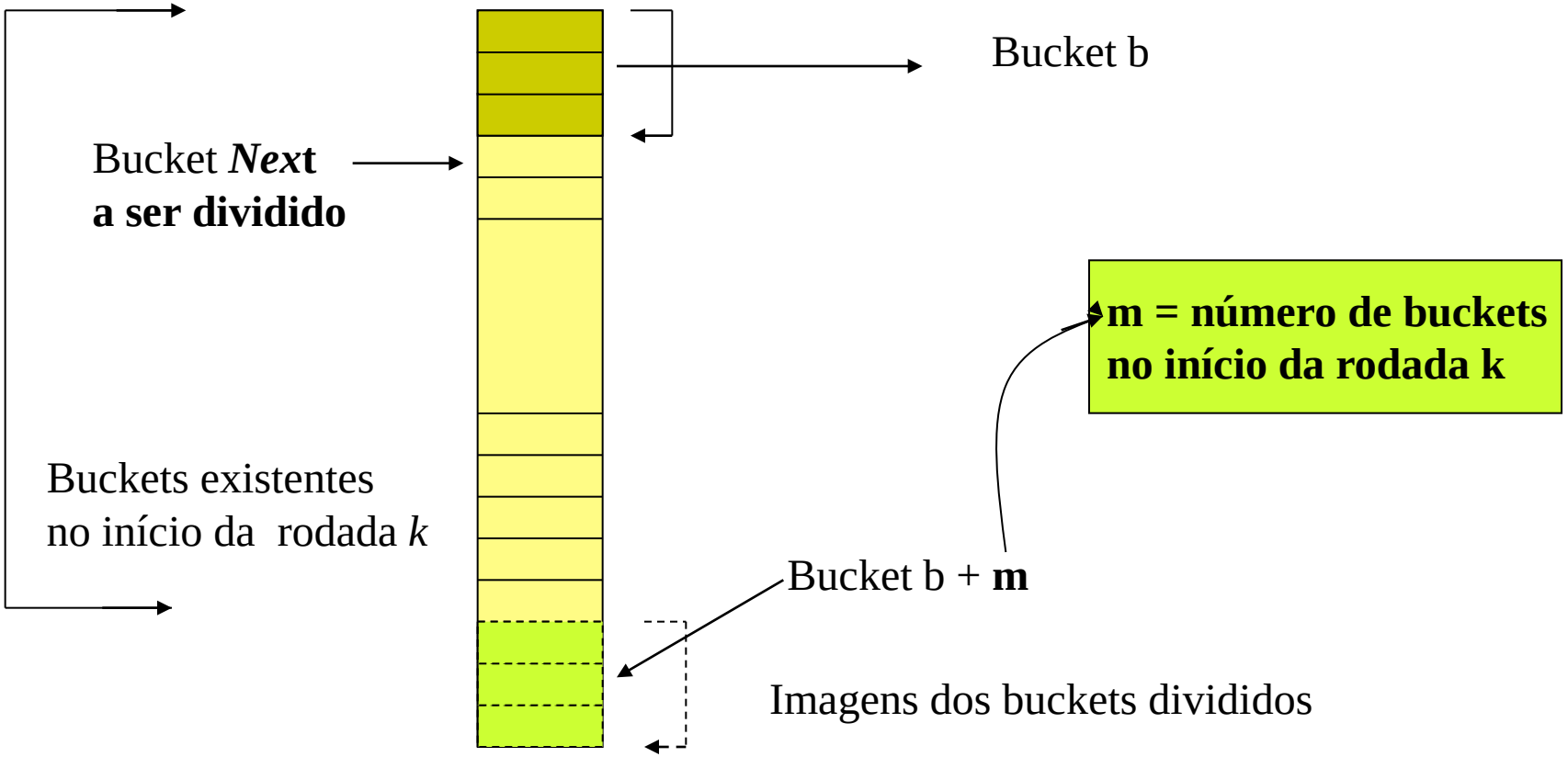
Hash Linear

- Função hash varia
- Não ha diretório de buckets
- Pode haver páginas de overflow à medida que os buckets se enchem
- Regularmente, função hash se modifica e páginas de overflow são realocadas.

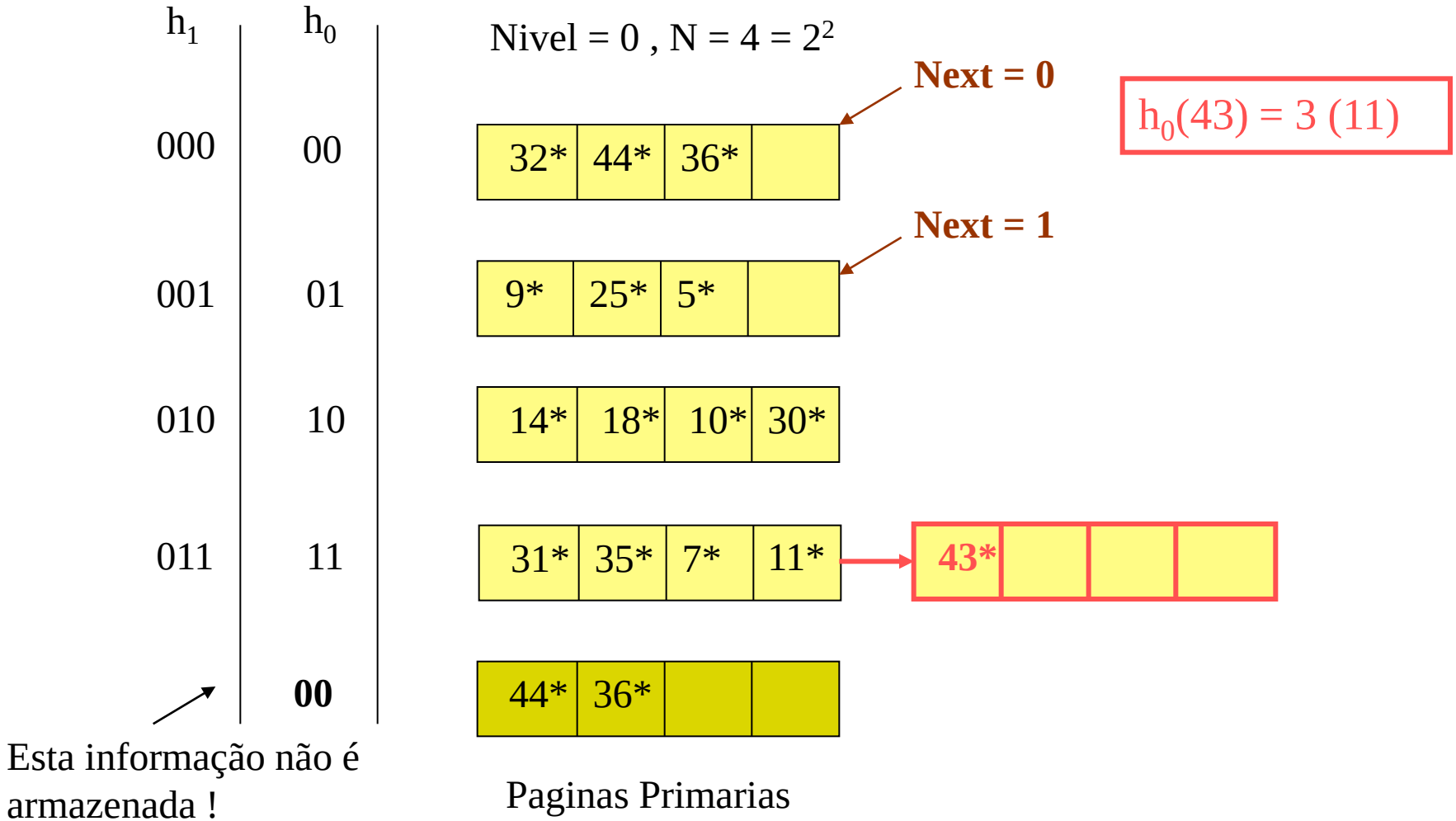
Parâmetros e Contadores

- *Nivel* = indica a rodada atual
  - Inicializado com 0
- *Next* = bucket que deve ser dividido, se necessario
  - Inicializado com 0
- N_m = número de buckets na rodada m
  - $N_0 = N$
  - $N_m = N*2^m$
- Somente o bucket com número *Next* é dividido.
  - Usa-se páginas de overflow para os outros buckets, se ficarem cheios.
  - Apos divisão, *Next* é incrementado

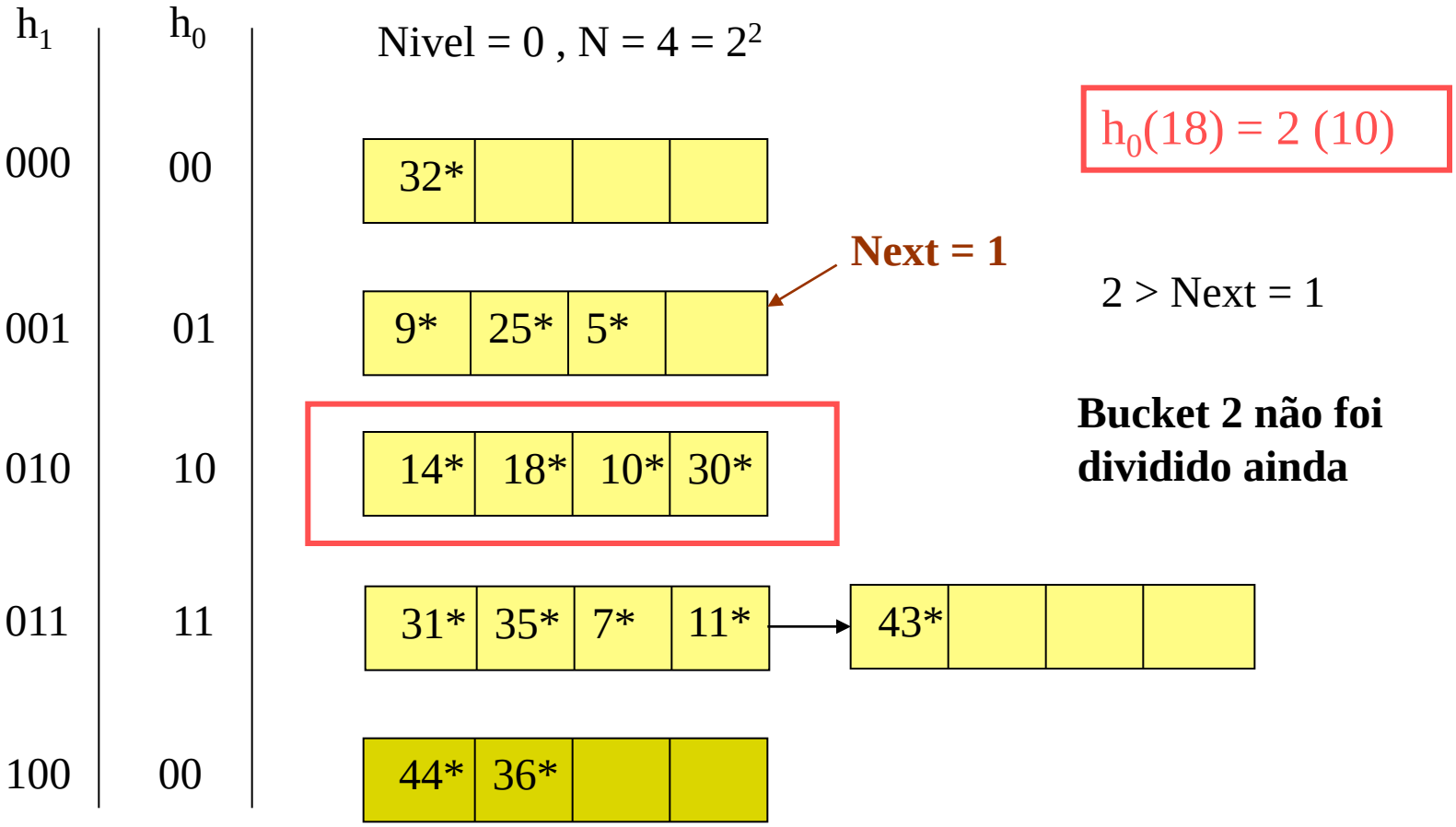
Esquema Geral : rodada *k*



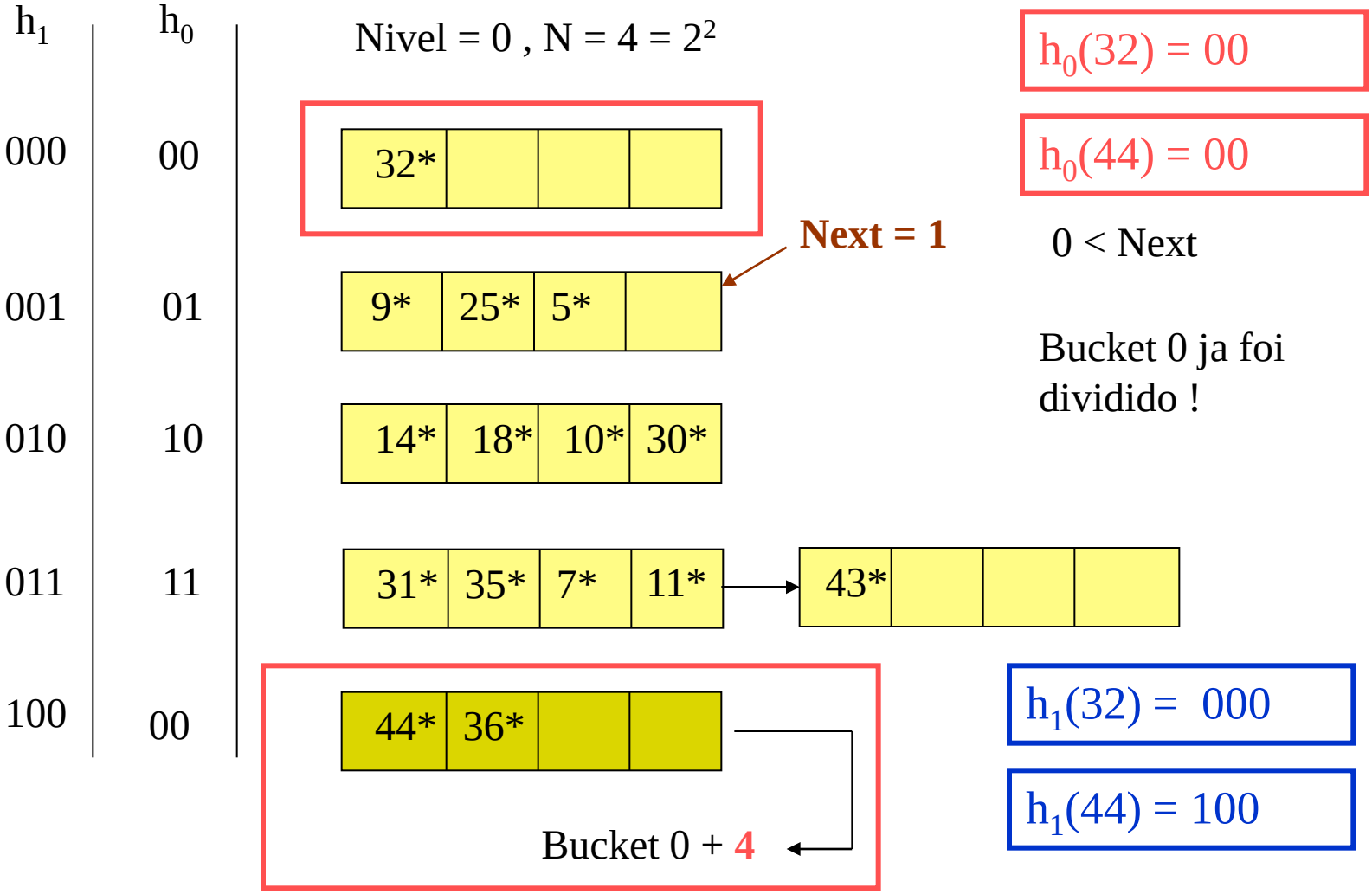
Inserção de 43*



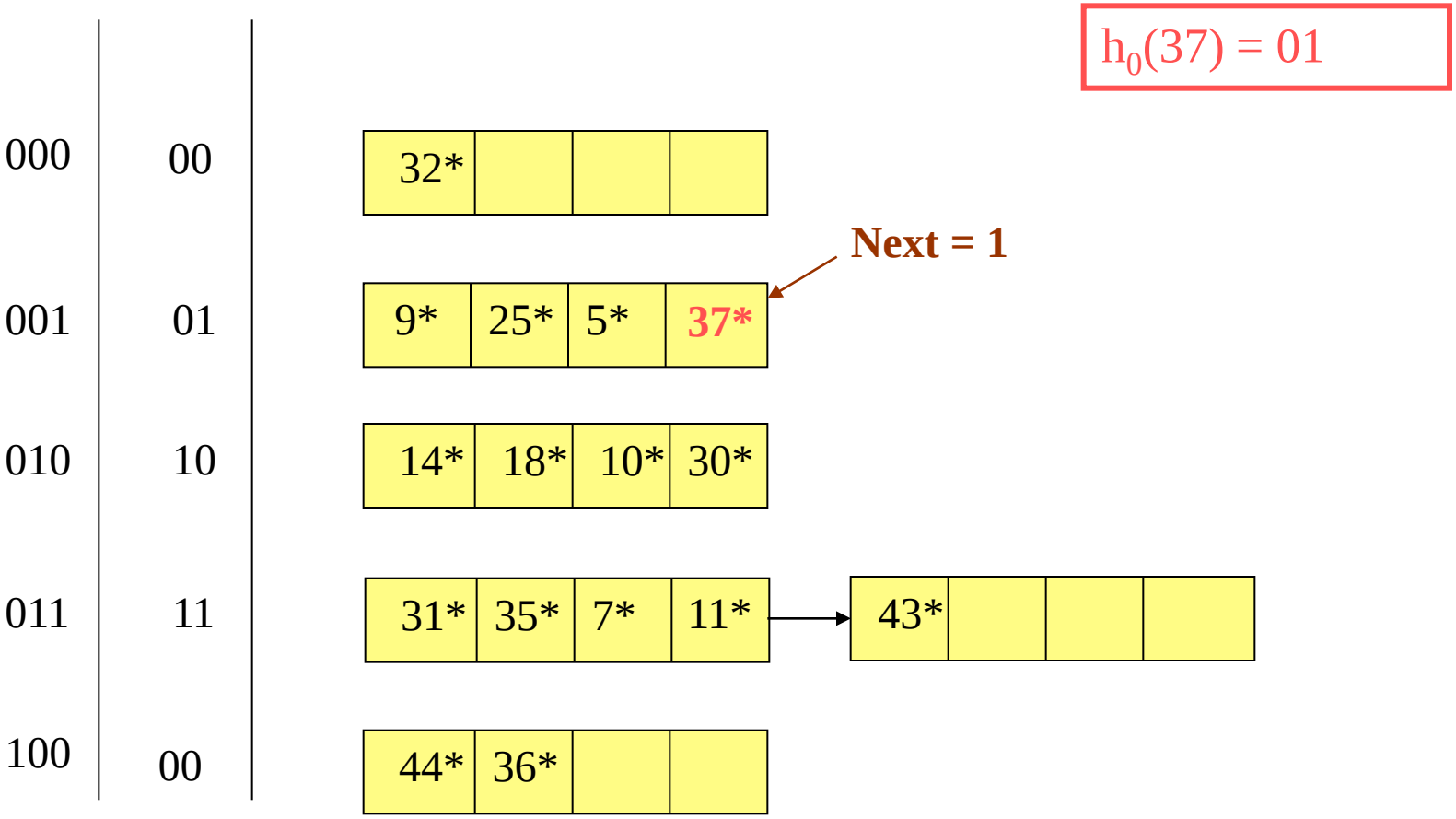
Busca de 18*


h1
h0
Nivel = 0 , N = 4 = 2^2
h0(18) = 2 (10)
000
00
32*
Next = 1
001
01
9*
25*
5*
2 > Next = 1
010
10
14*
18*
10*
30*
Bucket 2 não foi dividido ainda
011
11
31*
35*
7*
11*
43*
100
00
44*
36*

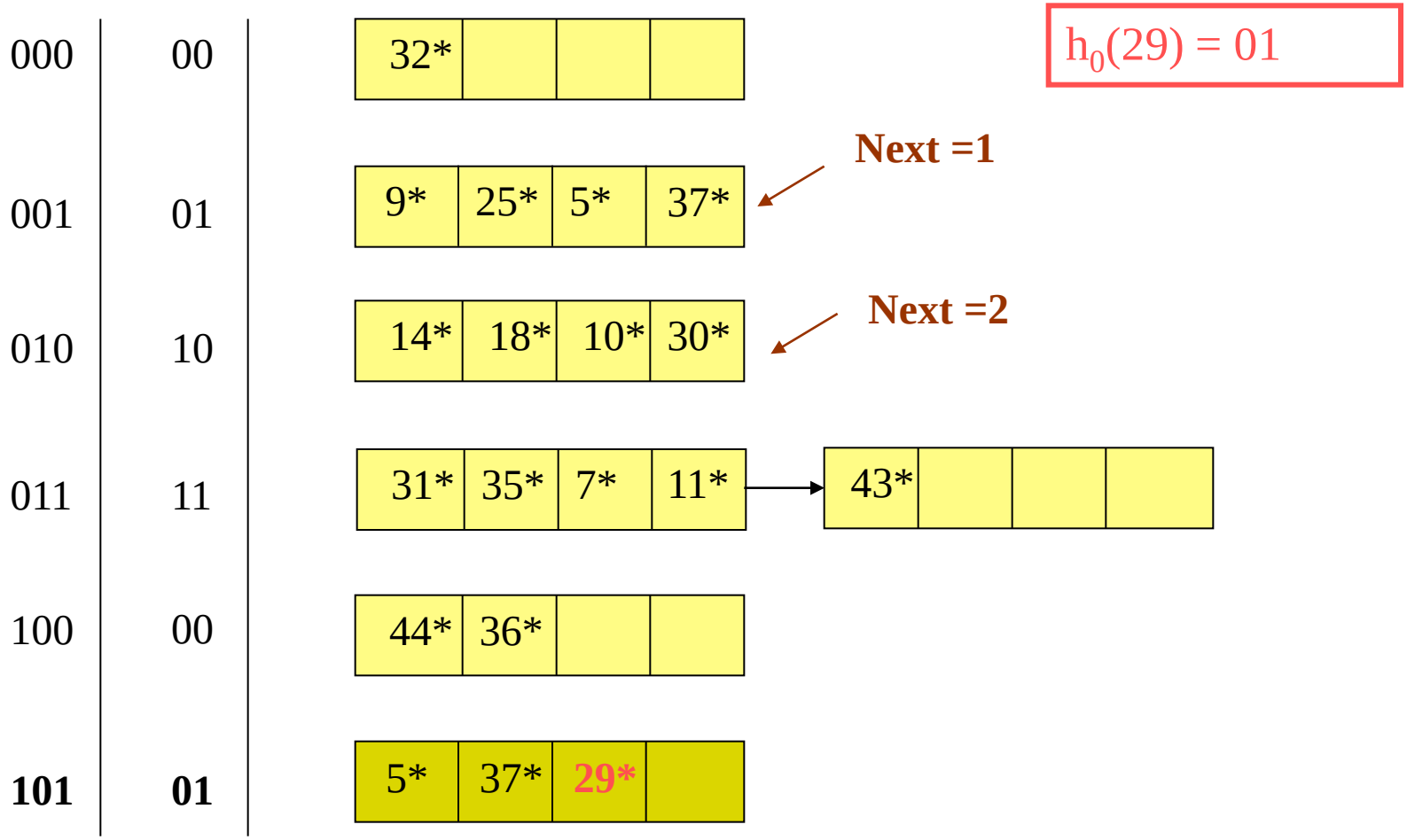
Busca de 32* e 44*



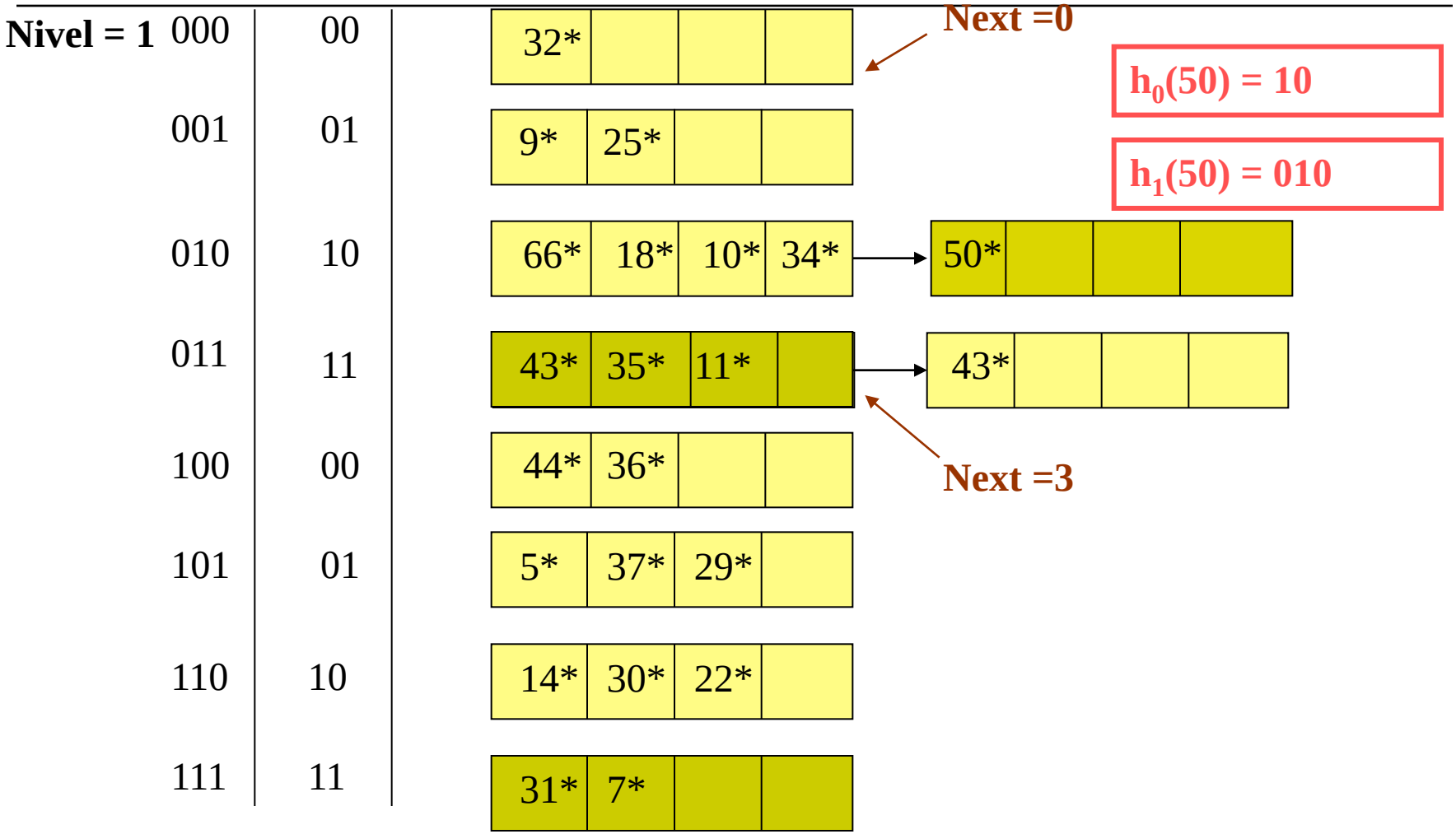
Inserção de 37*



Inserção de 29*



Incremento de Nível



Duplicar diretorio x duplicar buckets

- Passagem de h_i para h_{i+1} no Hash Linear corresponde a duplicar o diretorio no Hash Extensível
- Hash Extensível : Diretorio é duplicado num único passo
- Hash Linear : duplicação do número de buckets se faz gradualmente

Análise

- Hash Linear não precisa de diretório :
  - existe uma maneira precisa de se saber quais buckets foram divididos e quais devem ser divididos e em que circuntâncias
  - Buckets são alocados consecutivamente: é possível localizar a página do bucket *m* por um simples cálculo de offsets.
- Hash Extensível tem uma melhor ocupação dos buckets : só se divide o bucket apropriado.

Análise :

- Hash Linear tem um custo I/O menor do que Hash Extensivel, para seleções com igualdade, em caso de distribuição uniforme
  - Distribuição uniforme : os valores possíveis de h estão uniformemente distribuídos nos N_0 pacotes iniciais), **sem páginas de overflow**.
  - Não se carrega páginas do diretório.
- Hash Linear tem um custo maior para seleções com igualdade, em caso de distribuição não uniforme
  - Páginas são alocadas para buckets quase vazios.
  - Páginas de overflow podem ser alocadas caso necessario.