Hashing

Dirk Van Gucht

Based on slides by Hector Garcia-Molina

Hashing: hash function

- Let K be a domain of key values (K can be very large)
- Let R = [0,m) (usually m << |K|) be a range of values
- A hash function h maps K to R

  h: K -> R

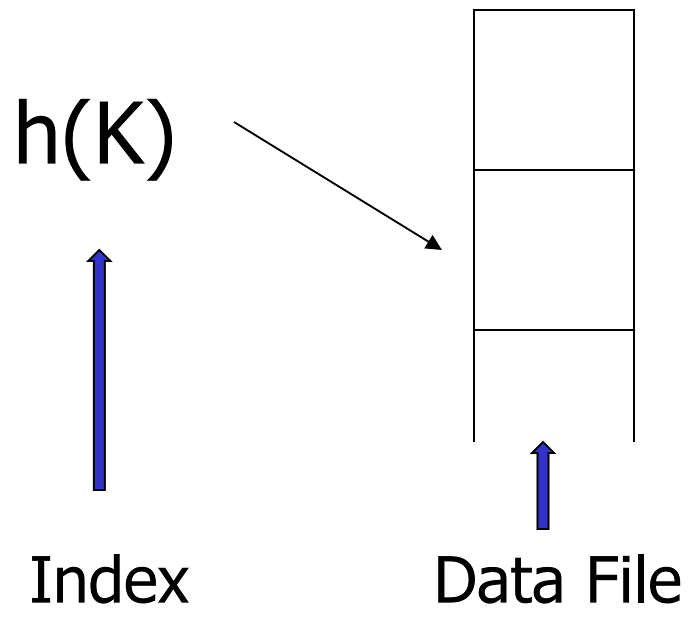
  Example: h(k) = k mod m
- Given a record r with key value k, h(k) provides an address (name) of a bucket in which to store r
- A bucket is stored in secondary memory as a block or a list of blocks
- Retrieving a bucket (and the records therein) can be done in O(n) where n is the number of blocks that store the bucket.

Hash function: collision

- Let r1 and r2 be two records with key values k1 and k2
- We permit that k1 = k2
- We say that h has a collision for r1 and r2 if h(k1) = h(k2)

  Consequence: r1 and r2 will be stored in the same bucket
- If K is the domain of a primary key, a collision will store records with different key values in same bucket (not desirable)
- If K is not the domain of a primary key, different records with the same key values will be placed in the same bucket (desirable). Partitioning.
- The latter property is exploited in key-value stores since records with the same key value will be sent to the same reducer

Next: example to illustrate inserts, overflows, deletes



EXAMPLE 2 records/bucket

(For simplicity, we identify a record with its key value)

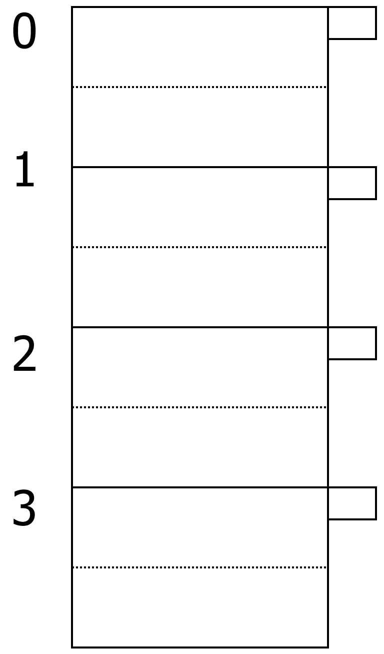
INSERT:

h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0



EXAMPLE 2 records/bucket

INSERT:

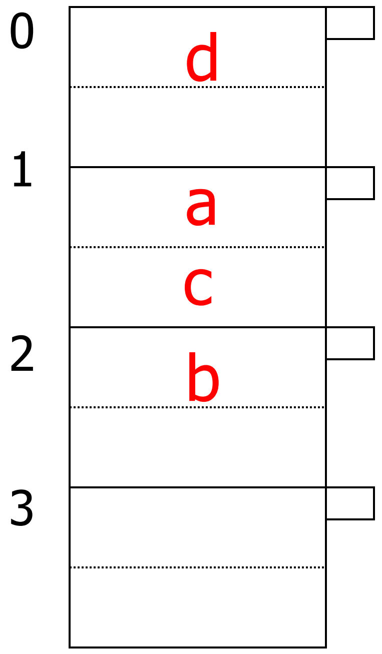
h(a) = 1

h(b) = 2

h(c) = 1

h(d) = 0

h(e) = 1



EXAMPLE 2 records/bucket

INSERT:

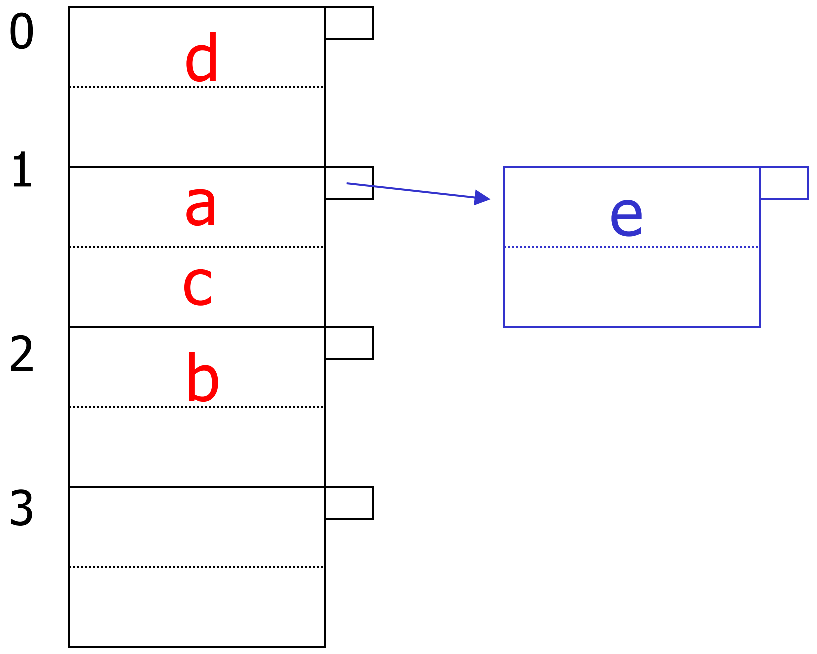
h(a) = 1

h(b) = 2

h(c) = 1

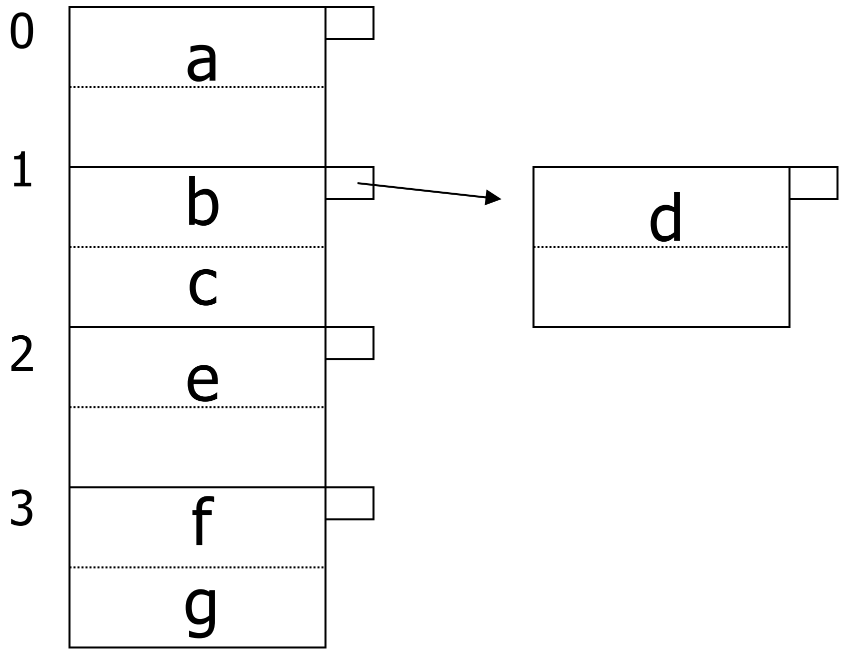
h(d) = 0

h(e) = 1



EXAMPLE: deletion

Delete:
- e
- f



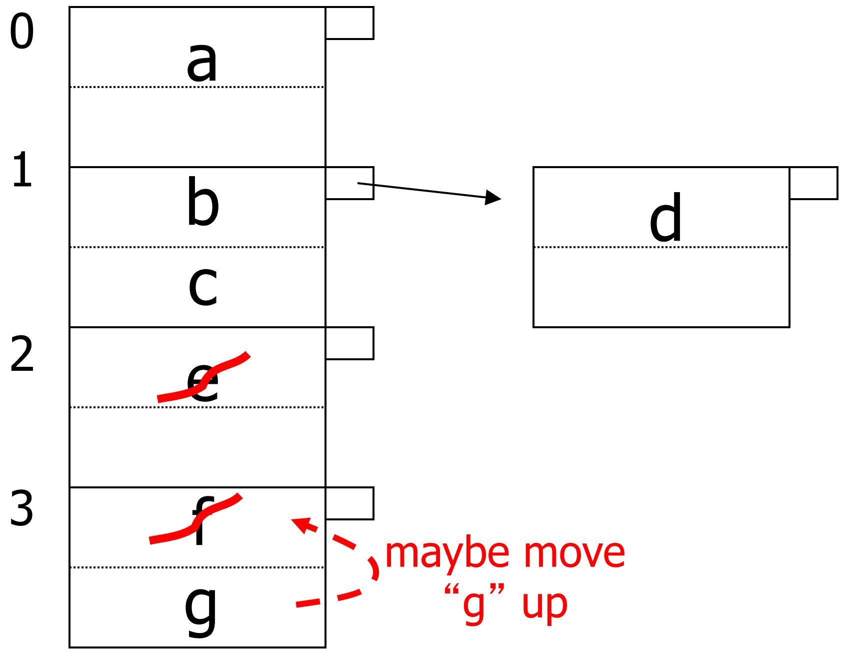
EXAMPLE: deletion

Delete:

e

f

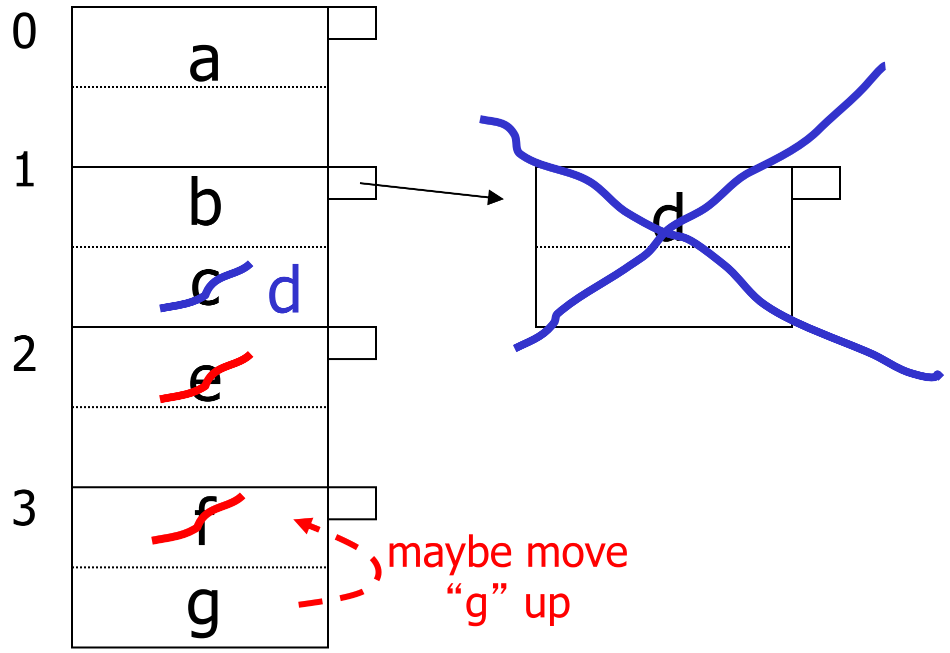
c



EXAMPLE: deletion

Delete:

- e
- f
- c



How do we cope with growth?

- Overflows and reorganizations
  Reorganization can be done by enlarging the range R and changing the hash function
  Reorganization requires complete rehashing and is linear in the |Data file|
- Dynamic hashing (extensible hashing)

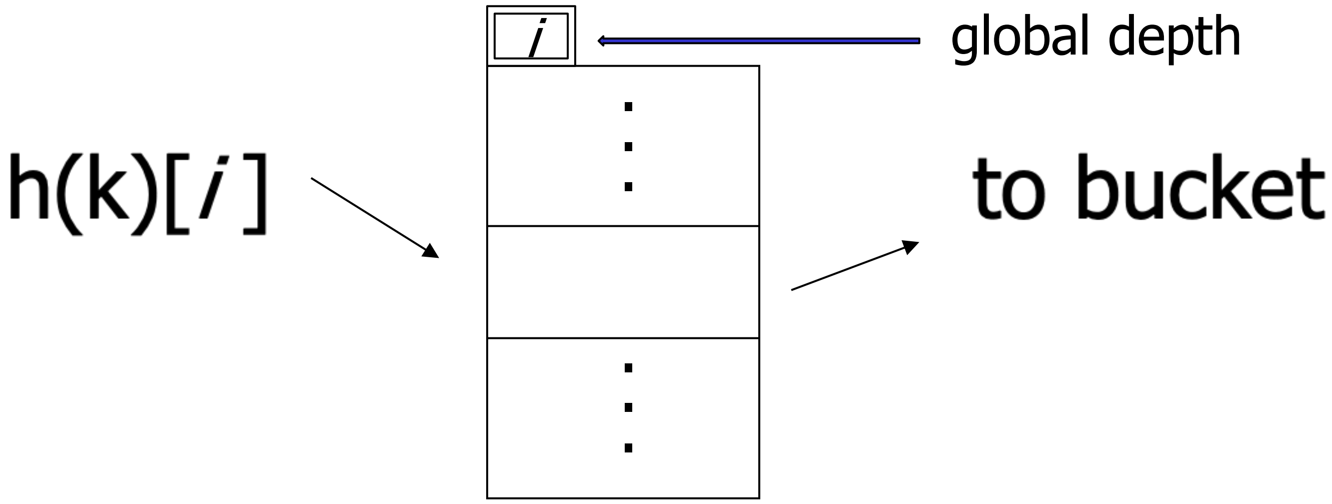
Extensible hashing: two ideas

(a) Use *i* of *b* bits output by hash function

$$h(k) \rightarrow \overset{\longleftarrow \quad b \quad \longrightarrow}{\boxed{\underbrace{001}_{}10101}}$$

use i $\rightarrow$ grows/shrinks over time....

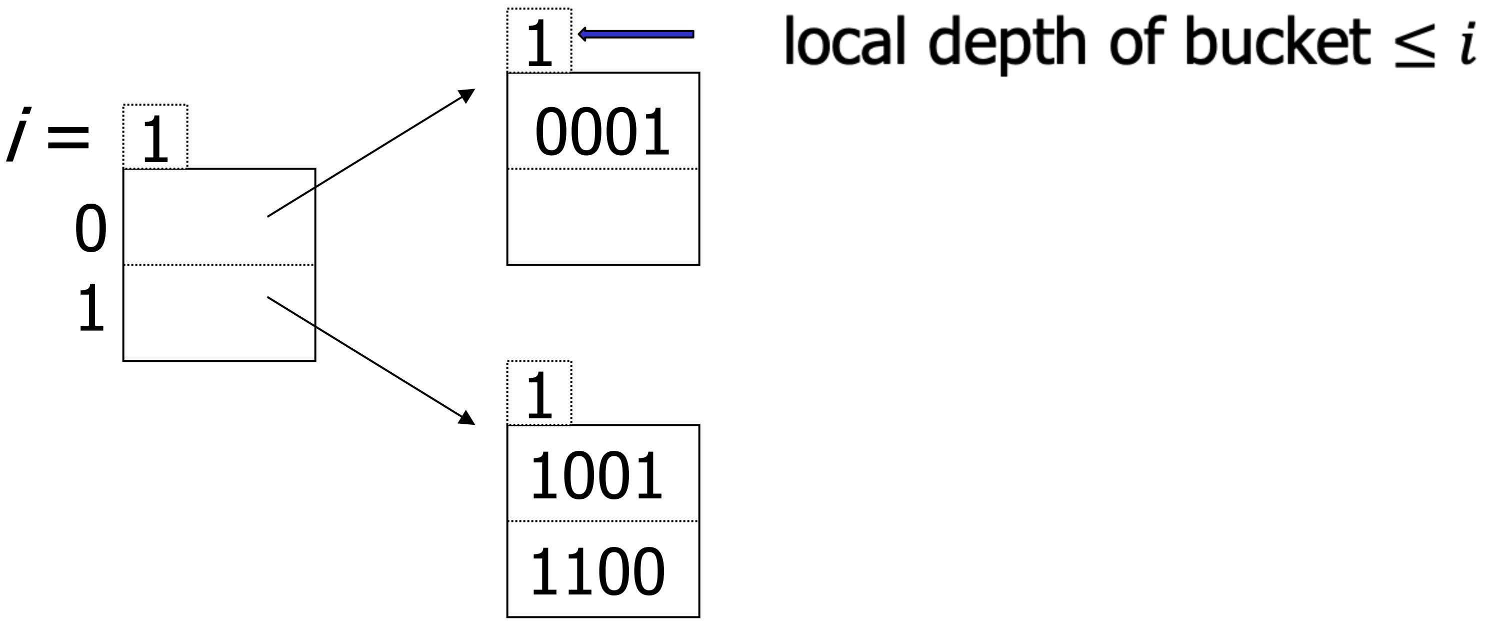
(b) Use directory of size 2^i



h(k) has b bits, but we will only look at its first i bits

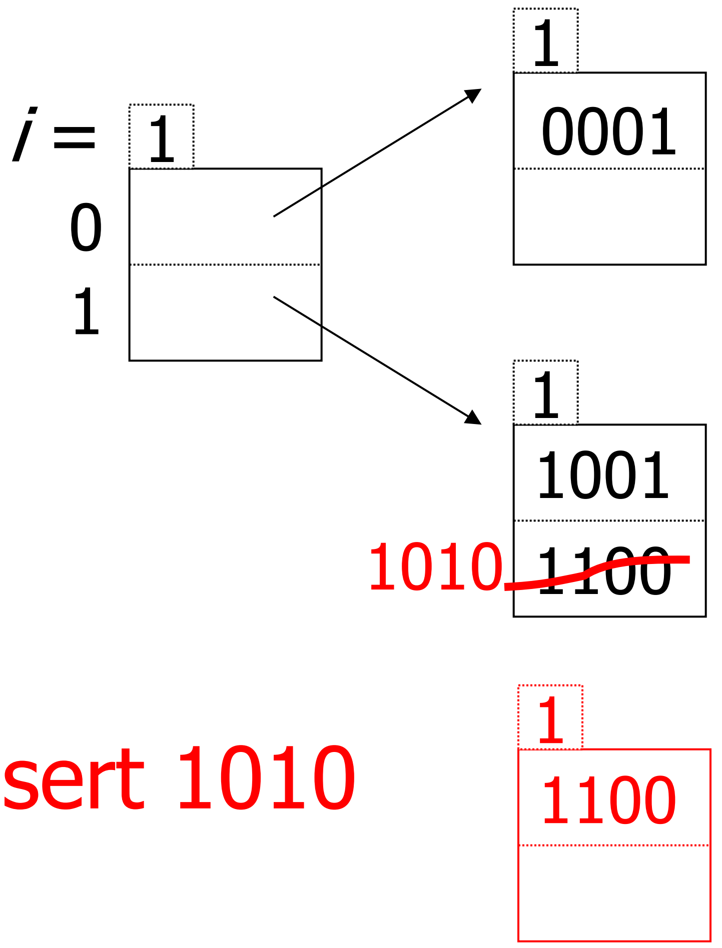
h(k)[*i*] consists of the first *i* bits of h(k)
these *i* bits specify the position for k the directory

Example: h(k) is 4 bits; 2 keys/bucket



Insert 1010

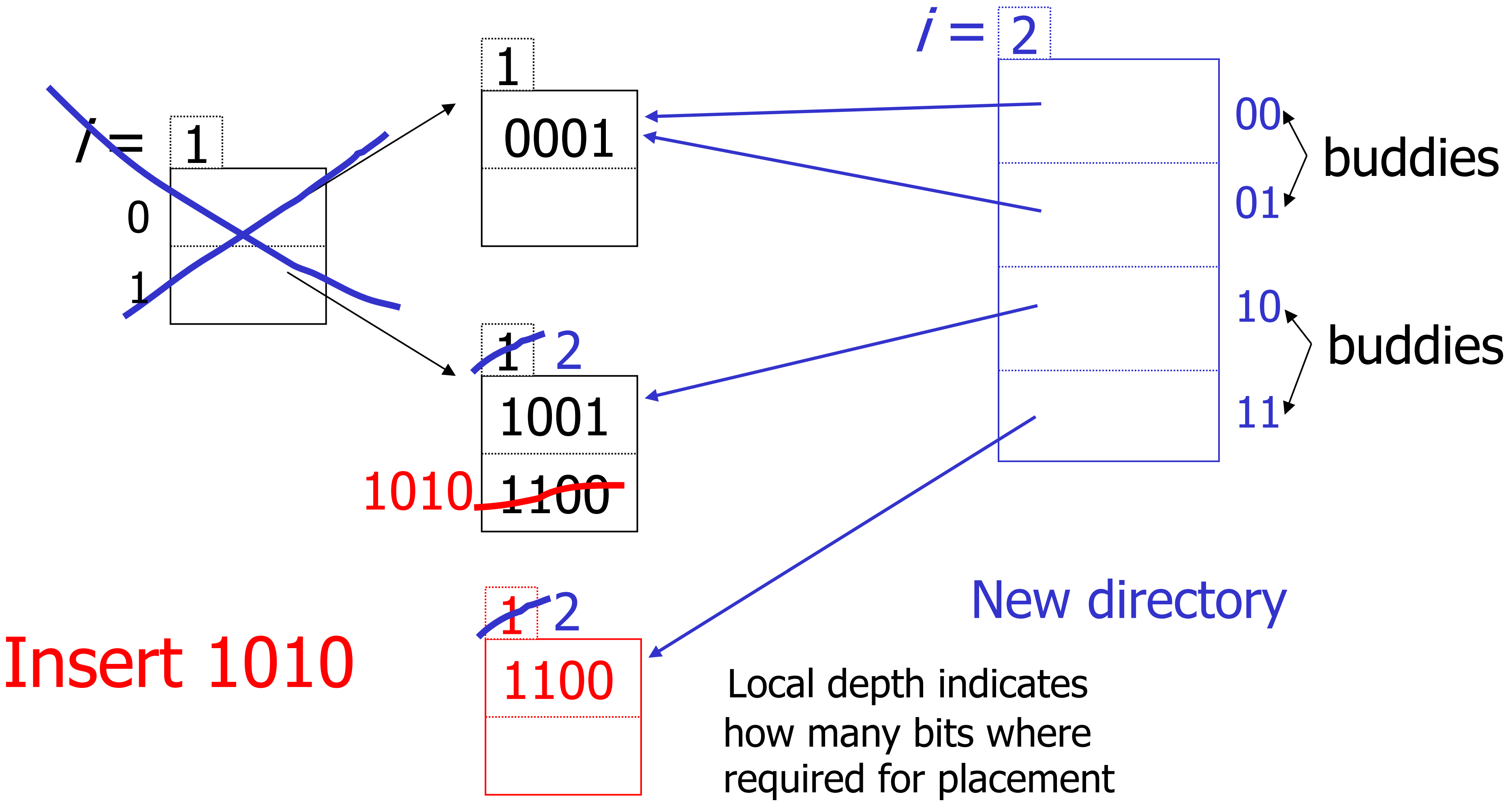
Example: h(k) is 4 bits; 2 keys/bucket



Insert 1010

Main idea:
we need 2 bits to distinguish
1001
1100
1010

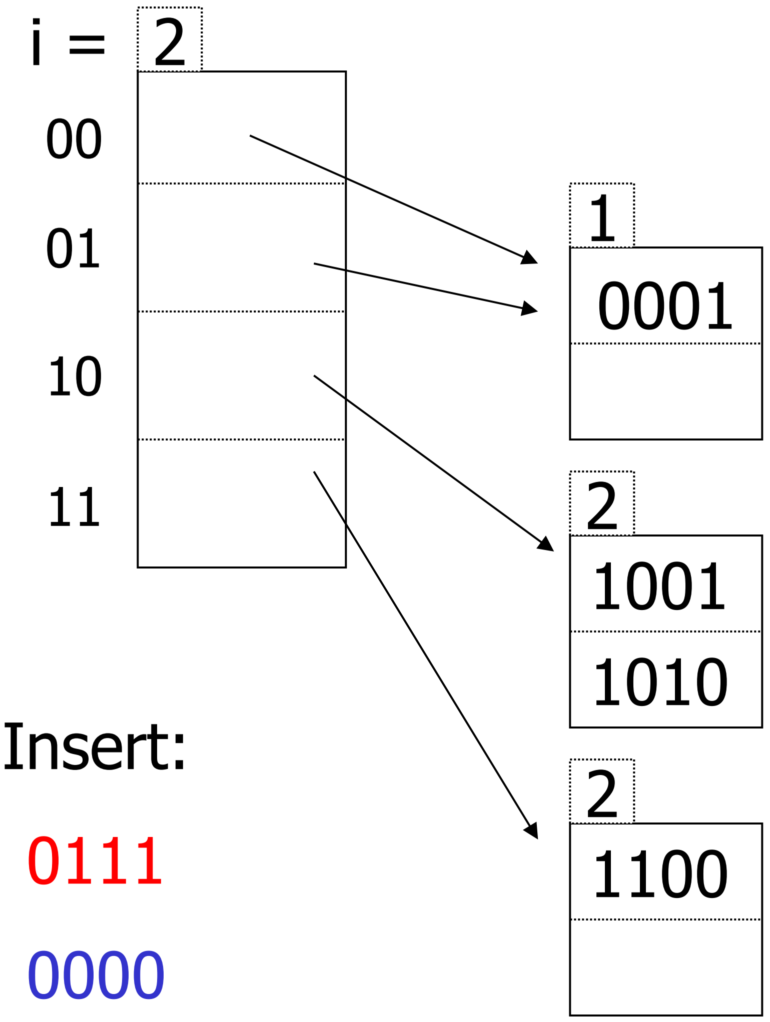
Example: h(k) is 4 bits; 2 keys/bucket



Insert 1010

Local depth indicates how many bits where required for placement of records (keys) in bucket

Example continued

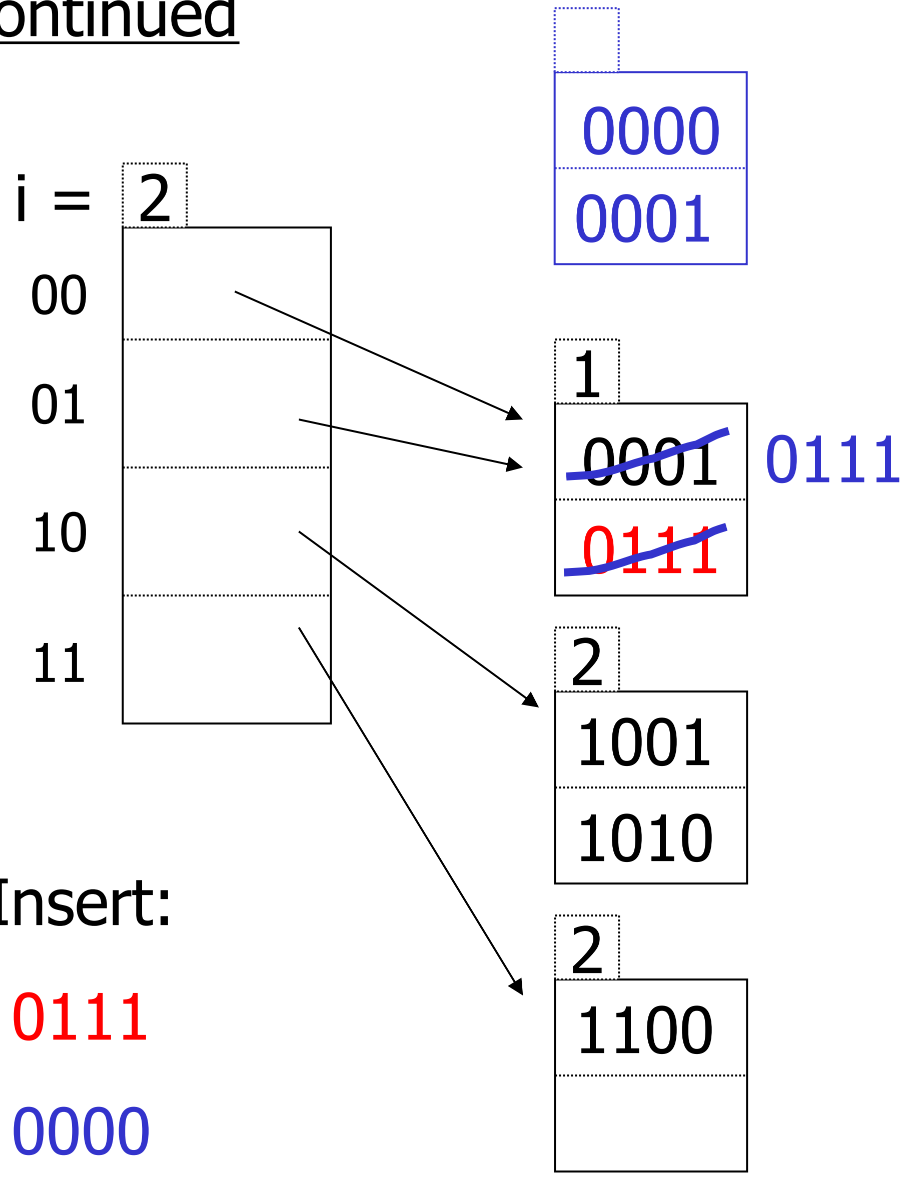


Insert:

0111

0000

Example continued

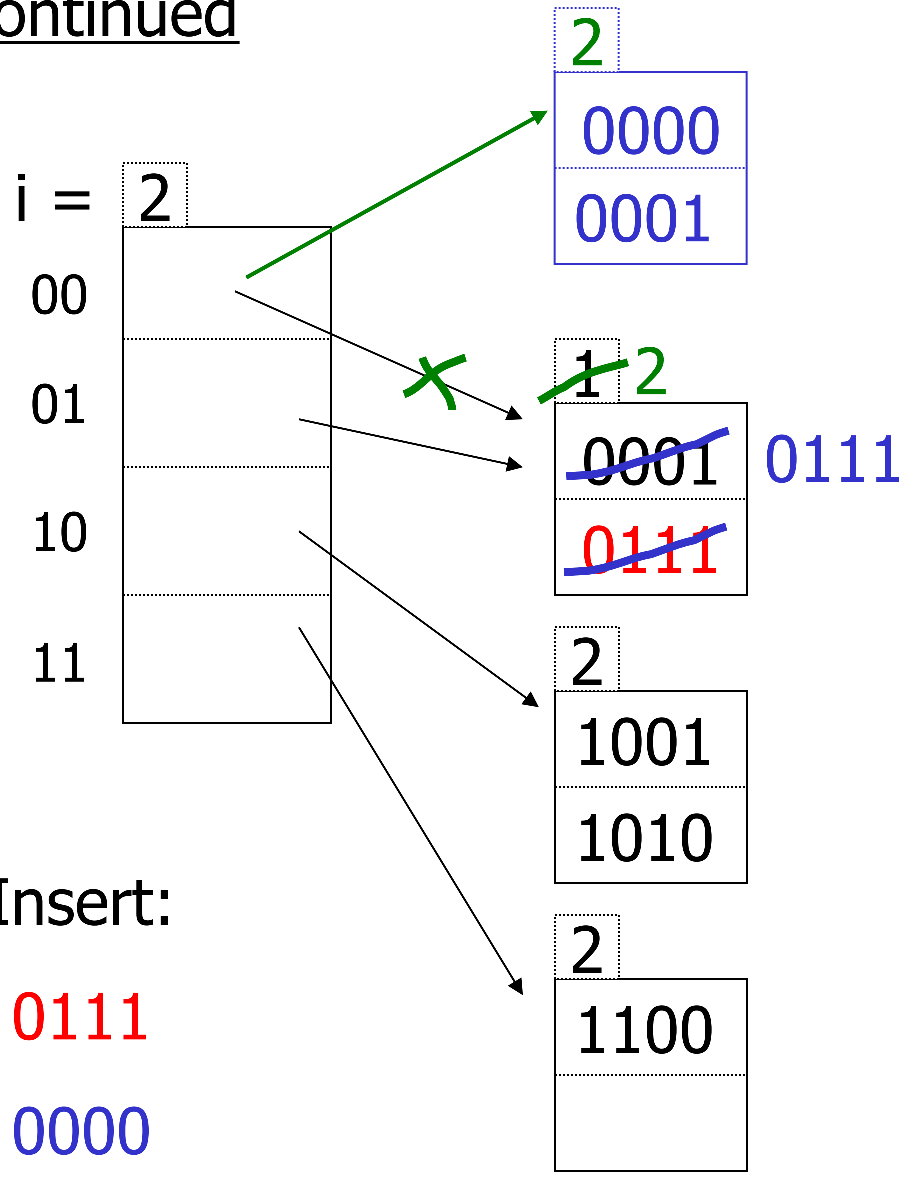


Insert:

0111

0000

Example continued

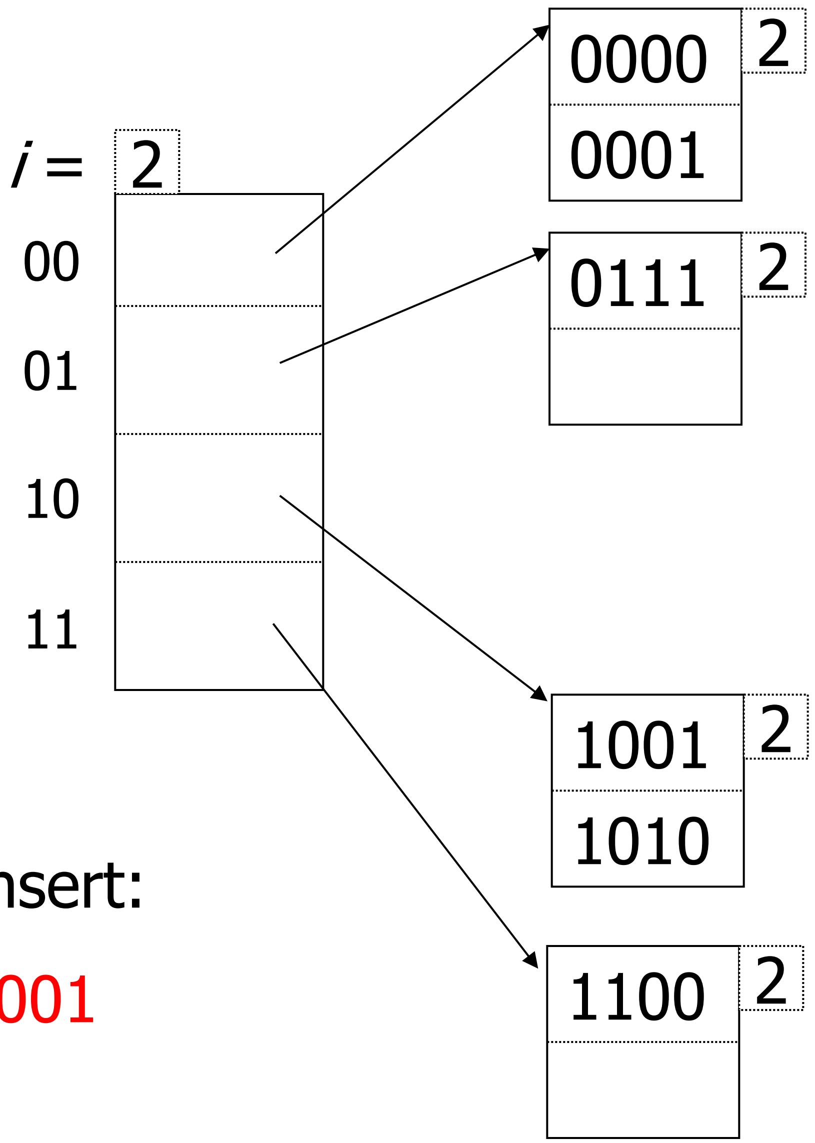


Insert:

0111

0000

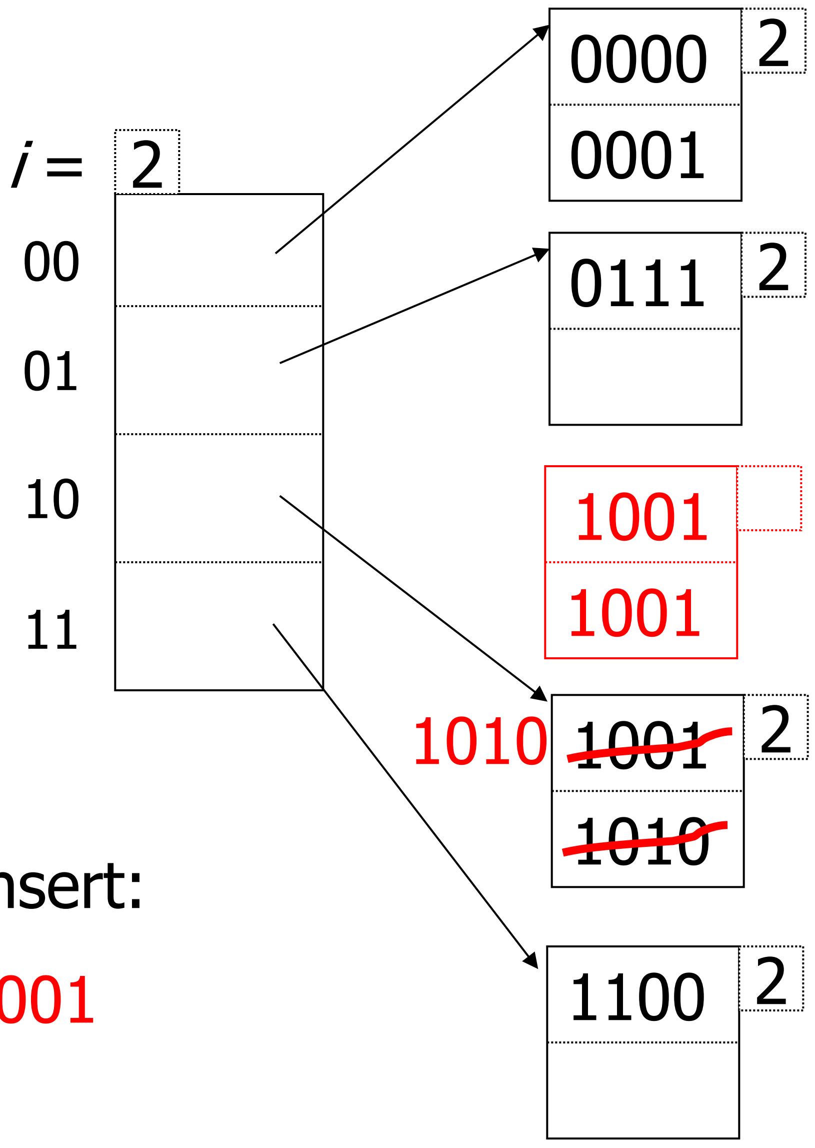
Example continued



Insert:

1001

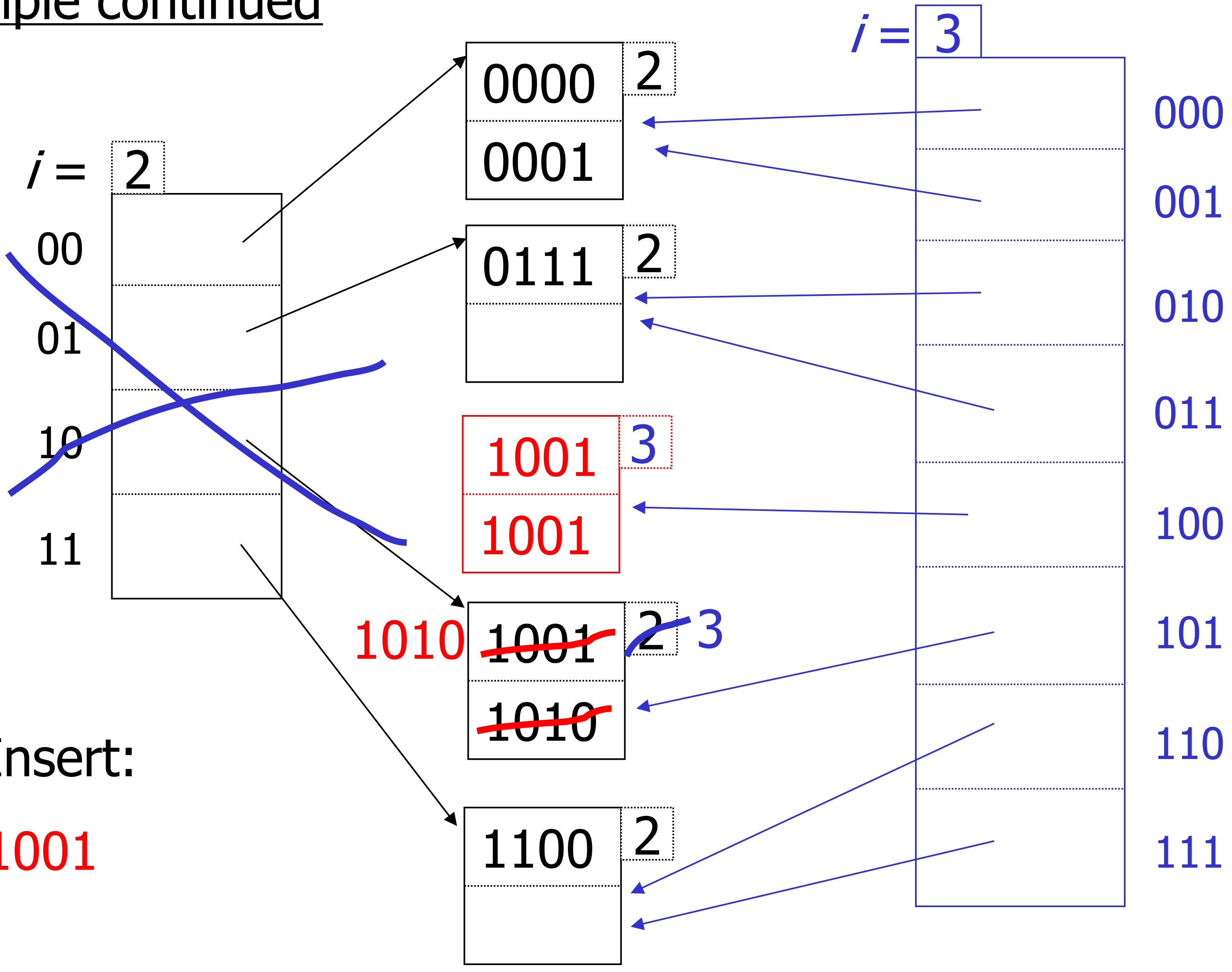
Example continued



Insert:

1001

Example continued



Insert:

1001

Extensible hashing: <u>deletion</u>

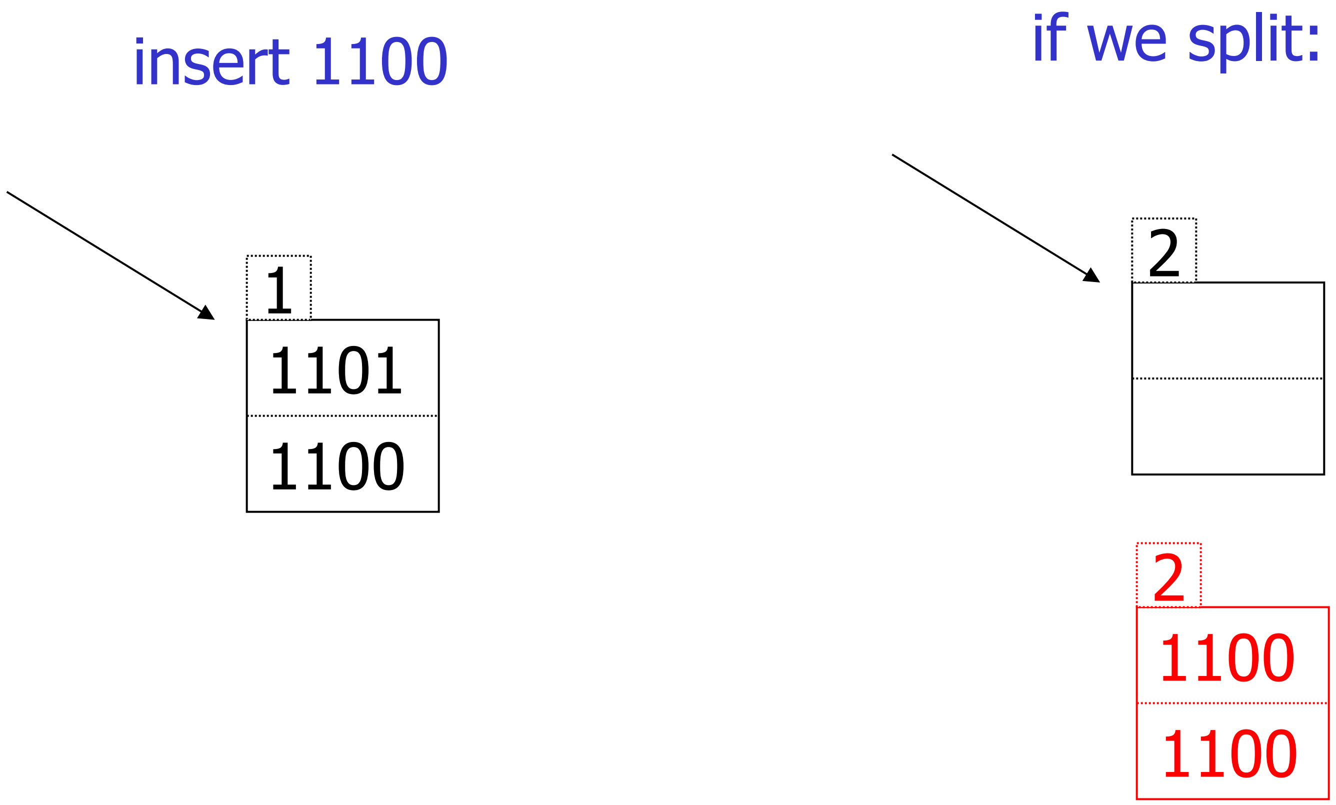
- Merge blocks and cut directory if possible
  (Reverse insert procedure)
- Two blocks can be merged after a deletion if all records in these blocks can fit in a single block
- One of these two blocks can be removed and the local depth of the other block one can be decreased by 1
- If all local depths are strictly smaller than the global depth i, then the directory can be cut in half and the

Deletion example:

- Run through insert example in reverse!

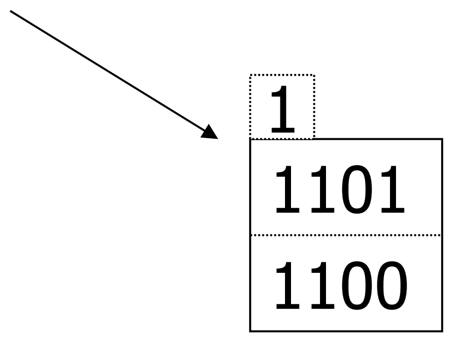
Note: Still need overflow chains

- Example: many records with duplicate keys

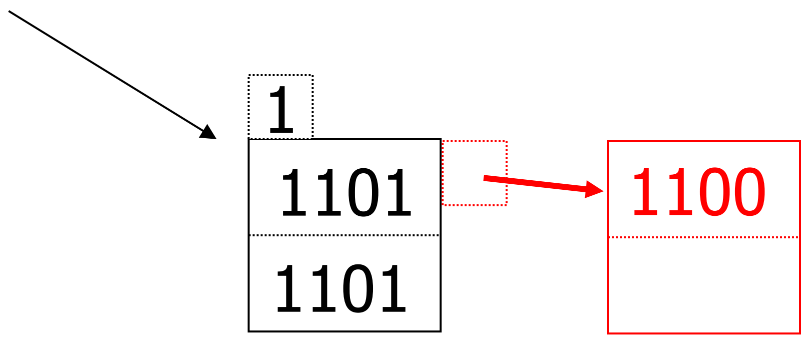


Solution: overflow chains

insert 1100



add overflow block:


1
1101
1101
1100