Linear Hashing

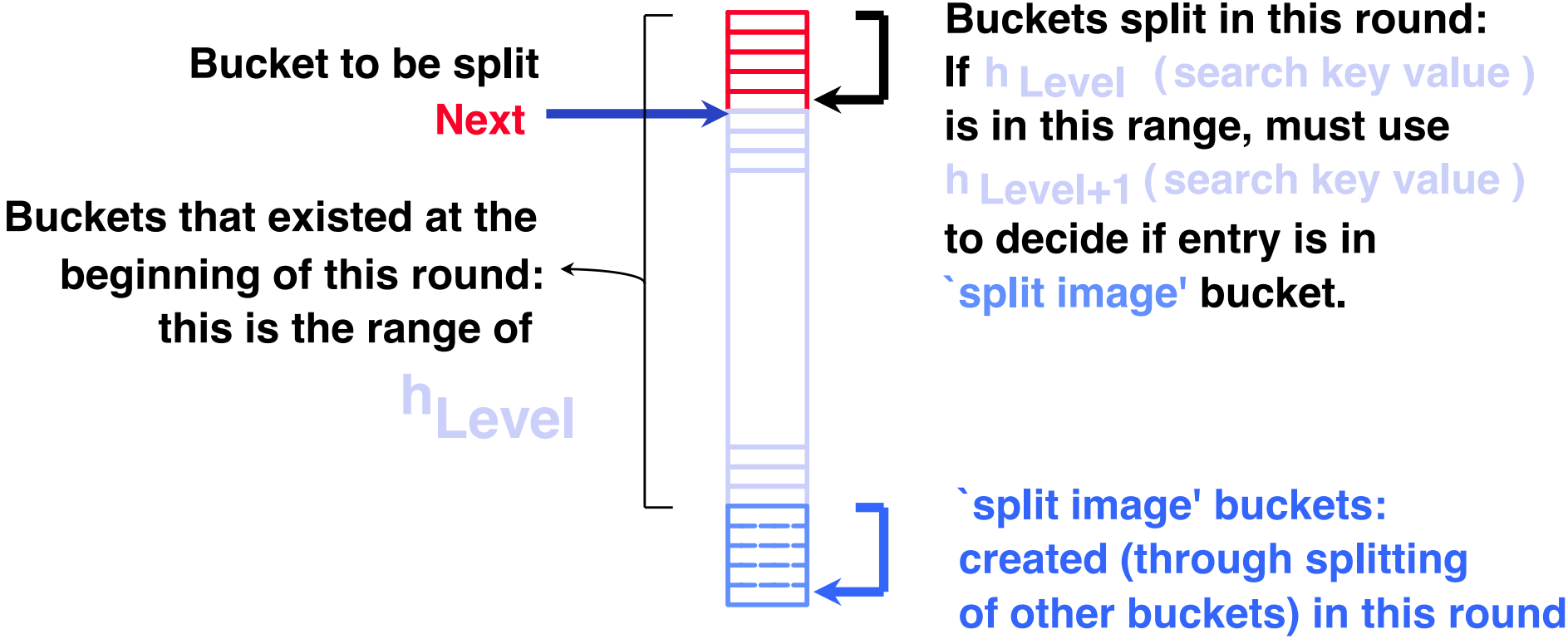
- This is another dynamic hashing scheme, an alternative to Extendible Hashing.
- LH handles the problem of long overflow chains without using a directory, and handles duplicates.
- *Idea*: Use a family of hash functions $\mathbf{h}_0$, $\mathbf{h}_1$, $\mathbf{h}_2$, ...
  - $\mathbf{h}_i(key) = \mathbf{h}(key) \bmod (2^i M)$; M = initial # buckets
  - **h** is some hash function (range is *not* 0 to M-1)
  - If $M = 2^{d0}$, for some *d0*, $\mathbf{h}_i$ consists of applying **h** and looking at the last *di* bits, where *di* = *d0* + *i*.
  - $\mathbf{h}_{i+1}$ doubles the range of $\mathbf{h}_i$ (similar to directory doubling)

Linear Hashing (Contd.)

- Directory avoided in LH by using overflow pages, and choosing bucket to split round-robin.
  - Splitting proceeds in `rounds'. Round ends when all M_R initial (for round *R*) buckets are split. Buckets 0 to *Next-1* have been split; *Next* to M_R yet to be split.
  - Current round number is *Level*.
  - **Search:** To find bucket for data entry *r,* find $\mathbf{h}_{Level}(r)$*:*
    - If $\mathbf{h}_{Level}(r)$ in range `*Next* to M_R', *r* belongs here.
    - Else, r could belong to bucket $\mathbf{h}_{Level}(r)$ or bucket $\mathbf{h}_{Level}(r) + M_R$; must apply $\mathbf{h}_{Level+1}(r)$ to find out.

Overview of LH File

- In the middle of a round.

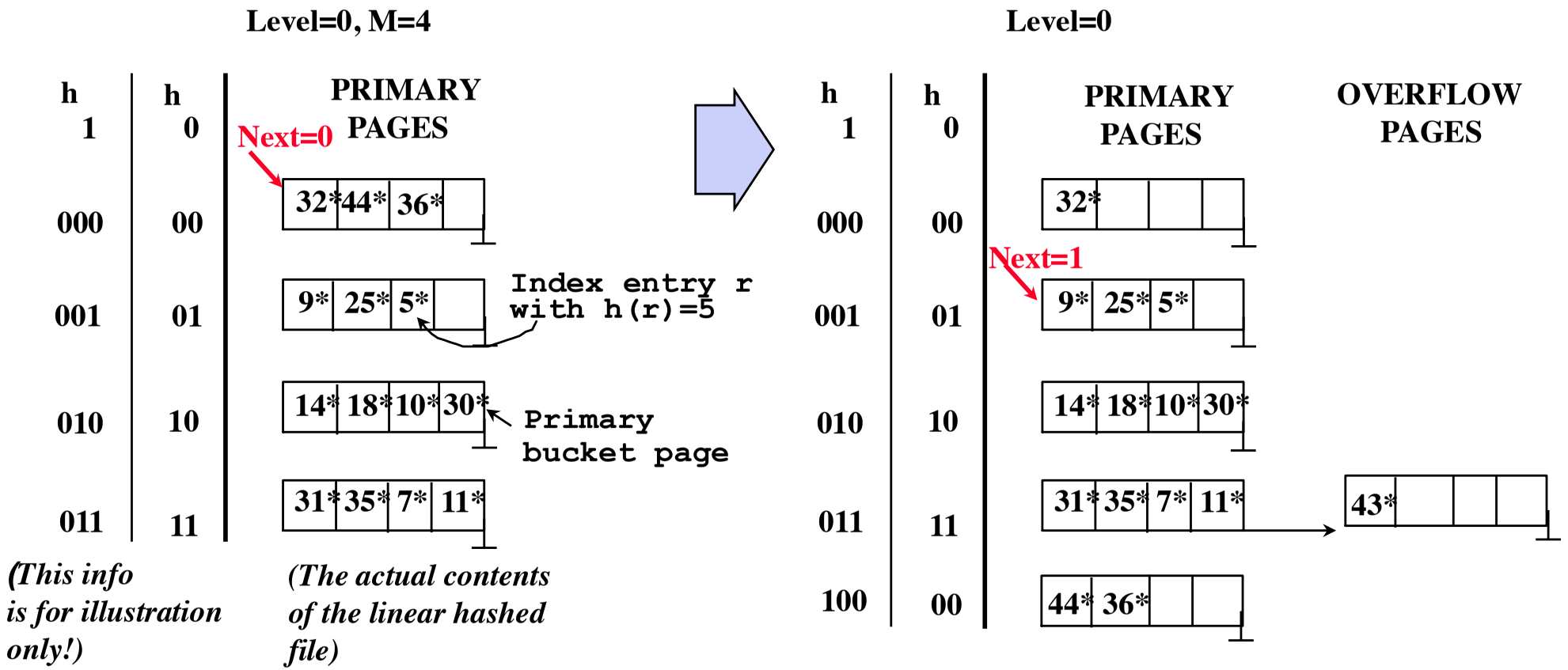


Linear Hashing (Contd.)

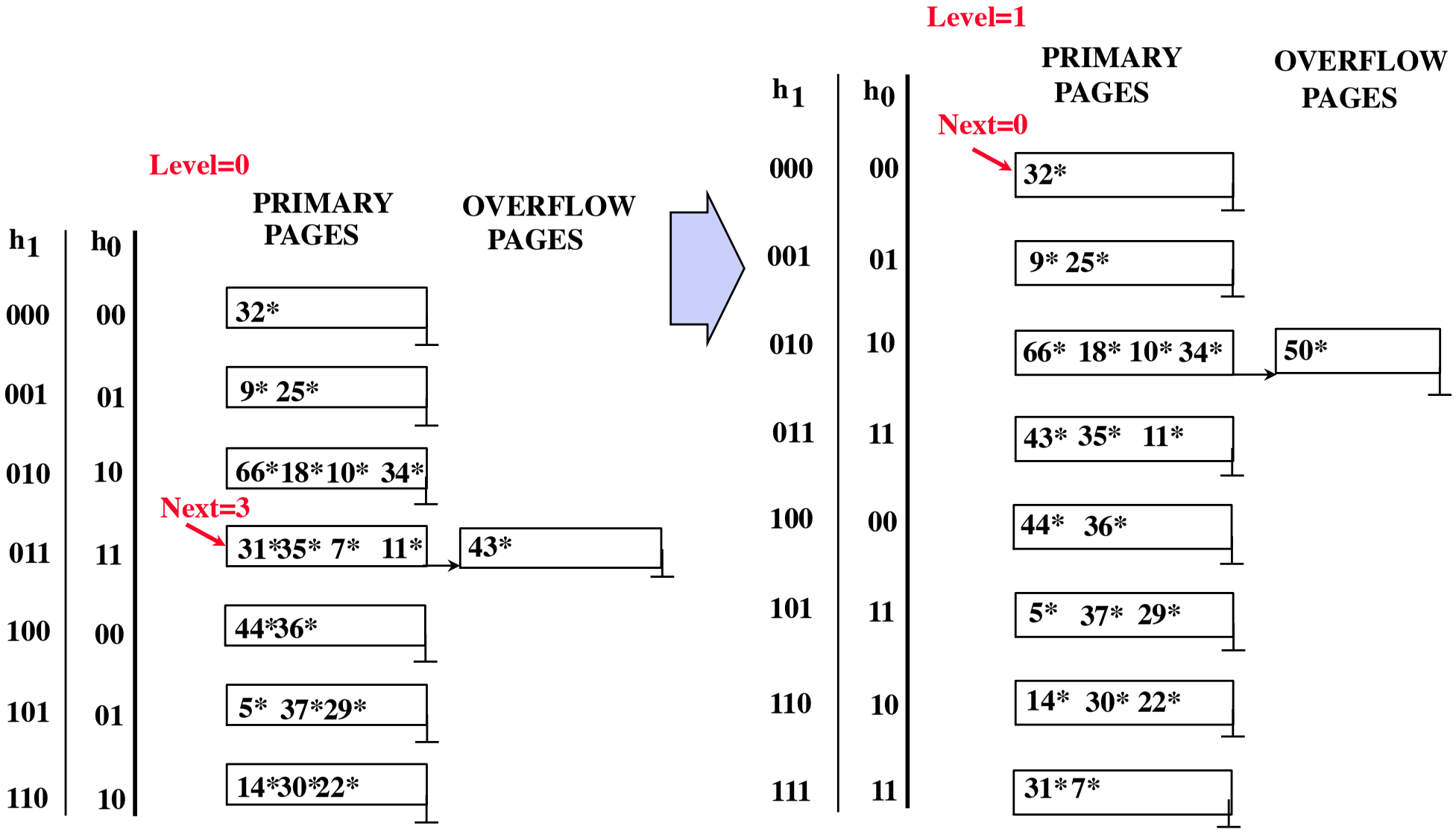
- **Insert**: Find bucket by applying $\mathbf{h}_{Level}$ / $\mathbf{h}_{Level+1}$:
  - If bucket to insert into is full:
    - Add overflow page and insert data entry.
    - (*Maybe*) Split *Next* bucket and increment *Next*.
- Can choose any criterion to `trigger' split.
- Since buckets are split round-robin, long overflow chains don't develop!
- Doubling of directory in Extendible Hashing is similar; switching of hash functions is *implicit* in how the # of bits examined is increased.

Example of Linear Hashing

- On split, $h_{Level+1}$ is used to re-distribute entries.



Example: End of a Round



LH Described as a Variant of EH

- The two schemes are actually quite similar:
  - Begin with an EH index where directory has *M* elements.
  - Use overflow pages, split buckets round-robin.
  - First split is at bucket 0. (Imagine directory being doubled at this point.) But elements <1,*M*+1>, <2,*M*+2>, ... are the same. So, need only create directory element *M*, which differs from 0, now.
    - When bucket 1 splits, create directory element *M*+1, etc.
- So, directory can double gradually. Also, primary bucket pages are created in order. If they are *allocated* in sequence too (so that finding i'th is easy), we actually don't need a directory! Voila, LH.