Tutorial 8

CSC 343
Winter 2018

NAFIZ HOSSAIN (NAFIZ.HOSSAIN@UTORONTO.CA)

UNIVERSITY OF TORONTO MISSISSAUGA

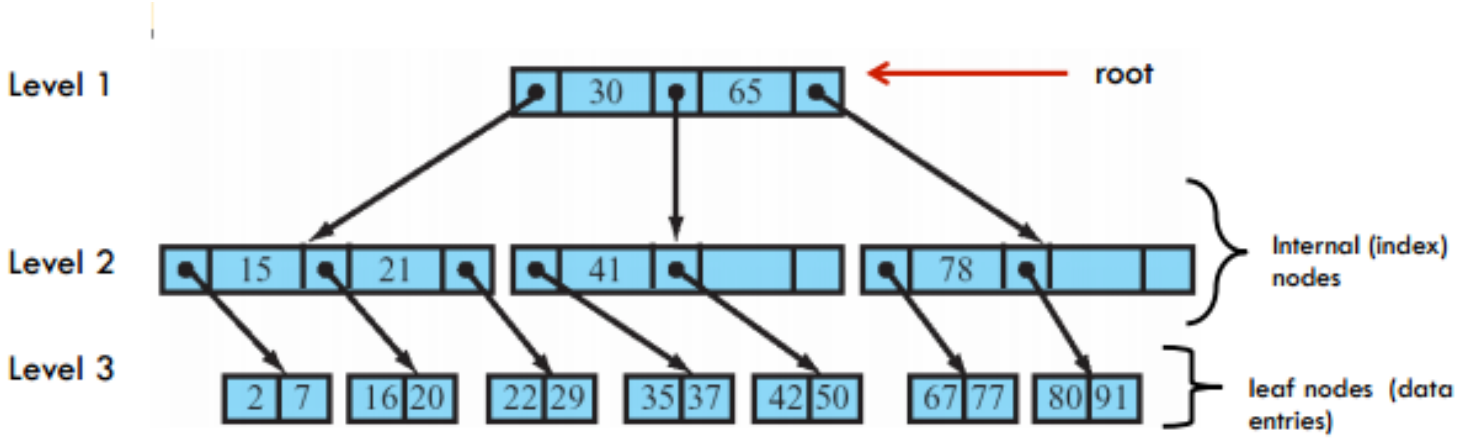
UNIVERSITY OF TORONTO MISSISSAUGA

Tree-based Indexes

The B+ tree structure is the most common index type in databases.

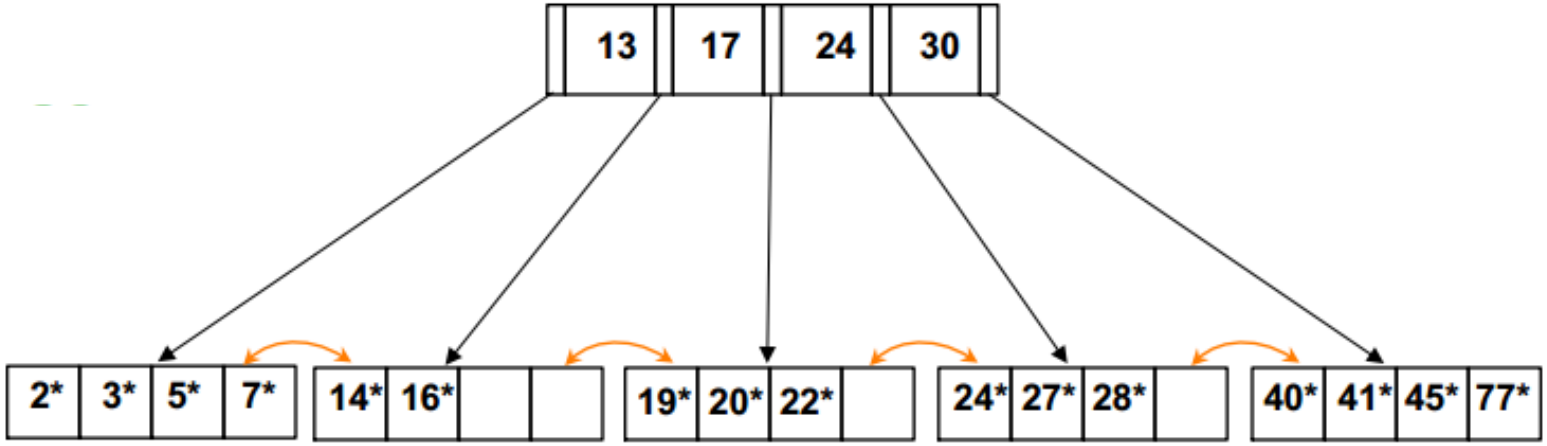
Each node is at least 50% full.

Supports equality and range-searches efficiently.



The lowest level of the tree, called the leaf level.

Tree-based Indexes (Cont.)



Example: to search for entry 5*

- Follow the left-most pointer, since 5<13.
- Yield result.

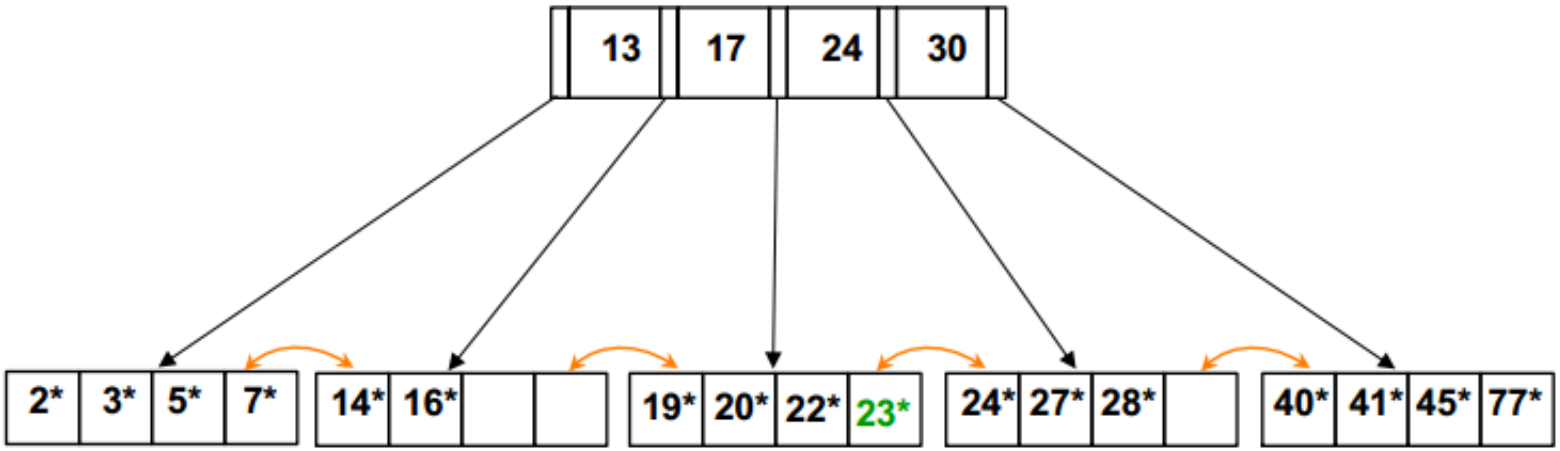
Example: to search for entry 15*

- Follow the second pointer, since 13<=15<17.
- Do not yield result.

Tree-based Indexes (Cont.)

Example: to insert entry 23*

- Follow the third pointer, since 17<=23<24.
- Have enough space, done.

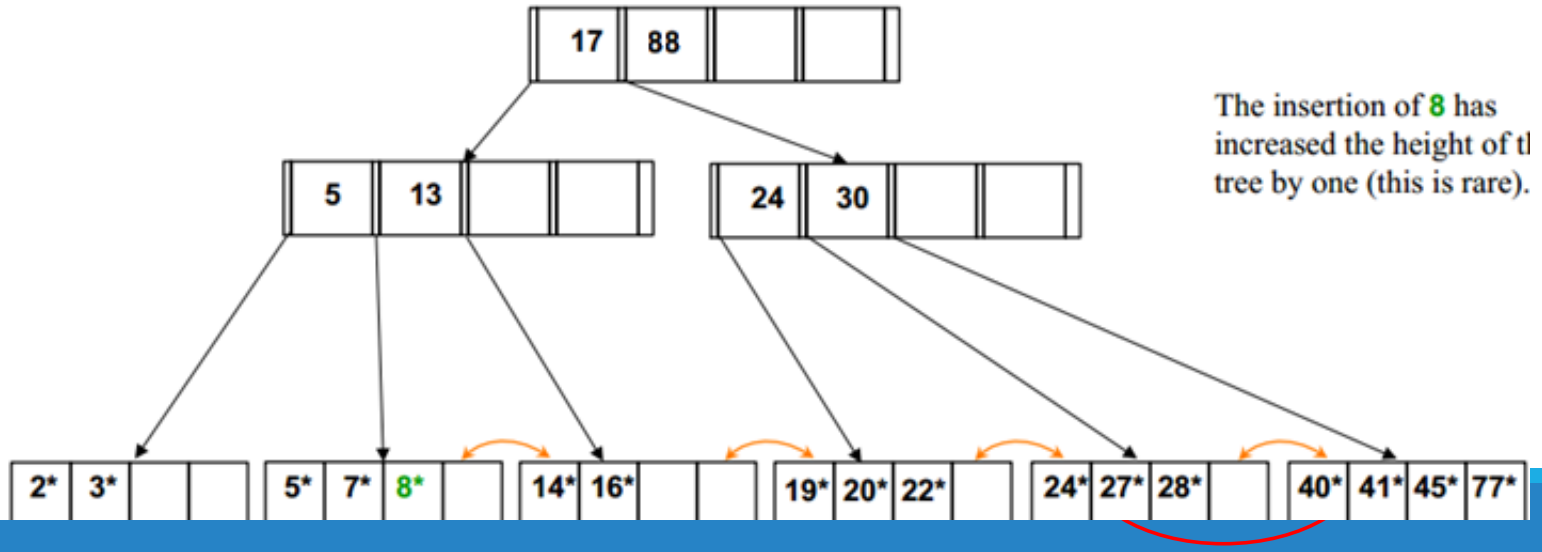


Tree-based Indexes (Cont.)

UNIVERSITY OF TORONTO MISSISSAUGA

Example: to insert entry 8*

- Follow the left-most pointer, since 8<13
- The insertion will cause overfill, split the leaf node into two nodes and redistribute the data evenly between them.
- “5” is middle key, so it is copied up.
- When inserting “5” into the parent node, it will cause overfill again. “17” is middle key and is pushed up.



UNIVERSITY OF TORONTO MISSISSAUGA

Tree-based Indexes (Cont.)

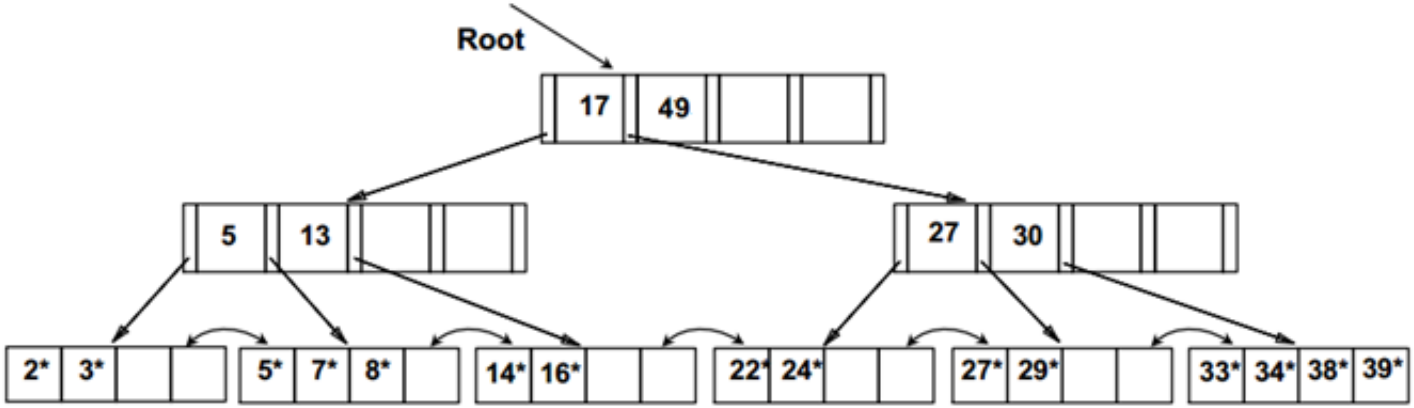
Example: to delete entry 19*

- Simply remove it and done because the leaf is at least half-full.

Example: to delete entry 20*

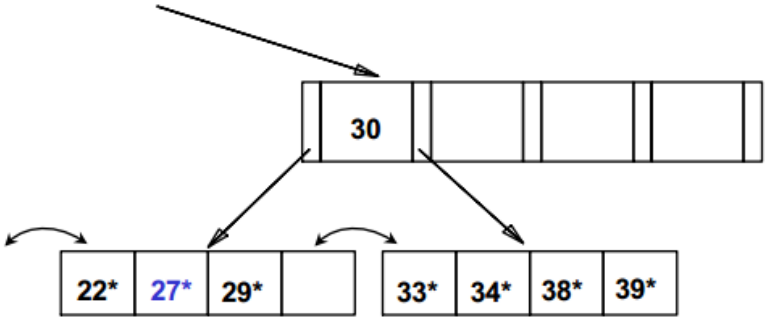
- After the deletion, the leaf contains only one entry, try to redistribute, borrowing from sibling (adjacent node with the same parent).
- Move entry 24* to the leaf page that contained 20* and copy up the new splitting key 27 into the parent.

Tree-based Indexes (Cont.)



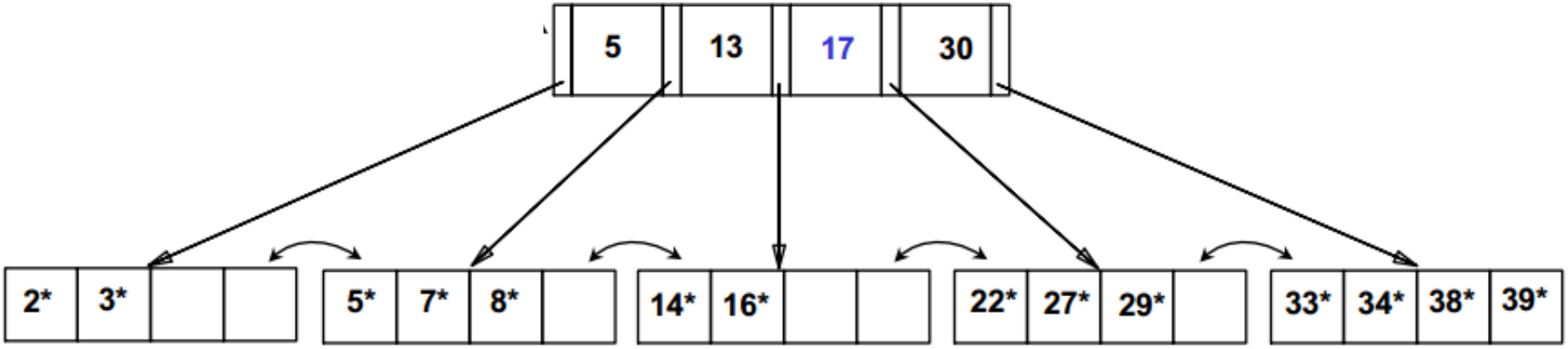
Example: to delete entry 24*

- After the deletion, the leaf contains only one entry 22*, and the sibling contains just two entries. Therefore, we can not redistribute entries.
- Merge leaf and sibling. 'Toss' the entry '27' in the parent.



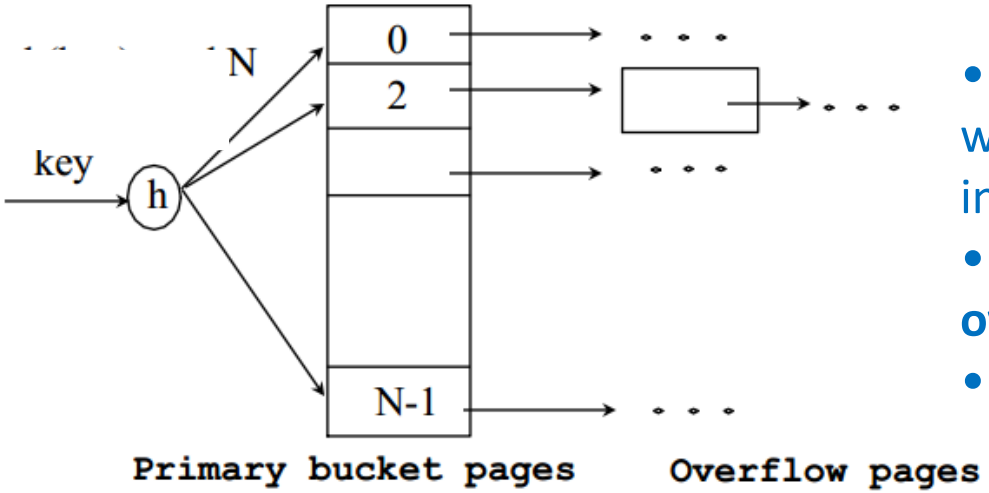
Tree-based Indexes (Cont.)

- Pull down of index entry.



Hash-based Indexes

Hash-based indexes are best for equality selections. They do not support efficient range searches.



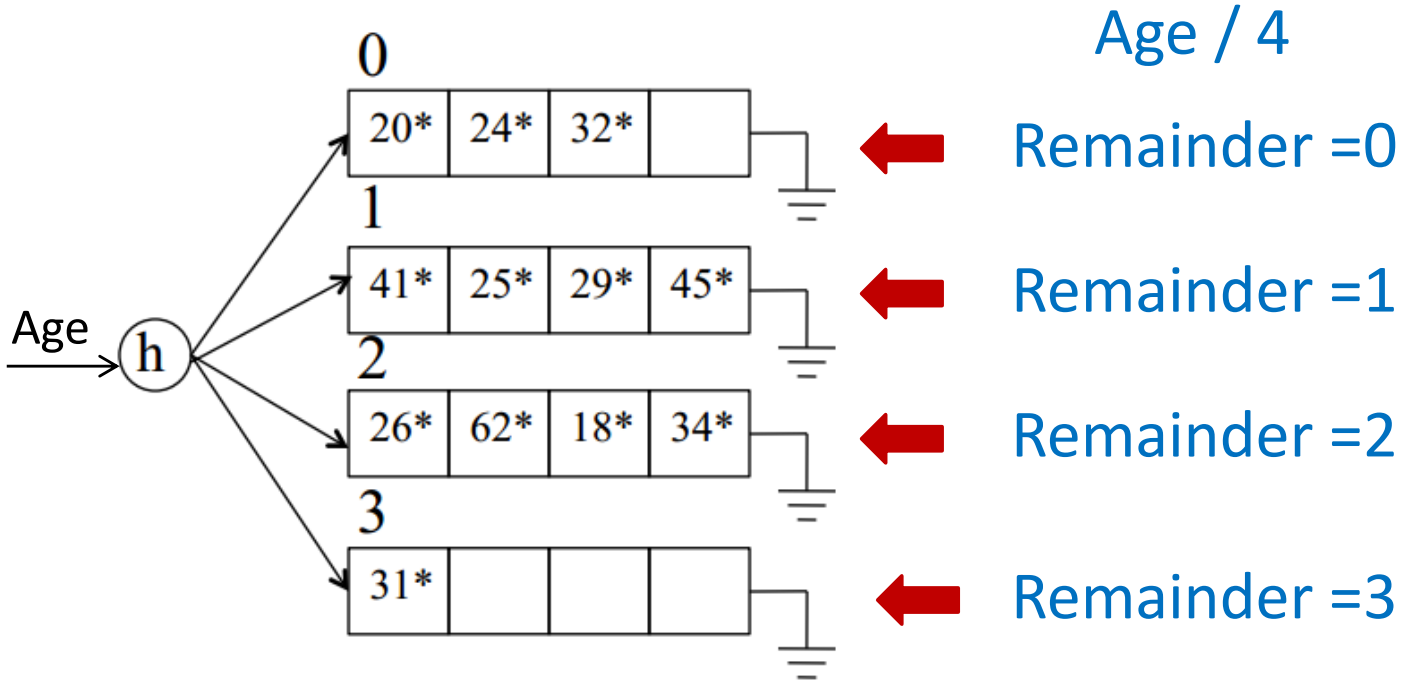
- a collection of buckets 0 through N-1, with one primary page per bucket initially
- bucket = **primary page**+ zero/more **overflow pages**
- buckets contains **data entries**

To search for data entry, we apply a hash function **h** to identify the bucket to which it belongs and then search this bucket.

Hash-based Indexes (Cont.)

Example

- Initially built over “Age” attribute, with 4 records/page and h(Age)=Age mod 4.

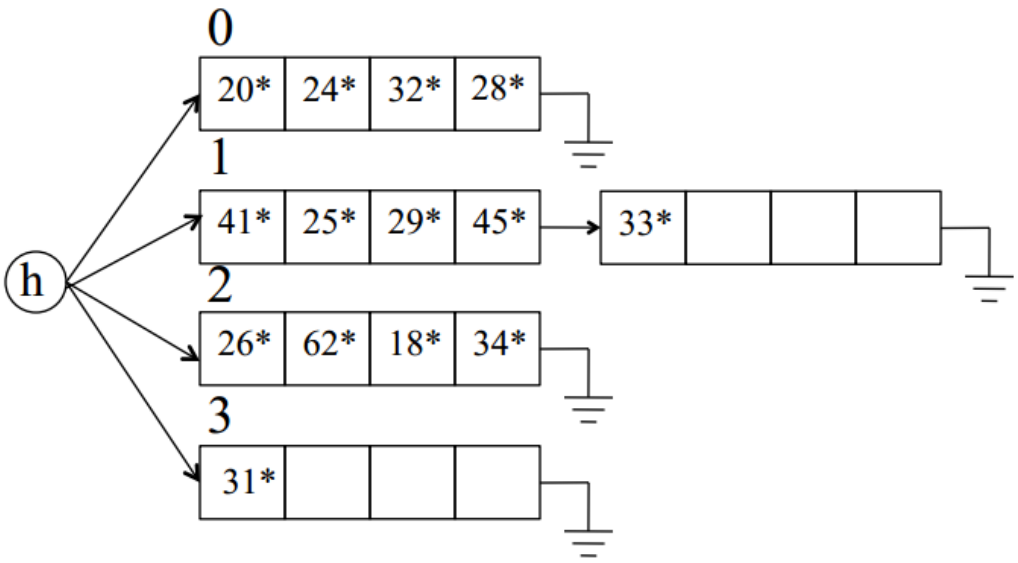


UNIVERSITY OF TORONTO MISSISSAUGA

Hash-based Indexes (Cont.)

To insert a data entry, we use the hash function to identify the correct bucket and then put the data entry there. If there is no space for this data entry, we allocate a new overflow page, put the data entry on this page, and add the page to the overflow chain of the bucket.

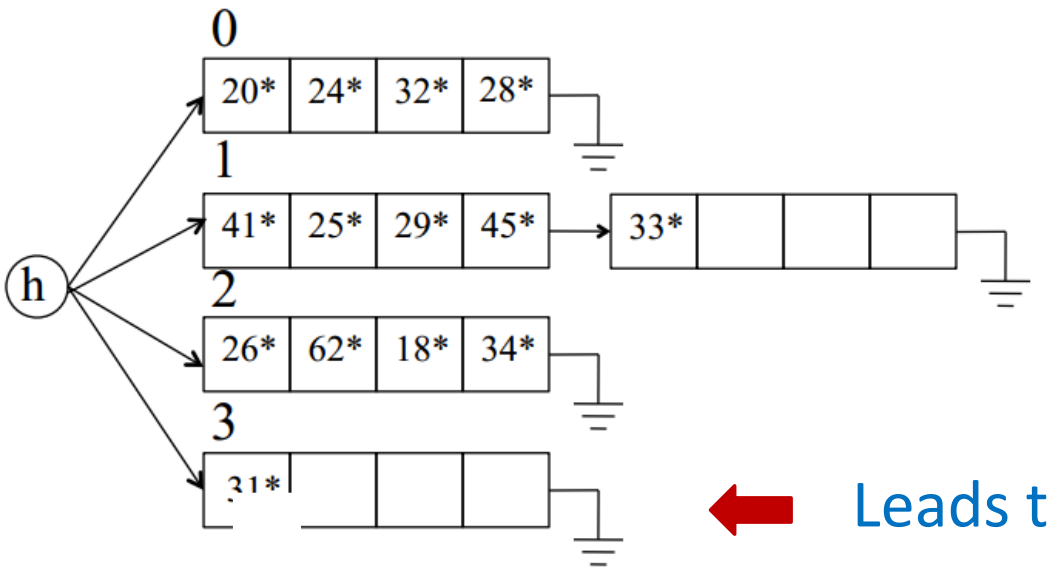
Example : adding 28, 33



Hash-based Indexes (Cont.)

To delete a data entry, we use the hashing function to identify the correct bucket, locate the data entry by searching the bucket, and then remove it. If this data entry is the last in an overflow page, the overflow page is removed from the overflow chain of the bucket and added to a list of free pages.

Example : deleting 31



Leads to empty page

UNIVERSITY OF TORONTO MISSISSAUGA

Any Questions?

- Do you have any questions?

- If you have any content that you would like to be added in a Tutorial, please let me know by Friday!
  - Email request to me: nafiz.hossain@utoronto.ca