CSE 350: Advanced Data Structures and Indexes (Spring 2026)

Lecture 16: B-Tree Bulk Loading

3/26/2026

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

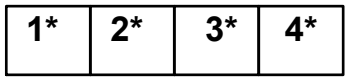
Loading a B+-tree with pre-existing data

- If we have a large collection of data, and we want to create a B-tree on their key
  - Doing so by repeatedly inserting records can be very slow.
  - Also leads to minimal leaf utilization --- why?

Loading a B+-tree with pre-existing data

- If we have a large collection of data, and we want to create a B-tree on their key
  - Doing so by repeatedly inserting records can be very slow.
  - Also leads to minimal leaf utilization --- why?

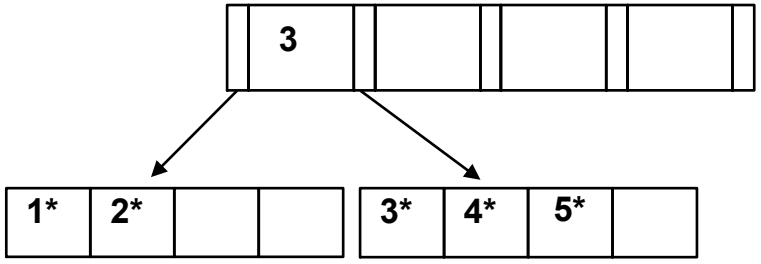
Special case : sequential insertion of 1, 2, 3, … N



Loading a B+-tree with pre-existing data

- If we have a large collection of data, and we want to create a B-tree on their key
  - Doing so by repeatedly inserting records can be very slow.
  - Also leads to minimal leaf utilization --- why?

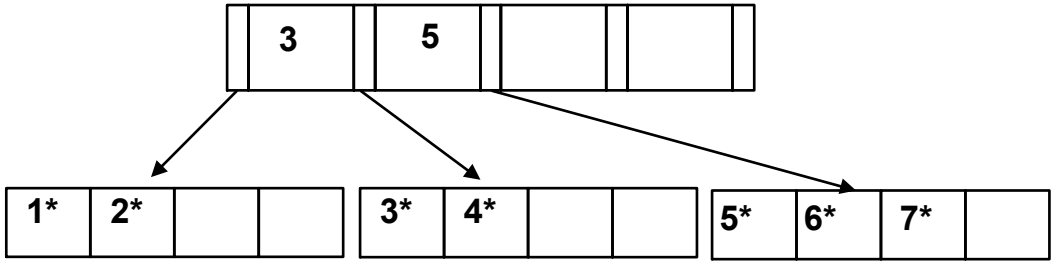
Special case: sequential insertion of 1, 2, 3, … N



Loading a B+-tree with pre-existing data

- If we have a large collection of data, and we want to create a B-tree on their key
  - Doing so by repeatedly inserting records can be very slow.
  - Also leads to minimal leaf utilization --- why?

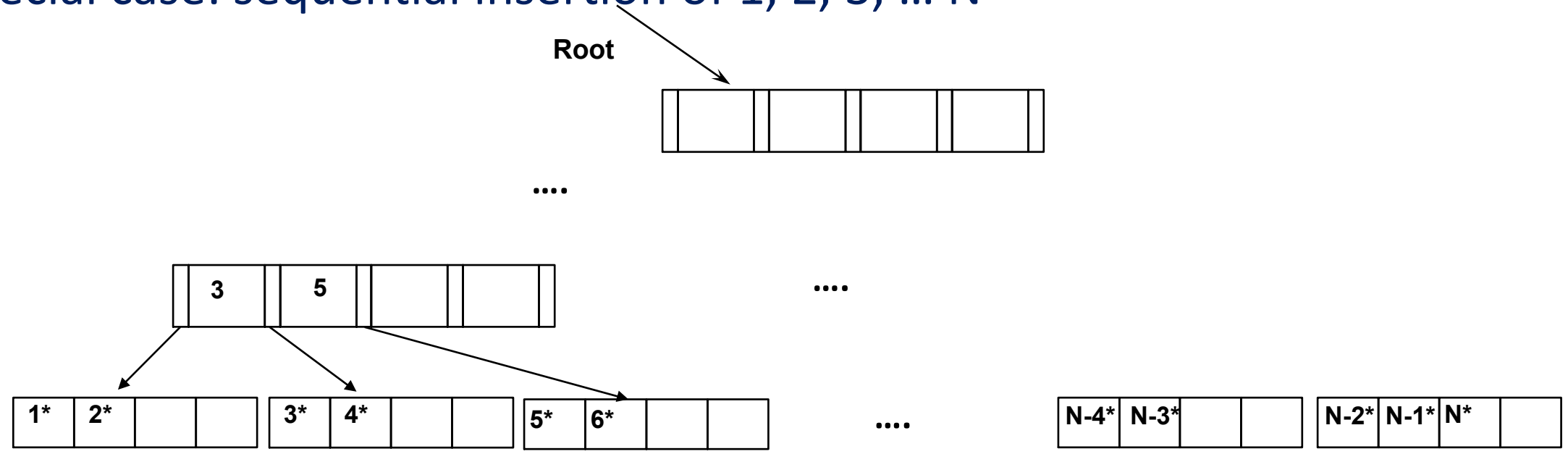
Special case: sequential insertion of 1, 2, 3, … N



Loading a B+-tree with pre-existing data

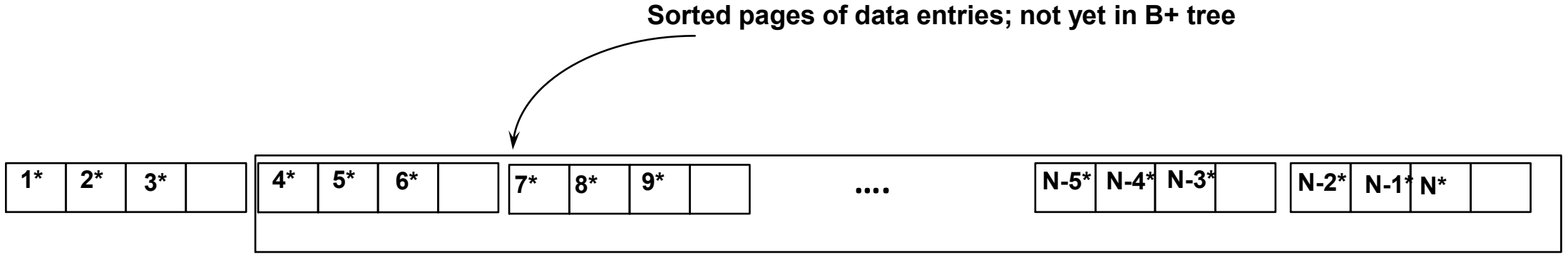
- If we have a large collection of data, and we want to create a B-tree on their key
  - Doing so by repeatedly inserting records can be very slow.
  - Also leads to minimal leaf utilization --- why?

Special case: sequential insertion of 1, 2, 3, … N



Loading a B+-tree with pre-existing data

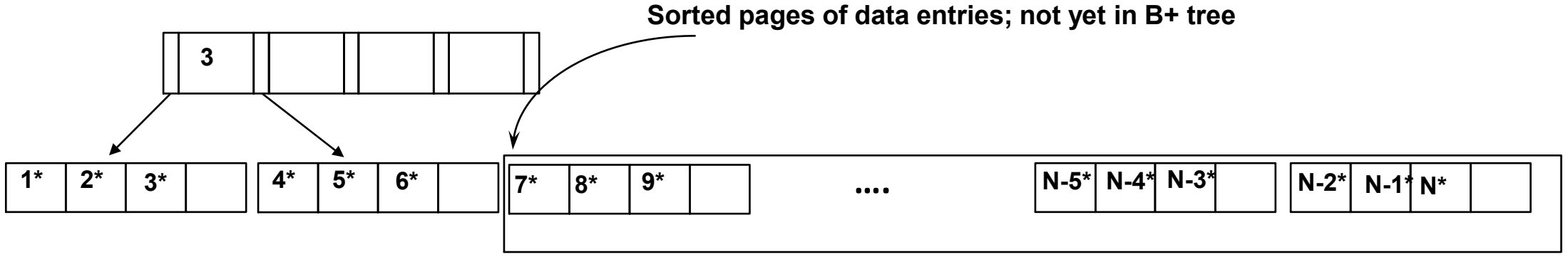
- If *Bulk loading* can be done much more efficiently.
  - fill factor: the default utilization ratio for leaf and internal pages (may vary for leaf and internal pages)
    typical values: 70%/80%
- *Initialization*: Sort all data entries, insert pointer to first (leaf) page in a new (root) page.



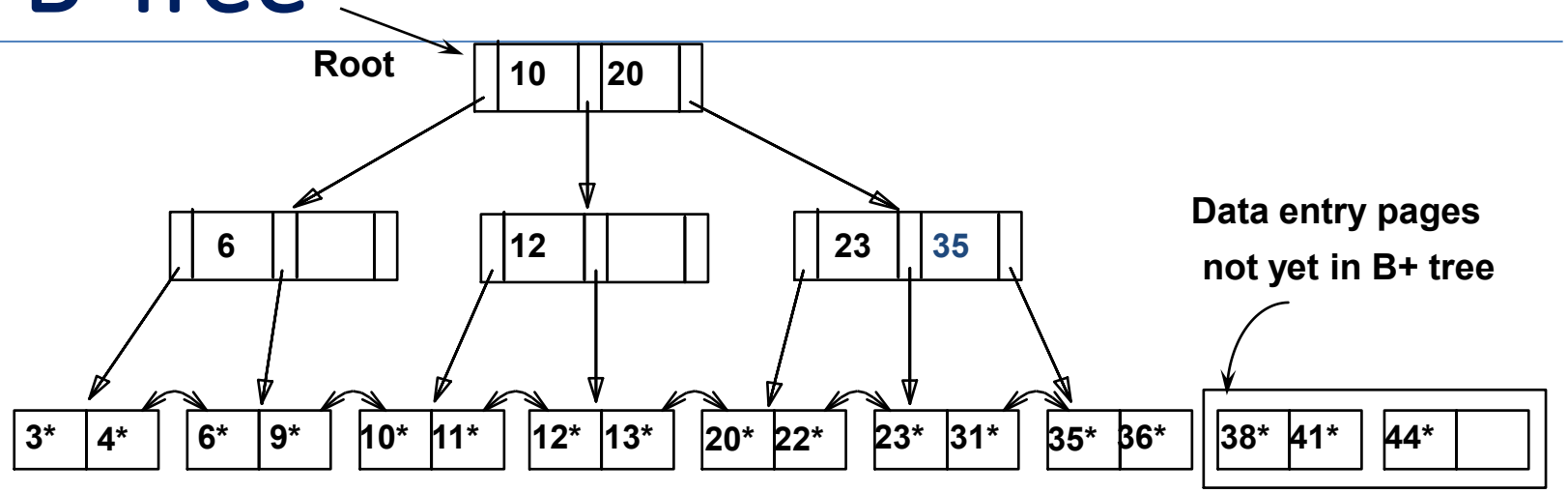
Loading a B+-tree with pre-existing data

- If *Bulk loading* can be done much more efficiently.
  - fill factor: the default utilization ratio for leaf and internal pages (may vary for leaf and internal pages)
    typical values: 70%/80%
- *Initialization*: Sort all data entries, insert pointer to first (leaf) page in a new (root) page.

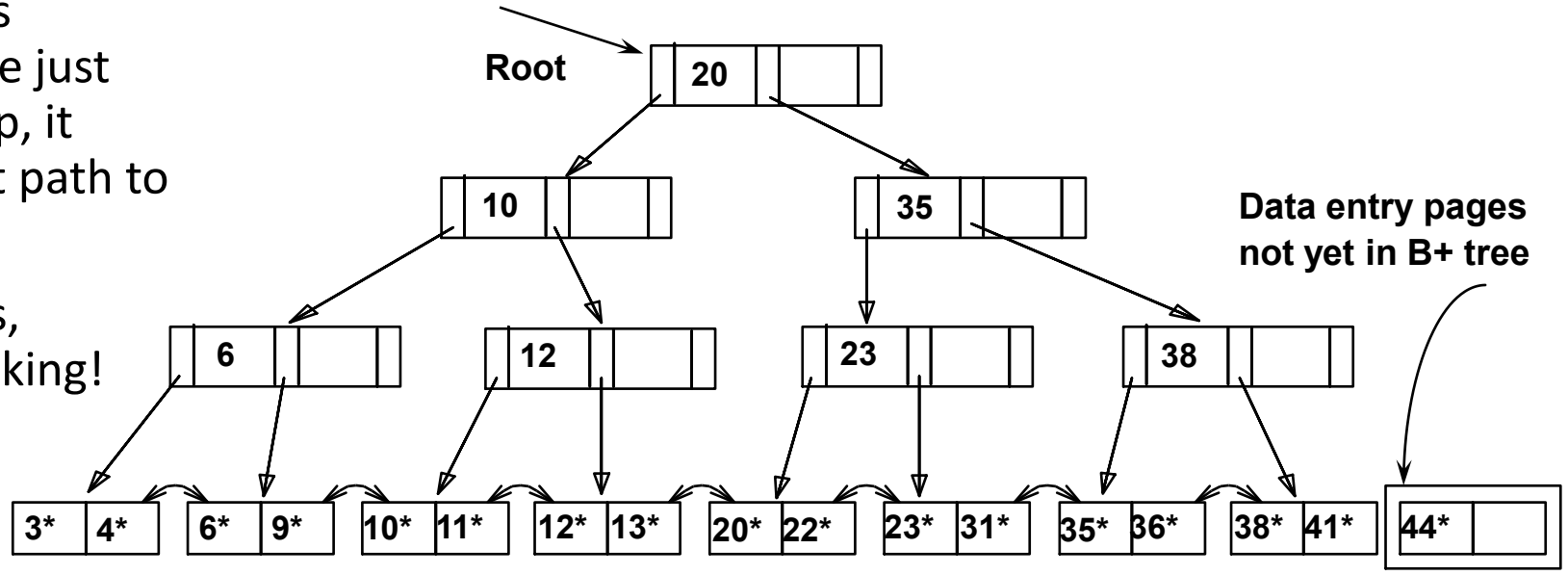
Create new root in memory, but don't write to disk yet
– only write when it's full or we finish bulk loading



Bulk loading of a B-Tree



- Index entries for leaf pages always entered into right-most index page just above leaf level. When this fills up, it splits. (Split may go up right-most path to the root.)
- Much faster than repeated inserts, especially when one considers locking!



Analysis of B-Tree storage cost

- Suppose the usable page size is P (bytes), each record is r (bytes), the index key is k bytes, record ID or page number is q bytes, and N records in total in the heap file.
- Assume we use alternative 2 for the data entries.
- Bottom-up analysis:
  - Number of pages in the heap file: $M = \lceil \frac{N}{\lfloor P/r \rfloor} \rceil$.
  - Number of data entries: N (one per record)
  - Size of a data entry: $k + q$ bytes (without considering alignments)
  - Number of pages in leaf level:
    - $N' = \lceil \frac{N}{\lfloor P/(k+q) \rfloor} \rceil$
    - If the average leaf page utilization ratio is u:

      $$N' = \lceil \frac{N}{\lfloor P * u/(k+q) \rfloor} \rceil$$

    - Let B be the number of data entries per leaf page
      - $B = \lfloor P * u/(k+q) \rfloor$

Analysis of B-Tree storage cost

fill factor: the default utilization ratio when bulk loading the tree

- Internal levels:
  - Fan-out/number of index entries per page
    $f = \left\lfloor \frac{P \times u - q}{k+q} \right\rfloor + 1$ (u is the average utilization ratio: [0.5, 1))
  - Number of entries in the index level right above the leaf level: N' (one entry per leaf-level page)
  - Number of pages required in this level: N'/f
  - Number of entries in the level above: N'/f
  - Number of pages in the level above: N'/f^2
  - Recursively pages in each level:
    - N', N'/f, N'/f^2 , N'/f^3 …. 1=N'/f^{h-1}
    - So $h = \lceil \log_f N' \rceil + 1 = \left\lceil \log_f \lceil \frac{N}{B} \rceil \right\rceil + 1$
    - total number of internal pages $1 + f + \ \ldots + f^{h-1} = \frac{f^h - 1}{f-1} = O(N') = O(N/B)$
- Total number of pages in a B-Tree: $O(N') = O(\frac{N}{B})$