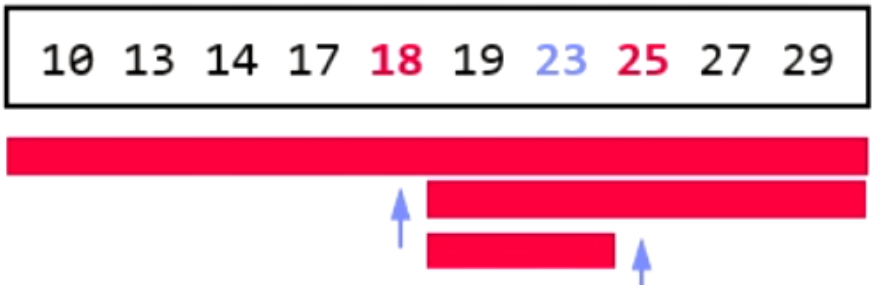
Binary Search

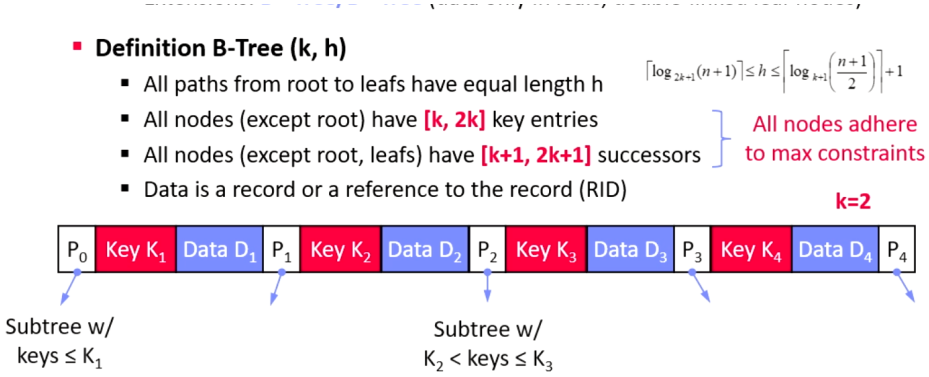
- pos = binarySearch(data,key=23)
- find key position within sorted data



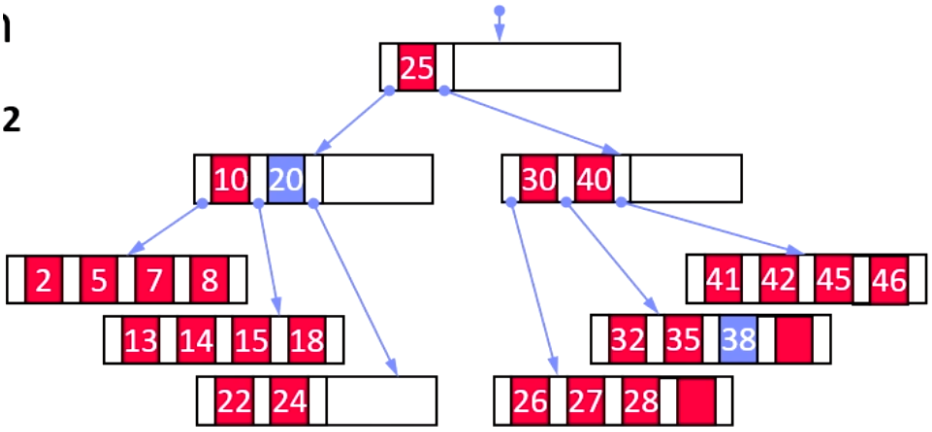
-
- optimizations
  - k-ary search for SIMD data-parallelism
    - ?
  - interpolation search: probe expexted pos in key range
    - e.g. search for “Bastian” in telephone book, dont start in the middle but rather at the beginning

BTree

- self balancing tree
- individual nodes stored as pages
  - [[Background Storage System]]
- each node contains data or reference to data
  - values sorted within node



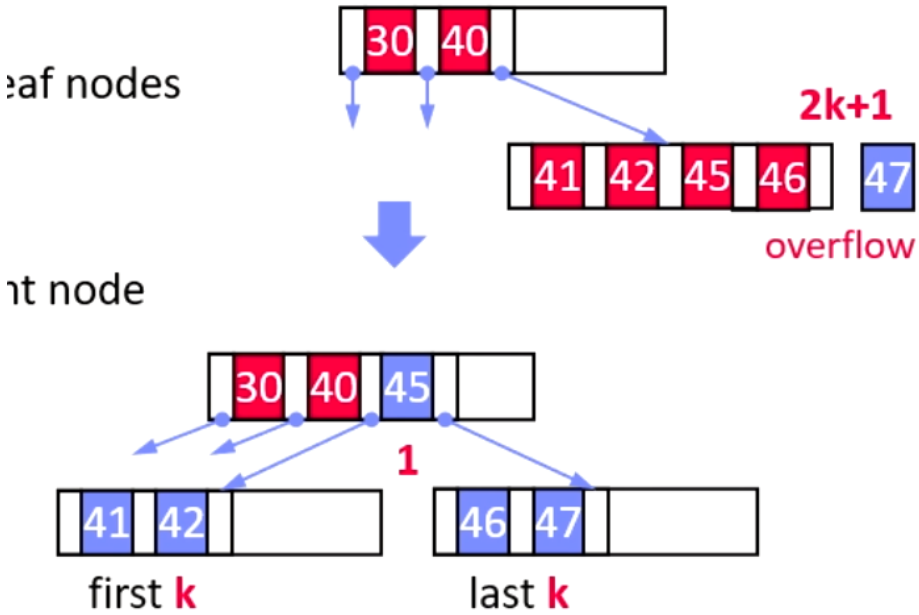
+ pointer left/right of value points to leaf with smaller/bigger values



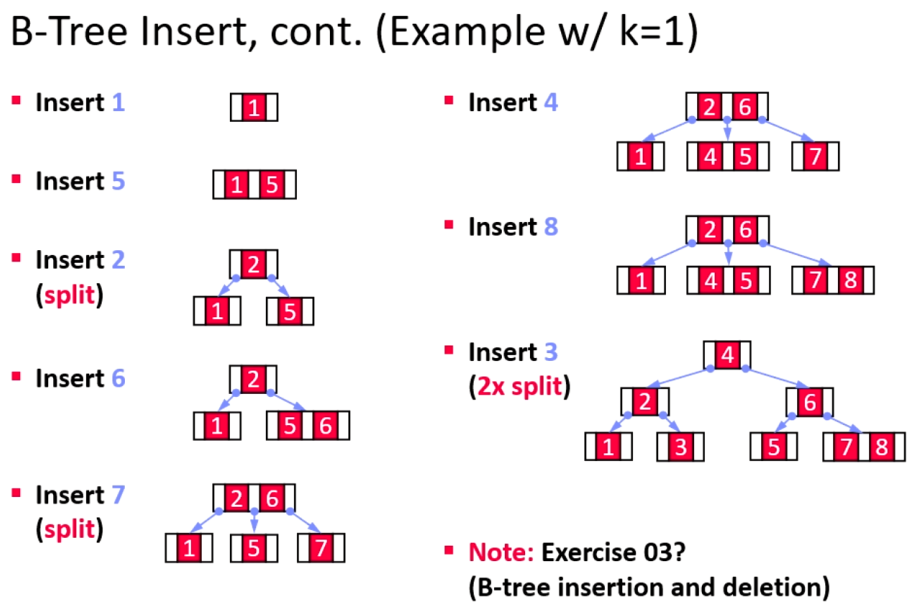
- **Lookup Q_K within a node**
  - Scan / binary search keys for Q_K, if $K_i=Q_K$, r
  - If node does not contain key
    - If leaf node, abort search w/ NULL (n
    - Decent into subtree Pi with $K_i < Q_K \leq$
- **Range Scan $Q_{L<K<U}$**
  - Lookup Q_L and call next K while $K<Q_U$ (kee

B-Tree Insert

- always insert into leaf nodes
- if node overflows (exceeds 2k entries) ==> node splitting
- node splitting
  - split into two leaf nodes
  - left node with first k entries
  - right node with last k entries
  - (k+1)th entry inserted into parent node
    - may cause recursive splitting
- self-balancing

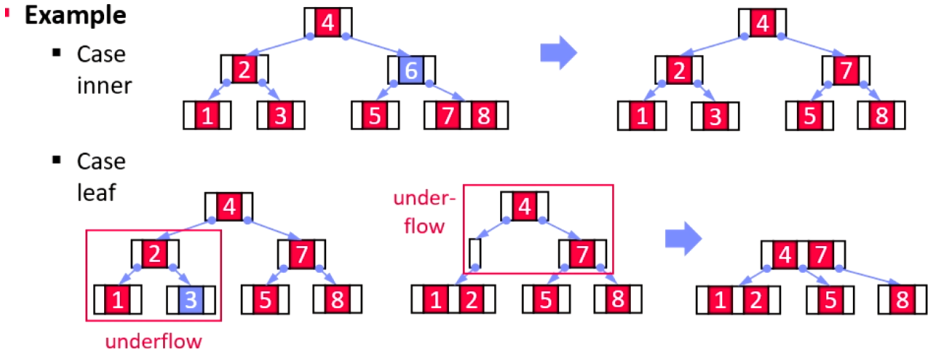


-
- Example

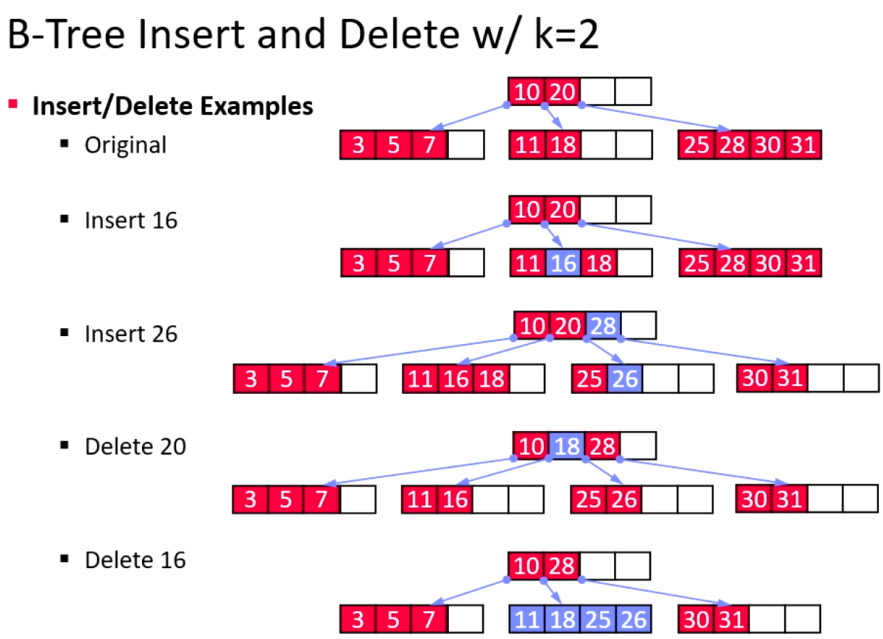


B-Tree Delete

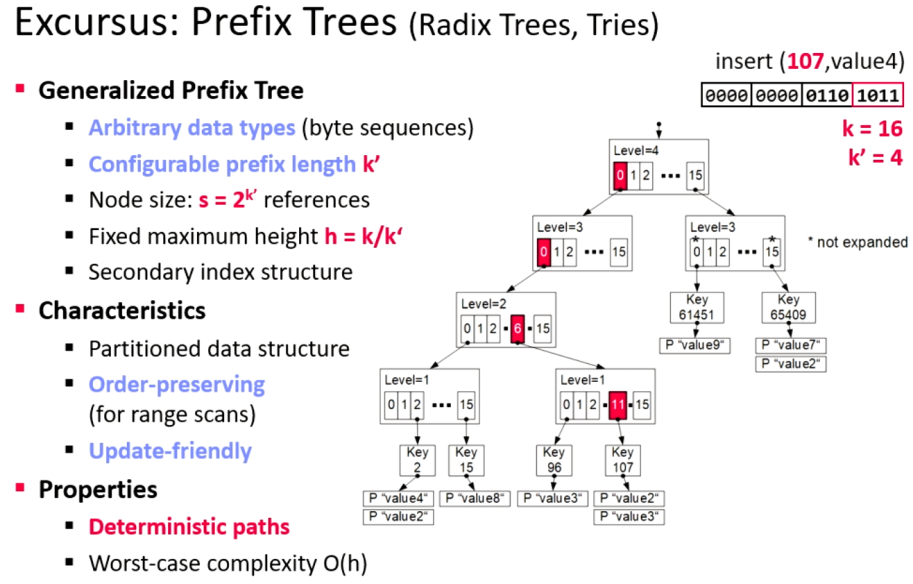
- deletion might cause underflow (<k entries)
  - underflow on inner node
    - ==> move entry from fullest successor (node below) into inner node
  - underflow on leaf node
    - ==> merge with sibling
- example



B-Tree Insert and Delete Example



Prefix Tree



Excursus: Learned Index Structures

- **A Case For Learned Index Structures**
  - Sorted data array, predict position of key
  - **Hierarchy of simple models** (stages models)
  - Tries to **approximate the CDF** similar to interpolation search (uniform data)

[Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, Neoklis Polyzotis: The Case for Learned Index Structures. **SIGMOD 2018**]