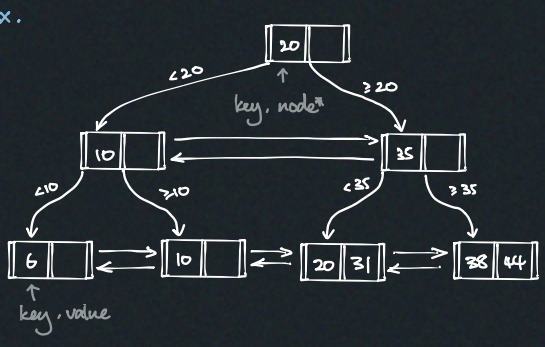
Lec 8 B+Tree

B-Tree, B+Tree, B*Tree, B^{link}-Tree, Bw-Tree, ...

B+ Tree

- Self-balancing, ordered
- Search, sequential access, insert, delete in $O(\log_f n)$ (f: fanout)
- Optimised for read/write to large blocks
- High fanout i.e. many children
- M-way search tree
- Perfectly balanced — each leaf at same level
- Each node other than root is half full $\frac{M}{2}-1 \le$ # keys $\le M-1$
- Siblings ptrs help access & concurrency — typically some 60% - 70% full
  - └ from B^{link}-Tree paper

Ex.



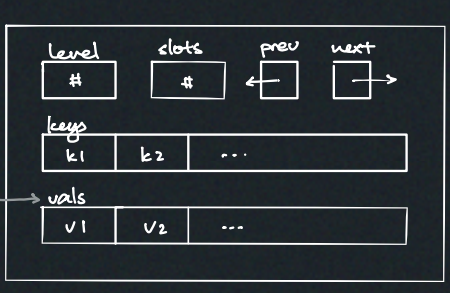
root
can have 1 entry

inner ← sorted (where NULL goes in ordering is impl dependent)

leaf ←
in DBs, leafs are pages

(inner, leaf: must be half full)

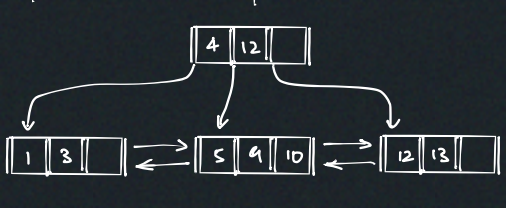
A node:



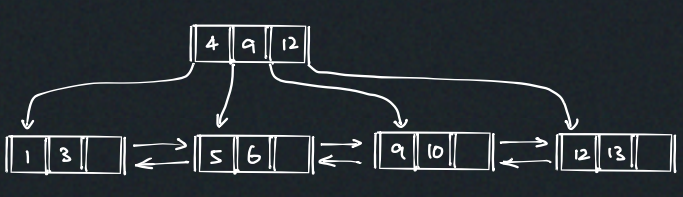
Algorithms

Insert

find node
if has free slot, add it
if node full, split



insert 6
↳



Delete

find node
remove it
if more than half full, done
else
  try grab things from siblings
  if fails merge with sibling
    - immediate nbor only
    - usually next works

if inner node underfilled, can try pull from root

Composit Index

key by multiple attributes, like tuple key
Just sort using a custom comparison func
Note efficient prefix search is possible

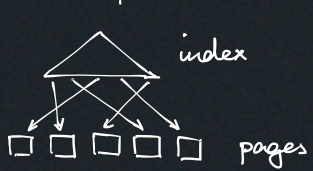
B-Tree with duplicate

→ keep page & slots id as part of the key
→ Overflow into linked list

Clustered Index

Clustered: pages pointed from leafs in same ordering as on heap
→ Tree traversal will access pointers sequential wrt address

Clustered



Non-clustered
Bad IO :(

Misc

Efficient B-Tree page sizes

HDD — 1M
SSD — 10K
Mem — 512B

Merge threshold — allow underfill, unbalanced, etc. depending on workflow

Var-len key

→ T-Tree puts pointers to keys
→ Var. len nodes
→ Padding

Key search within node

→ Binary search
→ Hardware: one instruction multiple checks
→ Interpolation search
  - └ approximate loc of key based on dist.

Prefix Compression

robbed robbing robot
↳ rob ← common prefix
bed bing ot

Duplicate key compression

$k_1 v_1$ $k_1 v_2$ $k_1 v_3$ $k_2 v_4$
↳ $k_1 v_1 v_2 v_3$ $k_2 v_4$

Suffix truncations

Store min characters needed on internal nodes to tell apart partiti

Pointer swizzling

Pin pages in mem
Temporily change disk ptr to mem ptr

Bulk insert

Sort then build B-Tree bottom up
More efficient than insert one by one

$B\varepsilon$-Tree

Write efficient
Each inner node keeps log