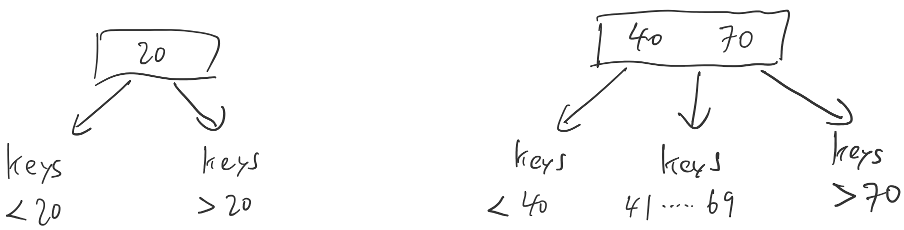
Lec17-B-trees

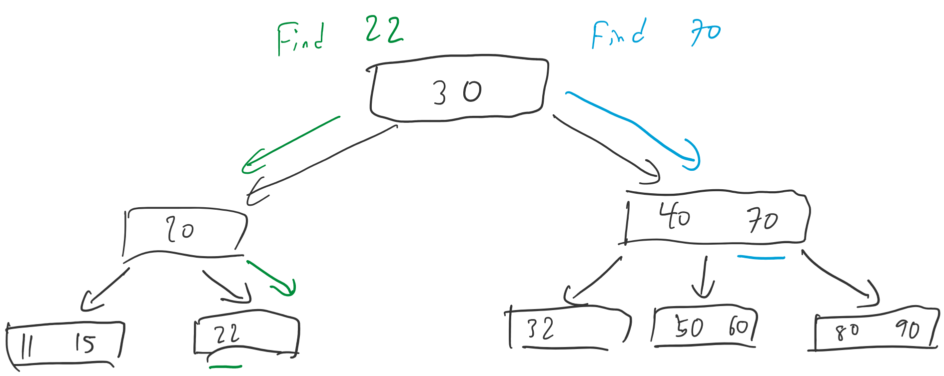
Tuesday, October 10, 2023 7:29 PM

We just saw one way to "balance" out operations in the splay tree.
Another alternative is to force exact balance by varying node sizes.

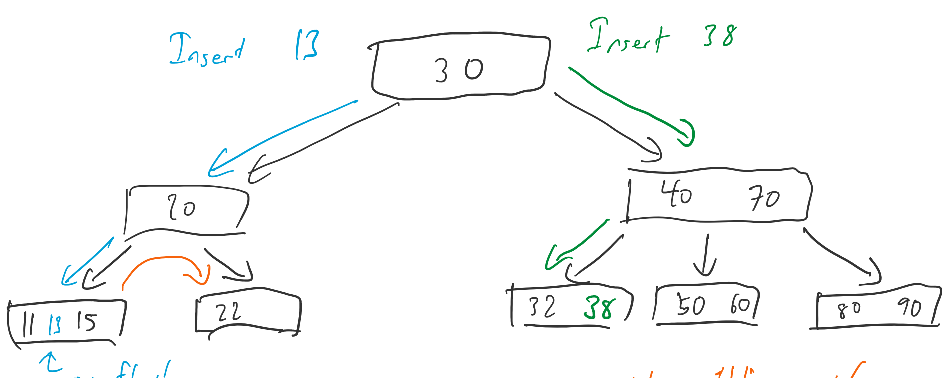
2, 3 - tree (later, a,b - tree)

- All leaves at same level
- Internal nodes have either $\underbrace{2 \text{ or } 3}_{}$ children.
  need respectively 1 or 2 keys in node

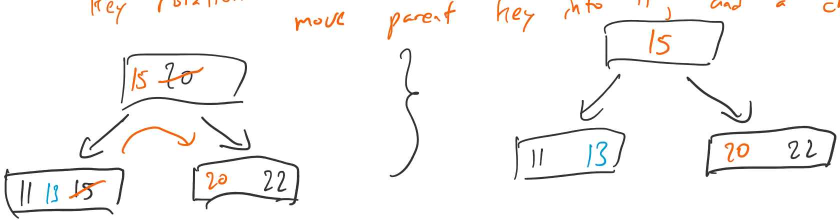




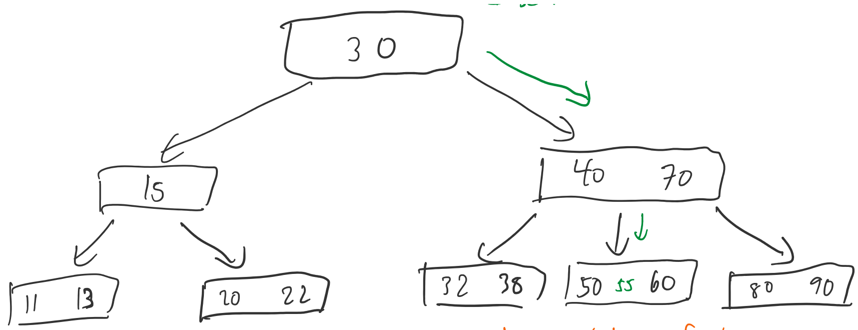
Standard BST walk down tree to find keys



key rotation - look for left or right sibling w/ space, move parent key into it, and a child into parent.

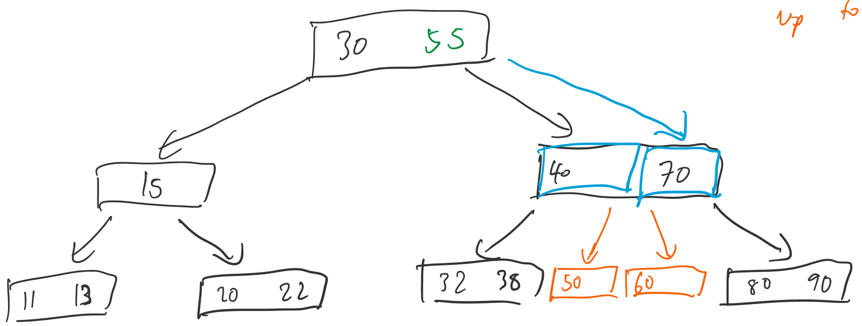


Insert 55



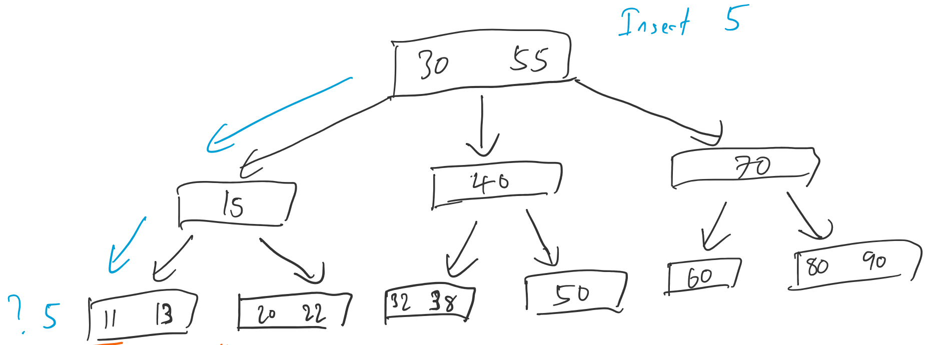
Key rotation failure

⇒ split node when node has 3 values & can't rotate. Push middle value up to parent.



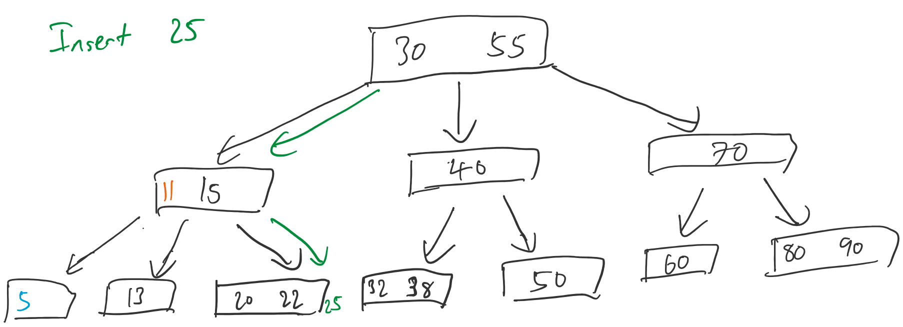
may need to recursively split

Insert 5

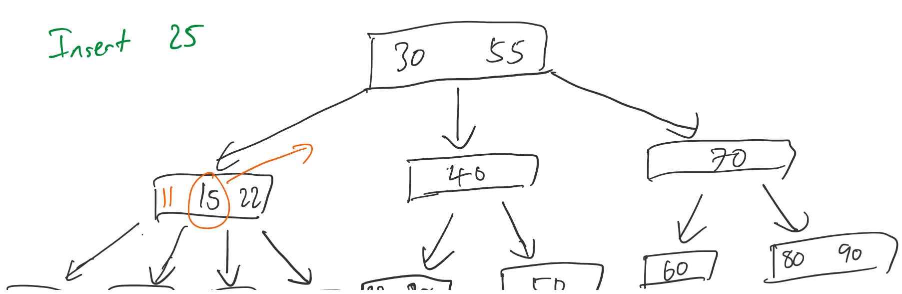


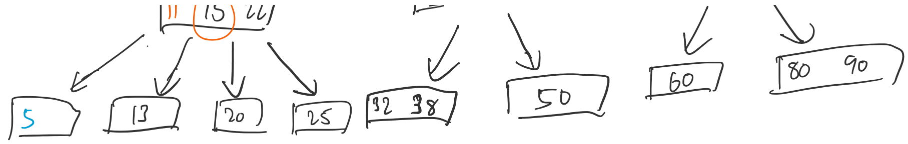
↳ push middle val up

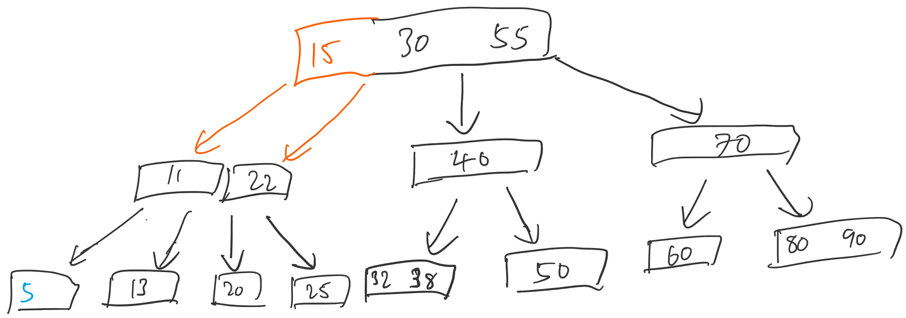
Insert 25

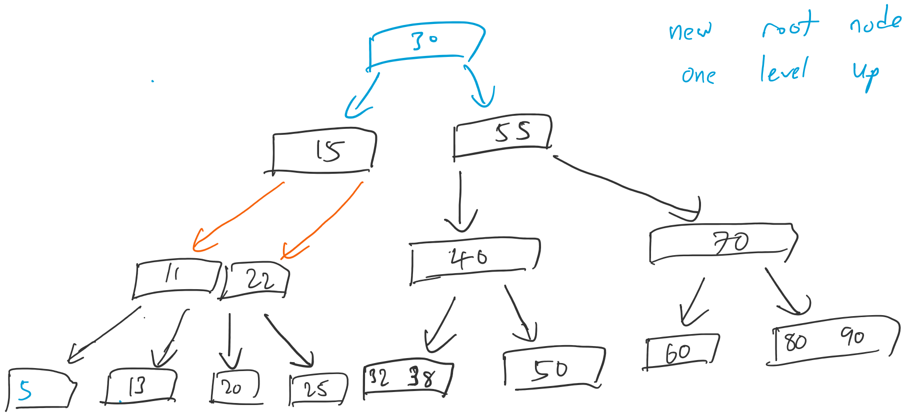


Insert 25









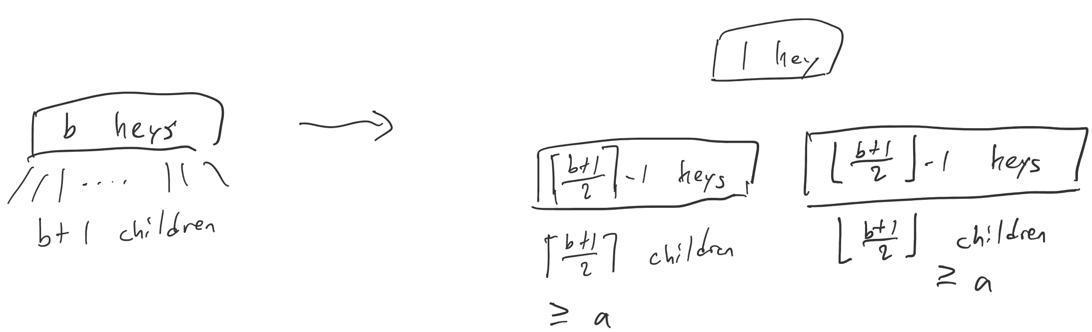
splitting process always keeps same tree height

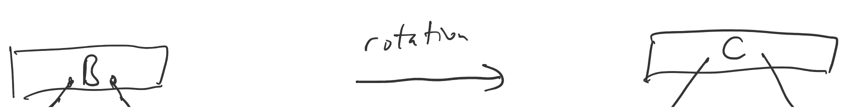
between

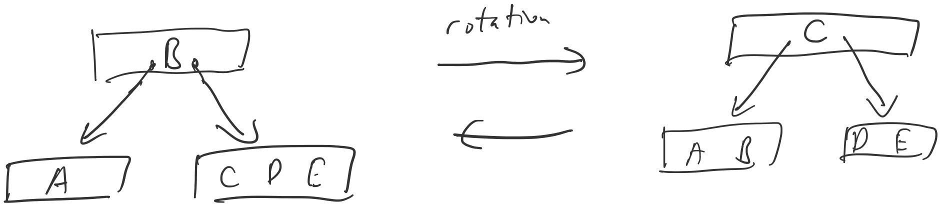
$\log_2 n$ & $\log_3 n$

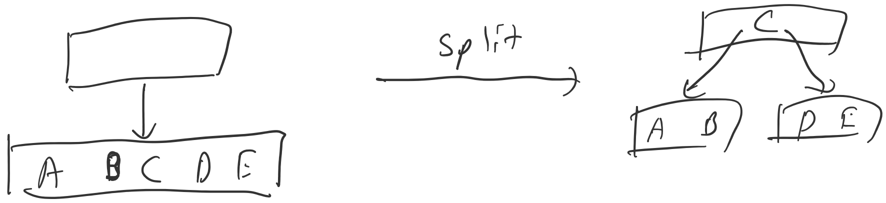
a, b - trees - generalization of 2, 3 trees

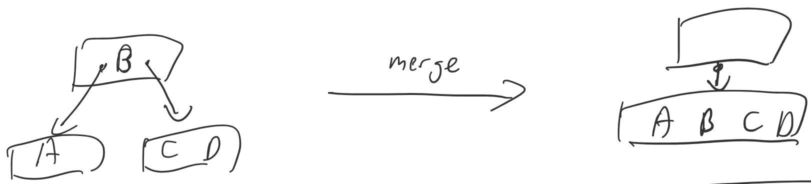
- all nodes (except root) have between a & b children
- root has b/t 2 & b children
- $a \geq 2$ (otherwise wouldn't branch)
- $b \geq 2a - 1$ (need enough children to make split work)











File system model

Page – contiguous block of data (e.g. 4096-byte chunk)

Probe – first access to a page (e.g. from mem to disk)

Time to probe is much larger than accessing data within a page

Cost model – $\underbrace{\text{minimize expensive probes}}_{\text{goal}}$

B-trees

A B-tree of order b is an a, b-tree with $\underbrace{b = 2a - 1}_{\text{choose largest allowed } a}$ ↳ usually chosen to match device characteristics / pages

Want large b if bringing node into memory is slow, but scanning in memory is fast.

Ex. B-tree of order 1023 has $a = 512$

$n = 10{,}000{,}000 \Rightarrow$ height $O(\log_a n) \approx 2.58 \Rightarrow$ read 3 blocks from disk.

Properties

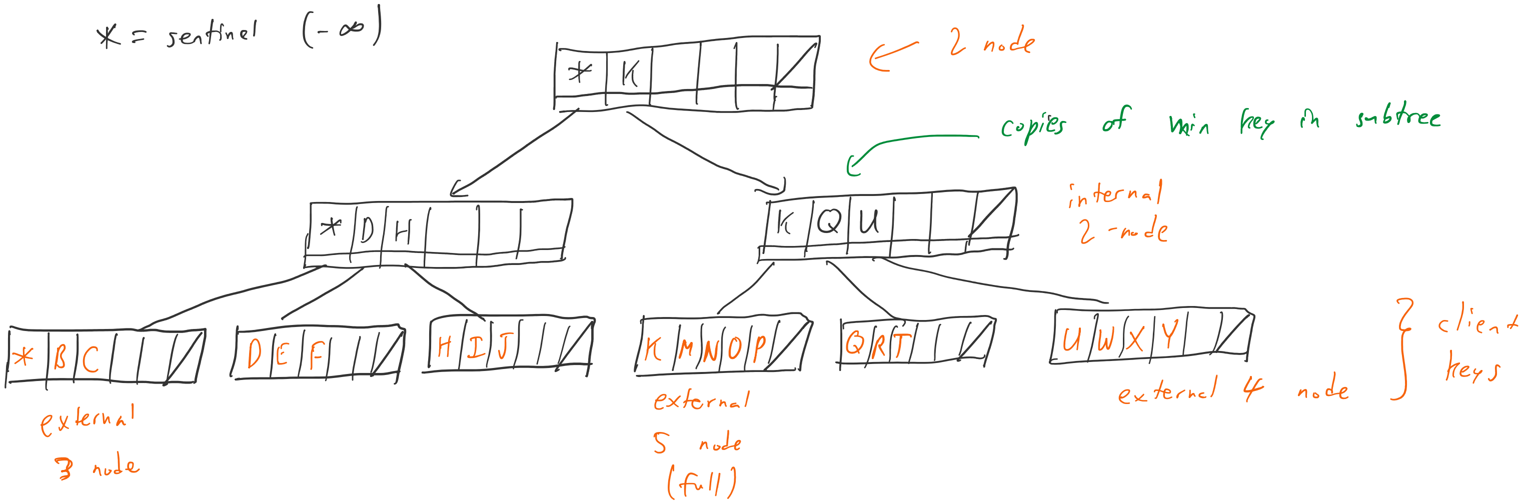
(a, b)-tree. Let $M = 2a = b + 1$. All nodes have between a & $M-1$ keys, except root which has at least 2.

All client keys are in leafs / external nodes.

Internal nodes contain copies of keys to guide search

Ex $M = 6$

$*$ = sentinel $(-\infty)$



Searching as usual in (a,b)-tree.
 └ follow the maximum link that is $\le$ key.

Insertion splits nodes of size M recursively upward
 └ never need to rotate keys, though you can.

Deletion may require rotating keys & merging

Balance: # levels $= O(\log n)$

$$\log_{M-1} n \le \#\text{ levels} \le \log_{\frac{M}{2}} n$$ because every internal non-root node has b/t $\frac{M}{2}$ & $M-1$ links.

$\Rightarrow$ insert / search need b/t $\log_{M-1} n$ and $\log_{\frac{M}{2}} n$ probes.

In practice, # probes is very small
 e.g. $M = 1024$, $n = 62$ billion $\Rightarrow$ $\log_{\frac{M}{2}} n \le 4$.

Aside: linearly searching through b keys is $O(1)$ if b constant, but might be slow.

However, can use other balanced tree (e.g. splay) tree at each node for efficient searching