Lec18-B-trees

Monday, October 7, 2024 10:00 PM

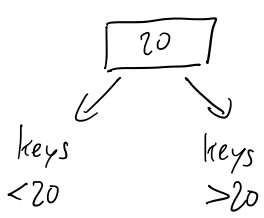
We just saw one way to "balance" out operations in the splay tree.

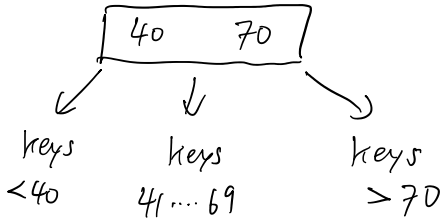
Another alternative is to force exact balance by varying node sizes.

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

- All leaves at same level
- Internal nodes have either 2 or 3 children.

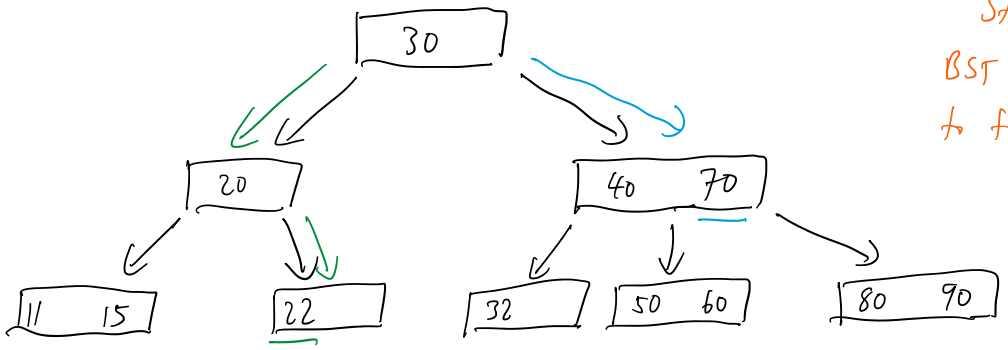
need respectively 1 or 2 keys in node.



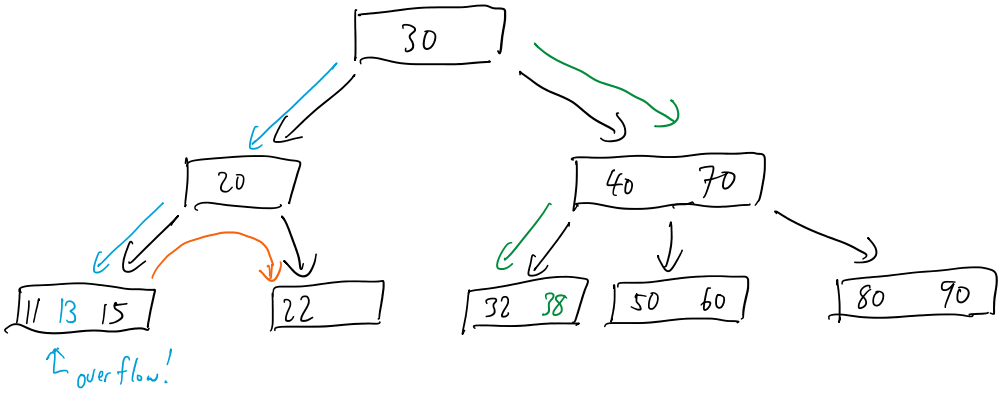


Find 22 Find 70

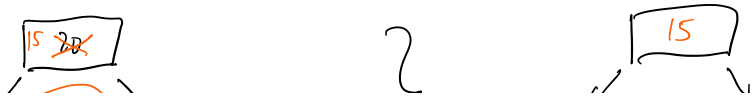
Standard BST walk to find keys

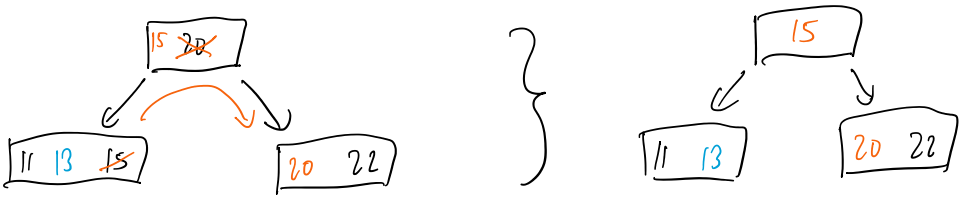


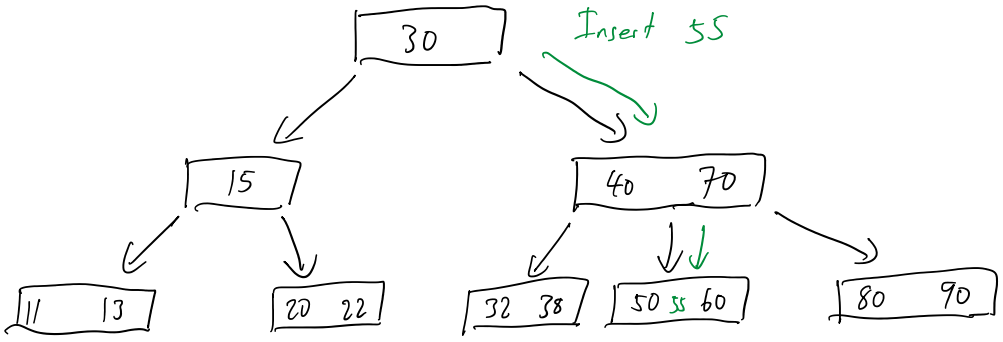
Insert 13 Insert 38



key rotation – look for left or right sibling with space, move parent key into it, and a child key into parent

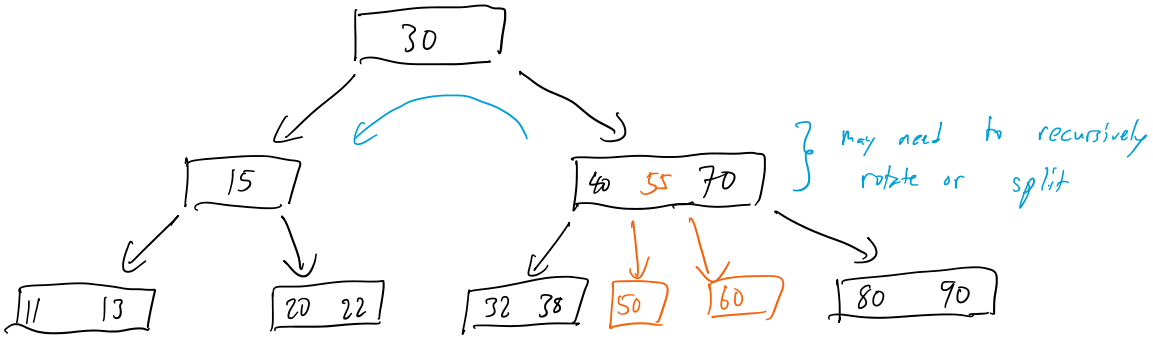


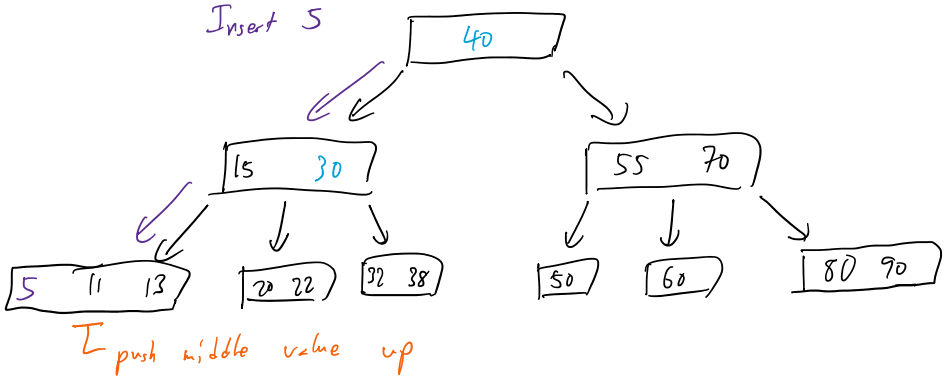




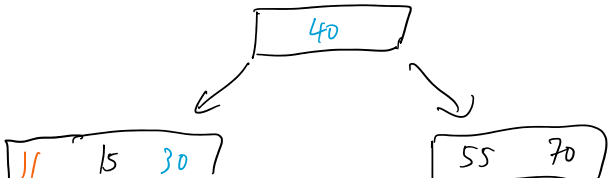
Key rotation failure

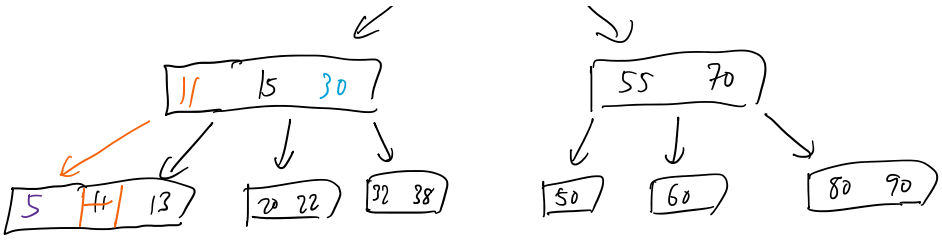
=> split node when node has 3 values & can't rotate, push middle value up to parent

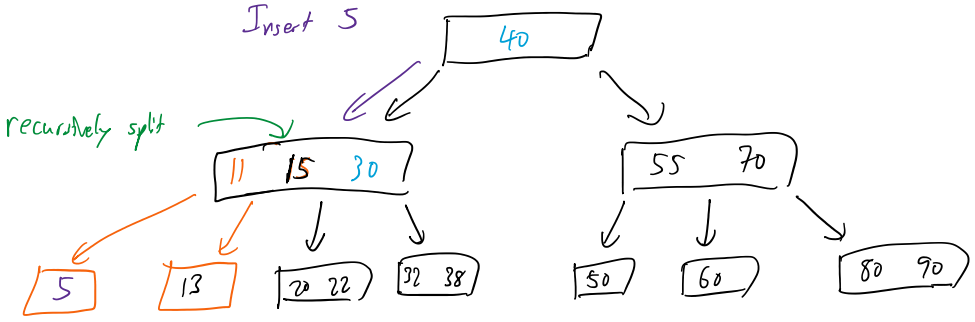


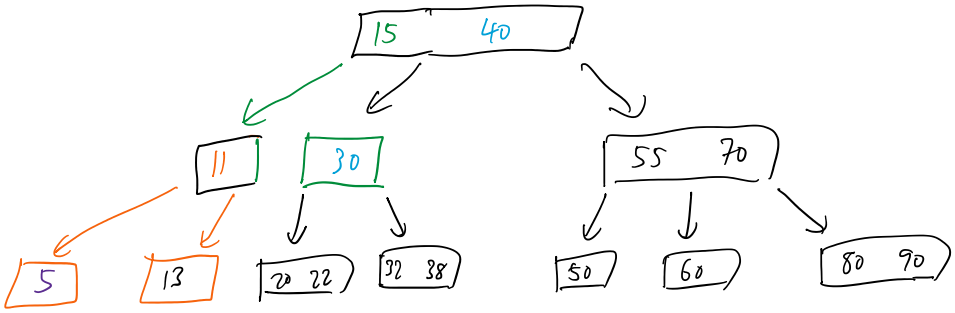


=> push middle value up

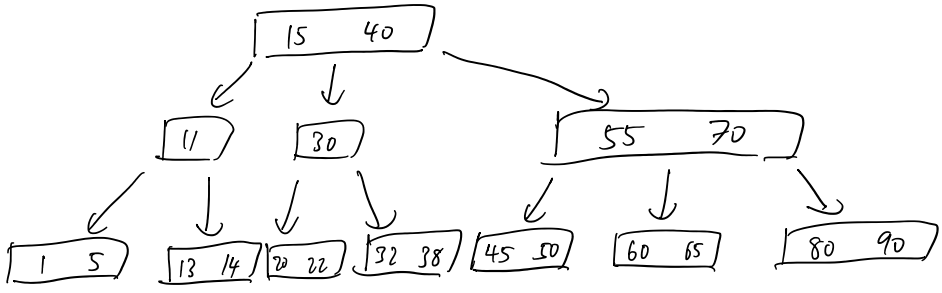




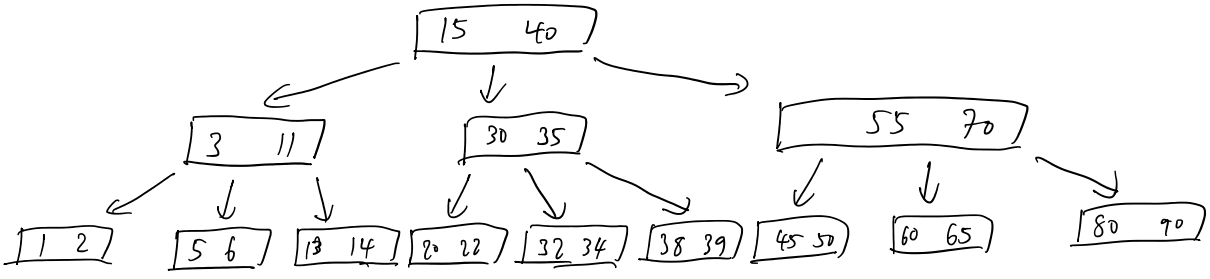




Insert 1, 14, 45, 65

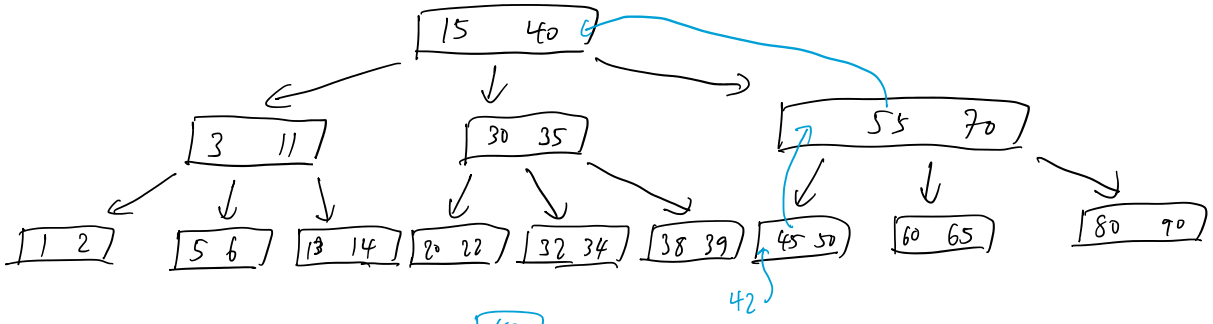


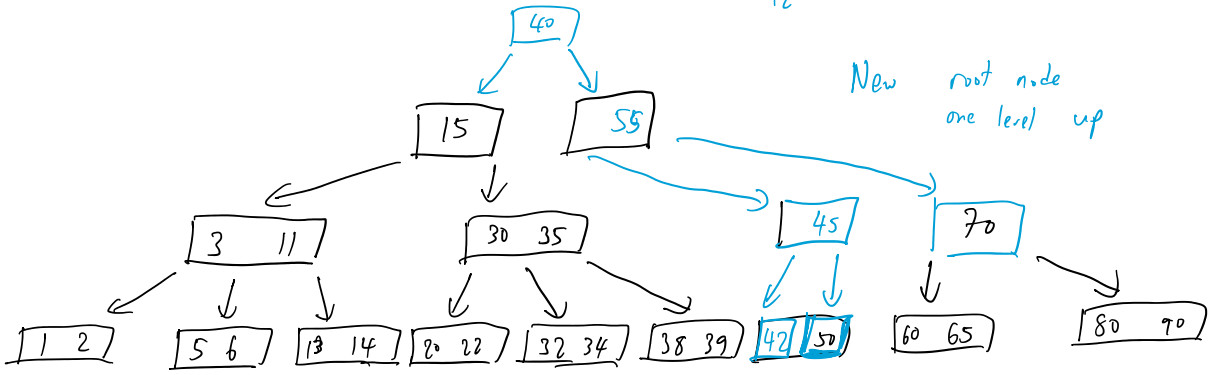
Insert 3, 2, 6, 35, 34, 39



Insert 42

Insert 42





New root node one level up

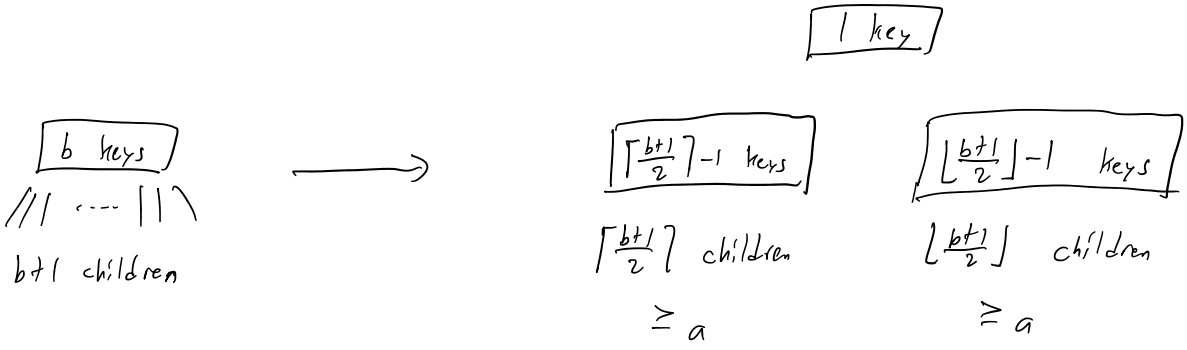
Splitting process always keeps same tree height for levels because we add on top.

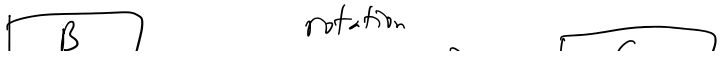
tree height between $\log_2 n$ and $\log_3 n$.

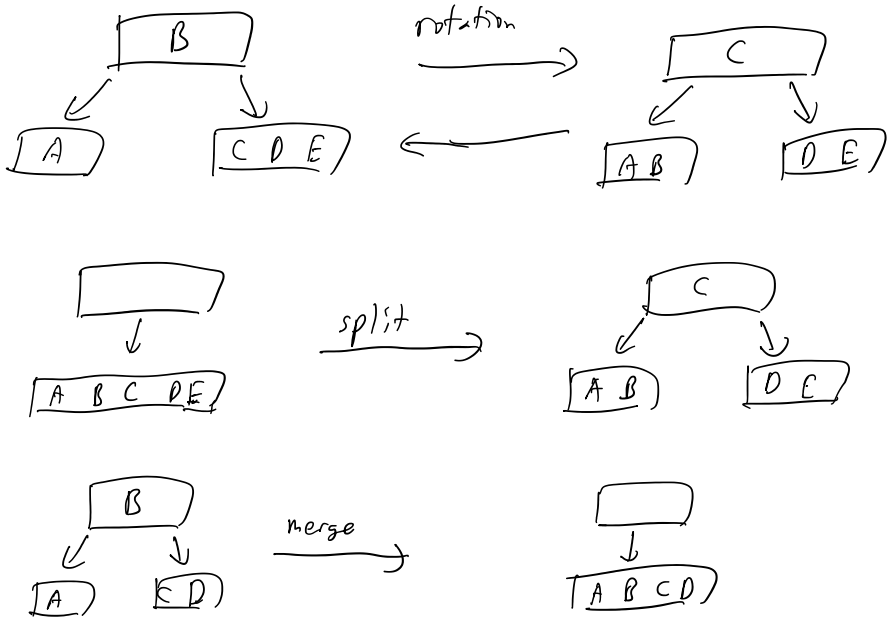
---

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

- all nodes (except root) have between a & b children
- root has between 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 disk to mem)

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

Cost model - minimize expensive probes

goal

B+-trees

usually chosen to match device characteristics / pages

A B-tree of order b is an a, b-tree with $b = 2a - 1$

Choose largest allowed a.

Want large b if bringing nodes 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

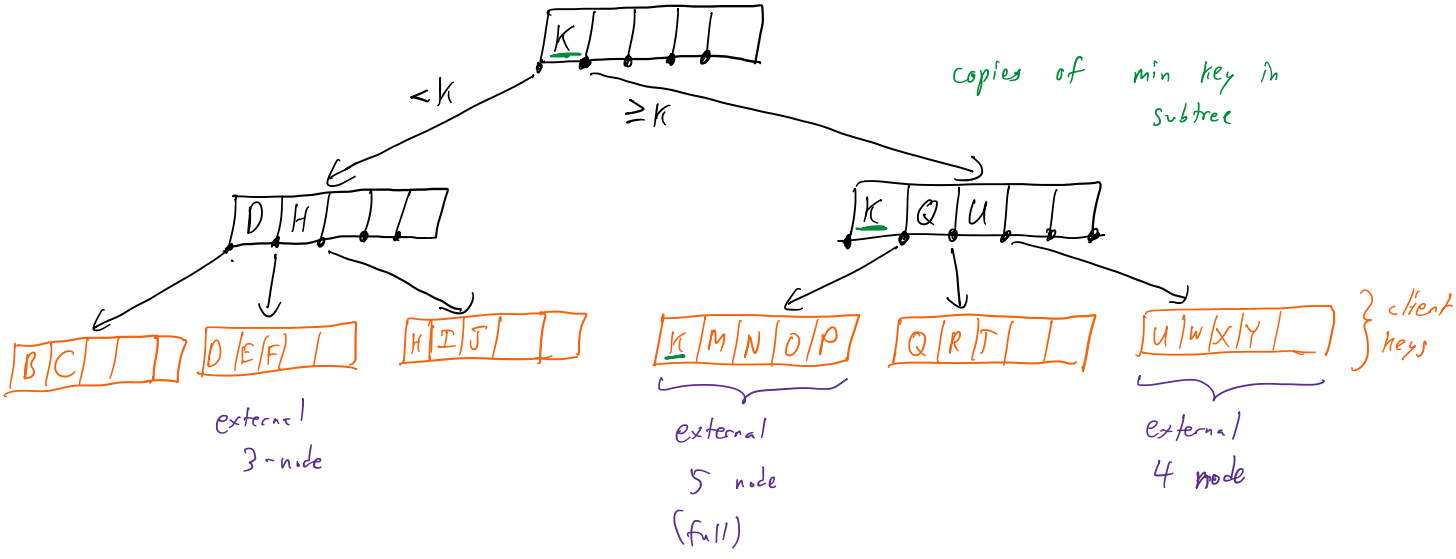
Properties

(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 } B+ tree instead of ordinary B tree

Internal nodes contain copies of keys to guide search

Ex. $M = 6$ $a = 3$



Searching as usual in (a, b)-tree
- follow max link that is ≤ 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 $= O(\log n)$

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

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

In practice # probes is very small.

e.g. $M = 1024$, $n = 62$ billion $\Rightarrow \log_{\frac{M}{2}} n \leq 4$.

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 more efficient searching.