ID1020 2018-09-24 #9

Balanced search trees

- 2-3 search trees
- red-black BSTs (left-leaning)
- B-trees

Challenge. Guarantee performance

2-3 trees, allow 1 or 2 keys per node

2-node: one key, two children

3-node: two keys, three children

Symmetric order
Perfect balance

Search
- Compare search key against key in node.
- Find interval containing search key.
- Follow associated link (recursively)

Insertion into a 2-node at bottom
- Add new key to 2-node to create 3-node

Insertion into a 3-node at bottom
- Add new key to 3-node to create temporary 4-node
- Move middle key in 4-node into parent.
- Repeat up the tree, as necessary.
- If you reach the root and it's a 4-node, split it into three 2-nodes.

Splitting a 4-node is a local transformation: constant #operations
Maintains symmetric order and perfect balance since each transformation maintains symmetric order and perfect balance

Perfect balance

tree height: worst case $\lg N$ (all 2-nodes)
best case $\log_3 N$ (all 3-nodes)

$c \lg N$ for all operations

Direct implementation complicated
- Maintaining multiple node types
- Multiple compares to move down tree
- Need to move back up the tree to split 4-nodes
- Large number of cases for splitting

Implement 2-3 trees with binary trees

Red black trees: regular BST with red "glue" links
- Widely used in practice
- Arbitrary restriction: red links lean left

1-1 correspondence between 2-3 and LLRB

A BST such that:
- No node has two red links connected to it.
- Every path from root to null link has same #black links
- Red links lean left.

"Perfect black balance"

Search is the same as for BST (ignore color) (faster because of balance)

Insertion

During internal operations, maintain
- Symmetric order
- Perfect black balance

Rotate and flip color

Insert into 2-node at bottom
- Do standard BST insert; color new link red.
- If new red link is a right link, rotate left.

Insert into 3-node at bottom
- Do standard BST insert; color new link red ← maintain symmetric order
- Rotate to balance the 4-node (if needed)
- Flip colors to pass red link up one level.
- Rotate to make lean left (if needed)
- Repeat case 1 or case 2 up the tree (if needed)

← fix color invariants

Java implementation, same for all cases
- Right child red, left child black: rotate left
- Left child, left-left grandchild red: rotate right
- Both children red: flip colors

Height of tree is $\leq 2 \lg N$ in the worst case

Xerox Parc invented a lot of things
- GUI
- Ethernet
- Laser printing
- Bitmapped display
- WYSIWYG text editor
- Red-Black trees

Phone company sued database provider for exceeding height limit of 80 triggering error-recovery process
Hibbard deletion was the problem, not RBBST
Expert witness: "If implemented properly, the height of a red-black BST with N keys is at most $2\lg N$."

B-trees (for data that cannot fit in primary memory)

Page - continous block of data
Probe - first page access (from main memory)

B-tree Generalize 2-3 trees by allowing up to M-1 key-link pairs per node.

A search/insertion in a B-tree of order M with N keys requires between $\log_{M-1} N$ and $\log_{M/2} N$ probes.

In practice number of probes is at most 4. ← M=1024; N=62 bilion $\log_{M/2} N \leq 4$

Red-Black are used as system symbol tables-

B-tree variants B+tree, B*tree, B#-tree

B-trees are widely used for file systems and databases.