Design and Analysis
of Algorithms I

Data Structures

Red-Black Trees

Binary Search Tree Structure

-- exactly one node per key

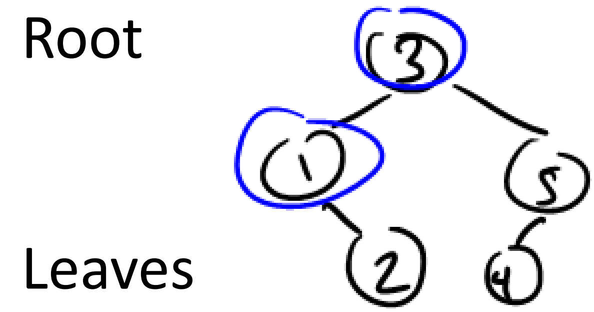
-- most basic version :

each node has

-- left child pointer

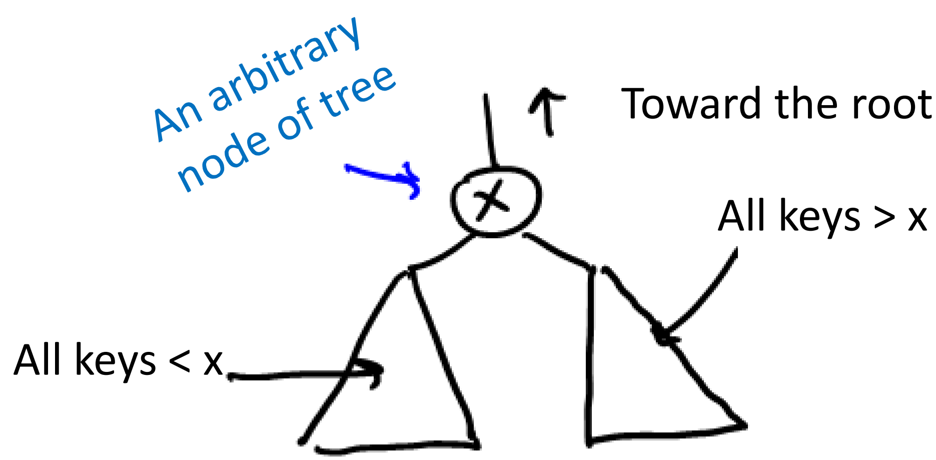
-- right child pointer

-- parent pointer



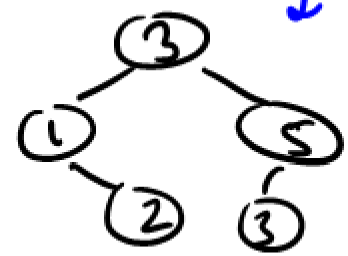
SEARCH TREE PROPERTY :

(should hold at every node of the search tree)



The Height of a BST

Height = 2



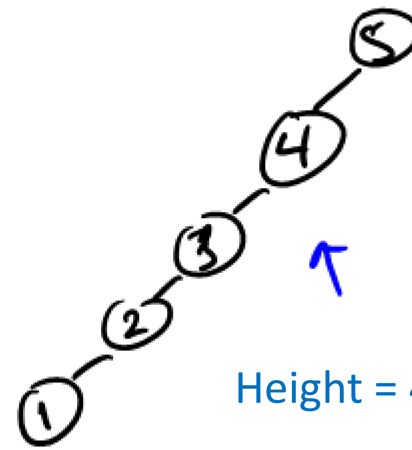
Note : many possible trees for a set of keys.

(aka depth) longest root-leaf path

Note : height could be anywhere from $\boxed{\sim \log_2 n}$ to $\boxed{\sim n}$

Worst case, a chain

Best case, perfectly balanced



Height = 4

Balanced Search Trees

Idea : ensure that height is always O(log(n)) [best possible]
⇒Search / Insert / Delete / Min / Max / Pred / Succ will then run in O(log(n)) time   [n = # of keys in tree]

Example : red-black trees [Bayes '72, Guibas-Sedgewick '78]

[see also AUL trees, splay trees, B trees]

Red-Black Invariants

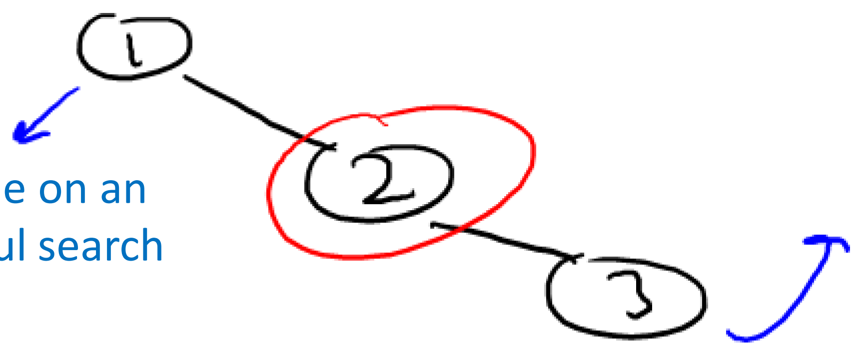
1. Each node red or black
2. Root is black
3. No 2 reds in a row
   [red node => only black children]
4. Every root-NULL path has same number of black nodes

Like in an
unsuccessful search

Example #1

Claim : a chain of length 3 cannot be a red-black tree

Proof

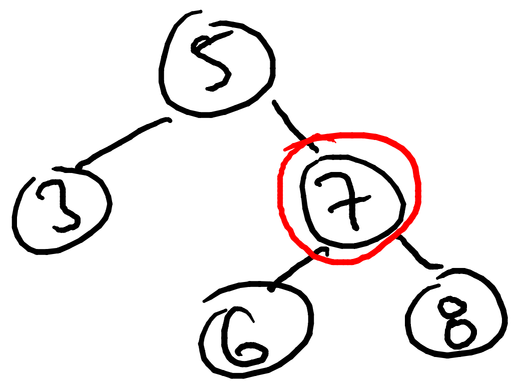


1 black node on an unsuccessful search for 0

2 black nodes on an unsuccessful search for 4

Tim Roughgarden

Example #2

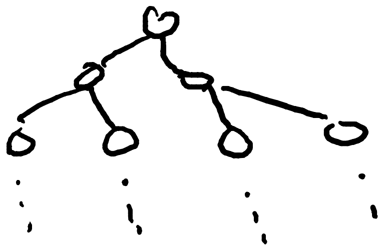


Height Guarantee

Claim : every red-black tree with n nodes has height $\leq 2\log_2(n+1)$

Proof : Observation : if every root-NUL path has >= k nodes, then tree includes (at the top) a perfectly balanced search tree of depth k-1.

=> Size n of the tree must Be at least $2^k - 1$



[k = 3]

Height Guarantee (con'd)

Story so far : size $n \geq 2^k - 1$, where k = minimum # of nodes on root – NULL path

=> $k \leq log_2(n+1)$

Thus : in a red-black tree with n nodes, there is a root-NULL path with at most $\log_2(n+1)$ black nodes.

By 4th Invariant : every root-NULL path has $\leq \log_2(n+1)$ black nodes

By 3rd Invariant : every root-NULL path has $\leq 2\log_2(n+1)$ total nodes.

Q.E.D.

Which of the search tree operations have to be re-implemented so that the Red-Black invariants are maintained?

- Search
- Delete
- Insert and Delete
- None of the above