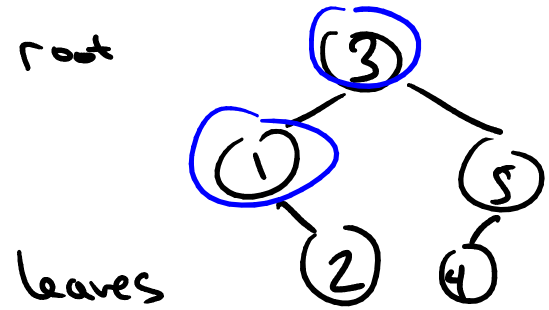
Design and Analysis of Algorithms I

Data Structures

Red-Black Trees

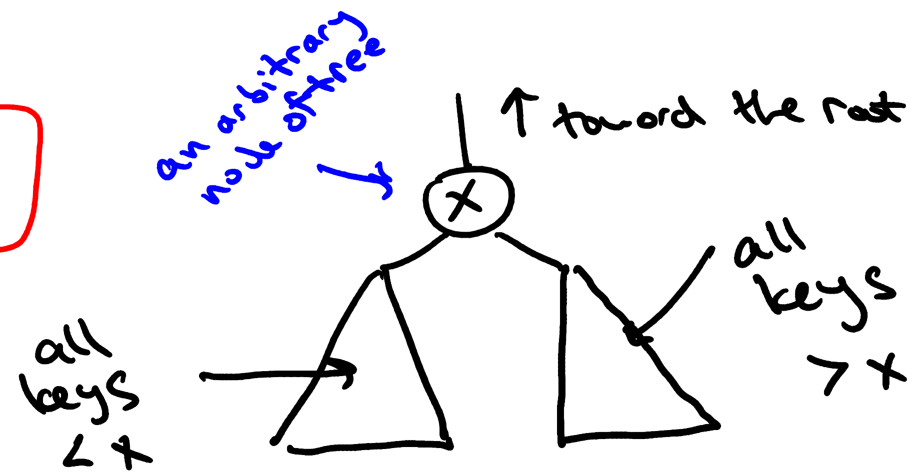
Binary Search Tree Structure

- exactly one node per key
- most basic version: each node has
  - left child pointer
  - right child pointer
  - parent pointer



SEARCH TREE PROPERTY:

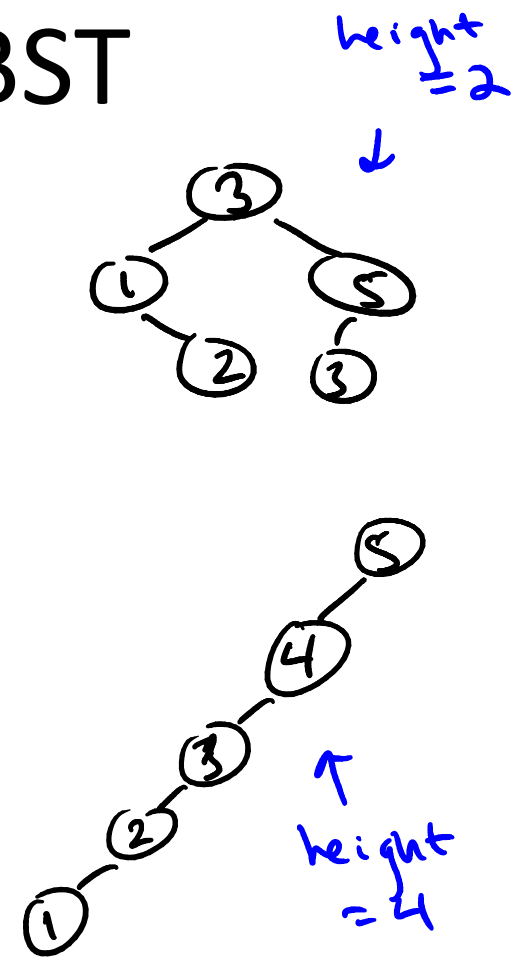
(should hold at every node of the search tree)



The Height of a BST

Note: many possible trees for a set of keys.

Note: height (aka depth) longest root-leaf path could be anywhere from $\approx \log_2 n$ (best case, perfectly balanced) to $\approx n$. (worst case, a chain)



Balanced Search Trees

Idea: ensure that height 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 [Bayer '72, Guibas-Sedgewick '78]

[see also AVL 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]

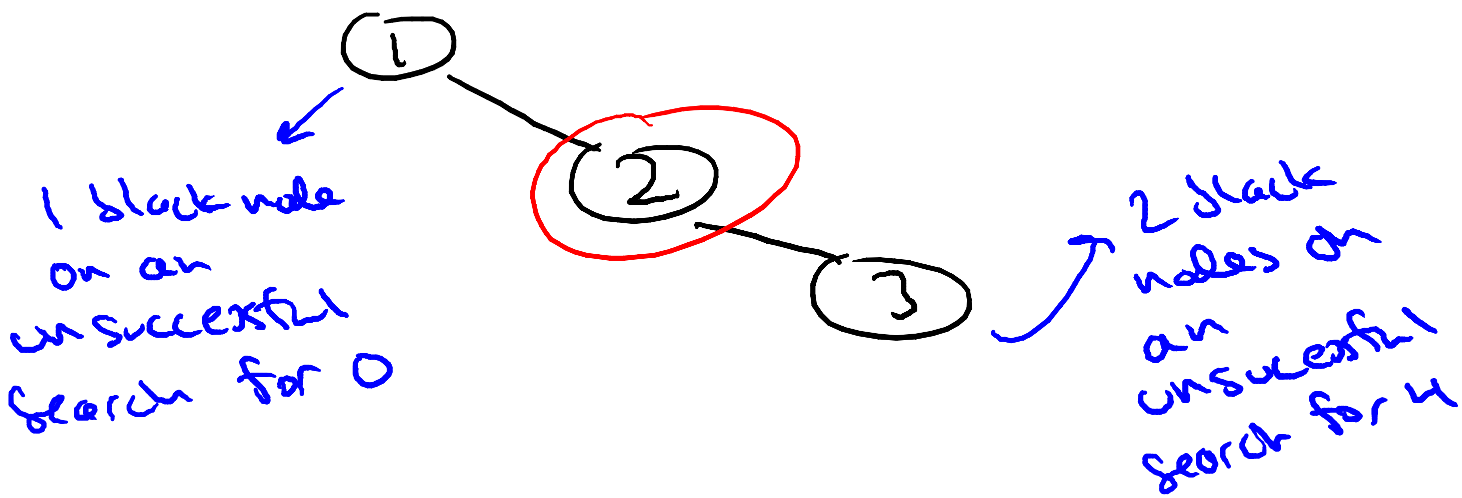
(4) every root-NULL path has same number of black nodes

like in an unsuccessful search

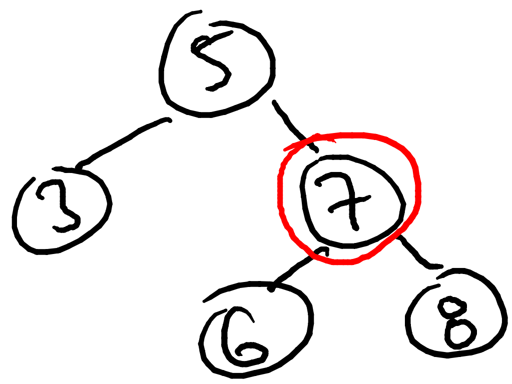
Example #1

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

Proof:



Example #2

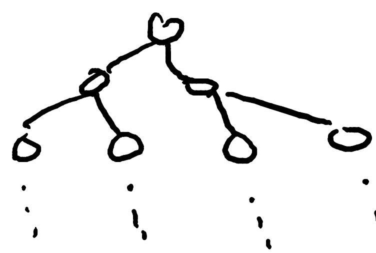


Height Guarantee

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

Proof: Observation: if every root-NULL path has $\geq k$ nodes, then tree includes (at the top) a perfectly balanced search tree of depth $k-1$

$\Rightarrow$ 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

$\Rightarrow k \leq \log_2(n+1)$

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