Algo

2/11/19

MT & Balanced Search Trees

Announcements

- Tues: Big O, D&C, MT, Proofs
- Sat: OH Olsson 120
- Batman problem is hard

Agenda

- substitution
- balanced search trees

Substitution

① $T(n) = 2T(\sqrt{n}) + \log_2(n)$ // how to turn this into recognizable MT form?

$aT(n/b) + n^c$ // sometimes look @ last value & sub it in

Trick: substitute

★ ② $n = 2^m$ // m comes from $\log_2(n)$, assuming $2^m = n$, cleverness #1

③ $m = \log_2 n$

substitute 2 & 3 into 1

$T(2^m) = 2T(\sqrt{2^m}) + m$

$= 2T((2^m)^{\frac{1}{2}}) + m$

$= 2T(2^{m/2}) + m$ substitute for this to look like MT

★ $S(m) = T(2^m)$ ← // new function to represent 2^m, cleverness #2

$S(m/2) = T(2^{m/2})$ ...

$S(m) = 2S(m/2) + m$ // in MT form → $a = 2$, $b = 2$, $c = 1$

$aT(n/b) + n^c$ //MT

$S(m)$ $S(m/2)$

$S(m) = T(2^m)$ $S(m/2) = T(2^{m/2})$

Using MT

$\log_2 2 = 1 = 1$

ease 2

$S(m) = \Theta(m' \log_2 m)$

↓

$T(n) = O(\log_2 n)(\log_2(\log_2 n))$

$T(n) = T(2^m)$

MT

$$T(n) = \begin{cases} aT(n/2) + n^c & \text{if } n > 2 \\ d & \text{if } n = 1 \end{cases}$$

① $\log_b a > c$ $T(n) = O(n^{\log_b a})$

② $\log_b a < c$ $T(n) = O(n^c)$

③ $\log_b a = c$ $T(n) = O(n^c \log_b n)$

Exam - FEB. 25

- Lectures 1-8
- closed book
- 8.5 x 11 cheatsheet 1 side
- Review session: Sat & example problems
- 7 questions, free-form answers

Binary Heaps

- not cache friendly

2-3 Tree // similar to a red-black tree

- allow 1 or 2 keys per node
  - 2-node: 1 key, 2 children
  - 3-node: 2 keys, 3 children
- in-order traversal yields keys in ascending order // symmetric order
- every path from leaf to root is same length

→ search: compare left & right

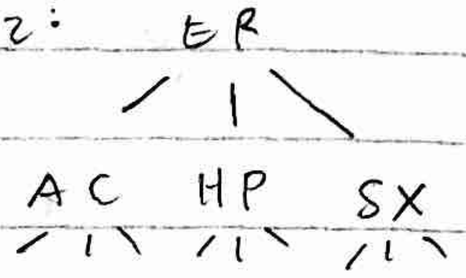
→ insert:

↳ 2-node: if 1 node, insert as key, if 2 node, put as child

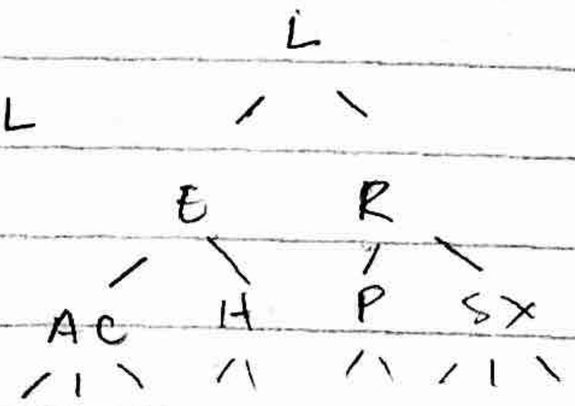
↳ 3-node: split 4-node, move middle key up to node

↳ if @ root, make the middle the new root

Quiz:



insert L

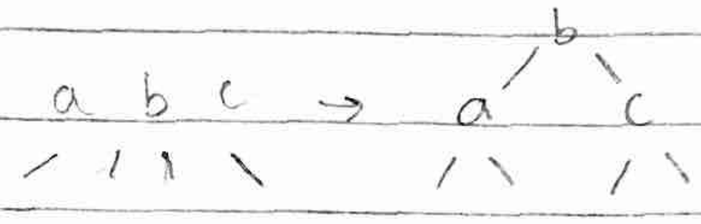


Invariants

- symmetric order & perfect balance
- each transformation must maintain these properties

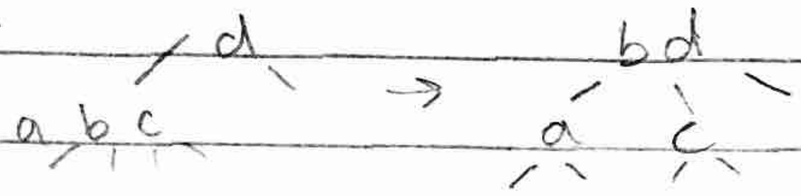
cases

root

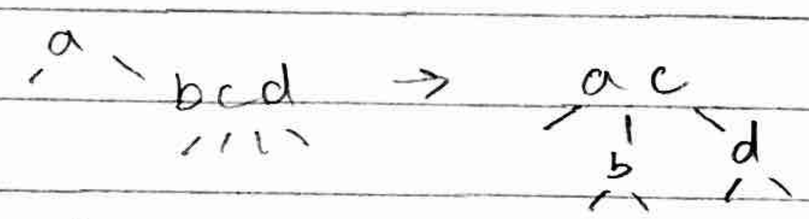


parent is 2-node

· left

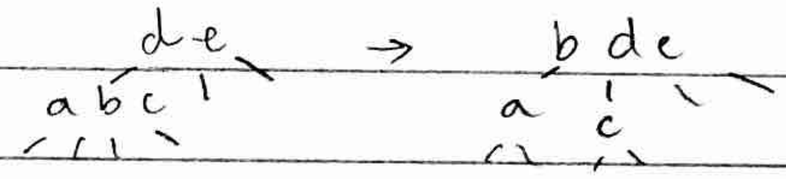


· right

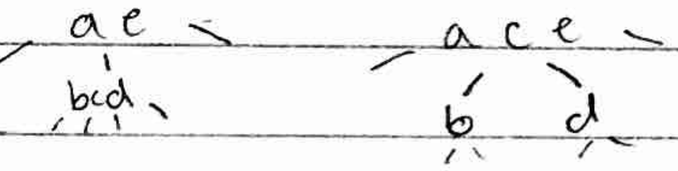


parent is 3-node

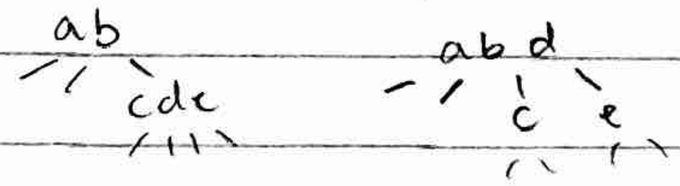
· left



· right



· middle



Worst-case

· all are 2 nodes $\log_2 N$

Best-case

· all are 3 nodes $\log_3 N$

} guarenteed log search time

→ as opposed to BST, worst case is a linear tree

Disadvantages

· cumbersome, lots of nodes

Implementation

1. regular BST
2. regular BST with glue nodes
★ 3. regular BST with red "glue" links
   ↳ red links have to be left-leaning

B-Tree

- generalize 2-3 trees by allowing up to M-1 key per node
  - $\geq$ 2 key-link pairs @ root
  - $\geq$ M/2 key-link pairs in other nodes
  - external node contain client keys
  - internal nodes contain copies of keys to guide search

http://algs4.cs.princeton.edu // textbook