Advanced Data Structures

(a,b)-trees
&
Bit vectors

Recap:

Last time: B-trees, &
external memory model

This time: Back to "normal"
analysis

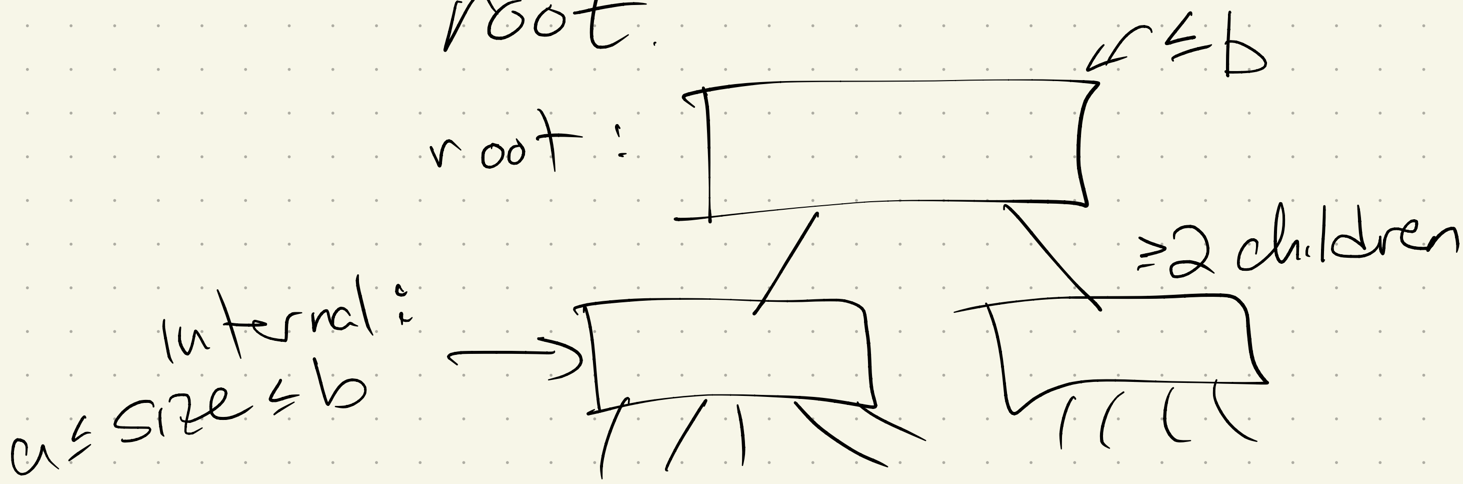
(External coming again
soon...)

Let a, b be constants, with

$$2a < b.$$

An (a,b)-tree is similar to a B-tree:

- the root has between 2 & b children
- every internal node has between a & b children
- all leaves have the same distance to the root.



[So B-trees $\approx$ $(B/2, B)$-trees in this notation] (a, b)-trees:

Most of the implementation & analysis is the same as for B-trees!

Since $2a < b$, still can use amortized accounting method.

Result: $\frac{\log b}{\log a} \cdot \log n$

$= \log_a b \cdot \log n$

Neat connection to red-black trees (if you've seen them):

<u>Dfn</u>: A red-black tree is a BST, where each node is red or black, with the following structure:

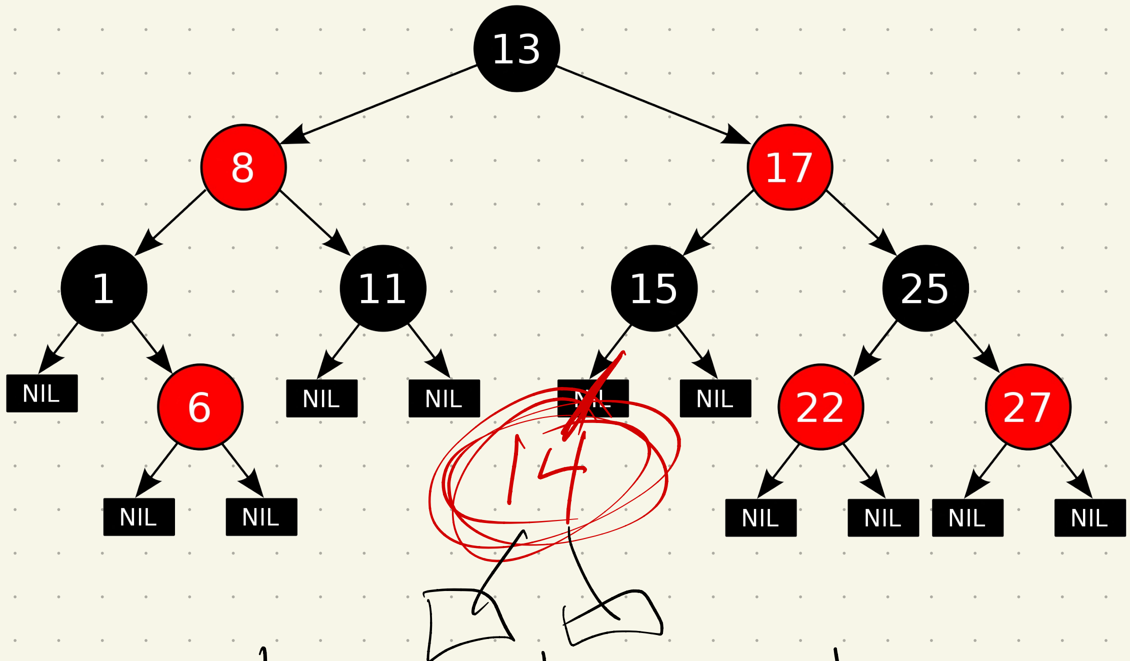
1: the root is black

2: Every "NULL" leaf is black

3: The children of a red node are black

4: All leaves have the same "black depth": # of node ancestors colored black -1.

Picture:



insert: color red (b/c keeps 4th & 2nd property)

But: 3rd may be violated!

Solution: rotations!

Result: $\log_2 n \le \text{height} \le 2\log_2 n$

Next: Cool connection

(see slides in link...)

Next data structure:

What if we restrict inputs?

Goal: Have a bounded set of possible elements, & want to store which ones are in my set.

ie: subset of 32-bit integer

or list of names (all ≤ 30 chars)

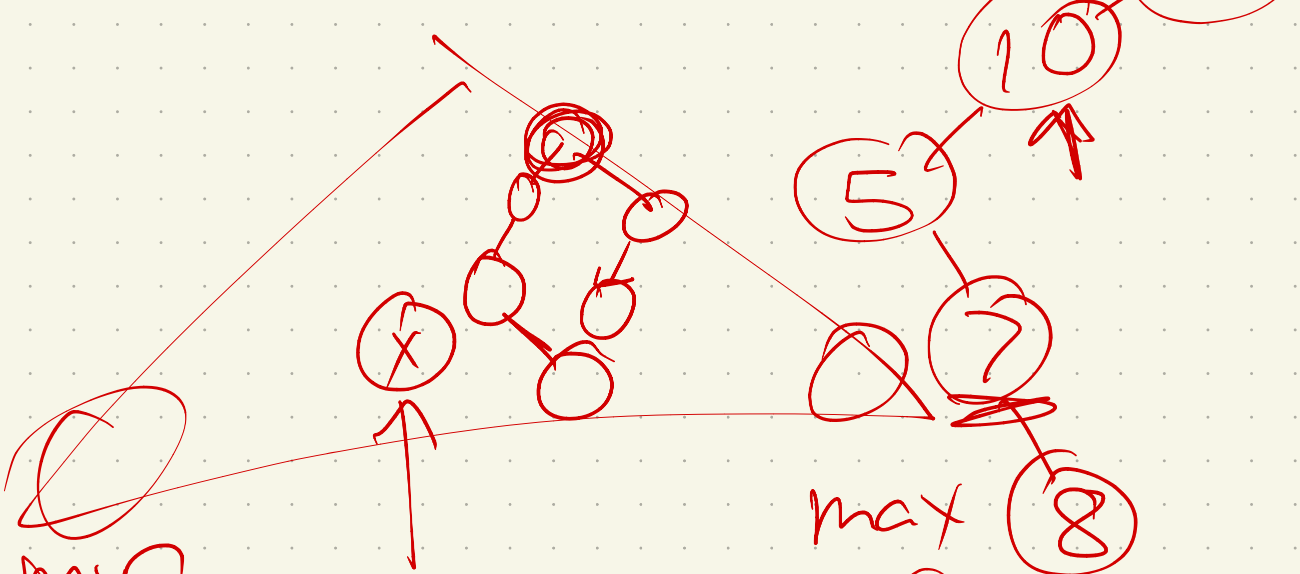
Operations

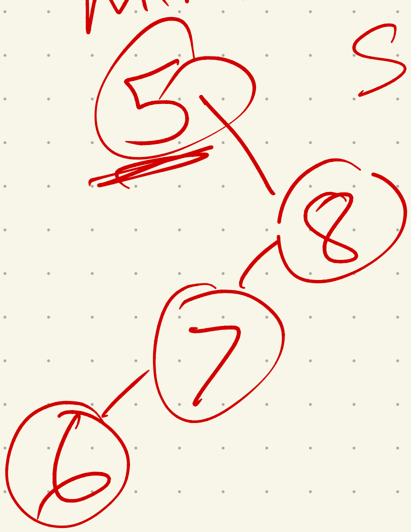
- insert(x)
- find (x)
- delete(x)
- max / min
- successor(x)
- predecessor (x)

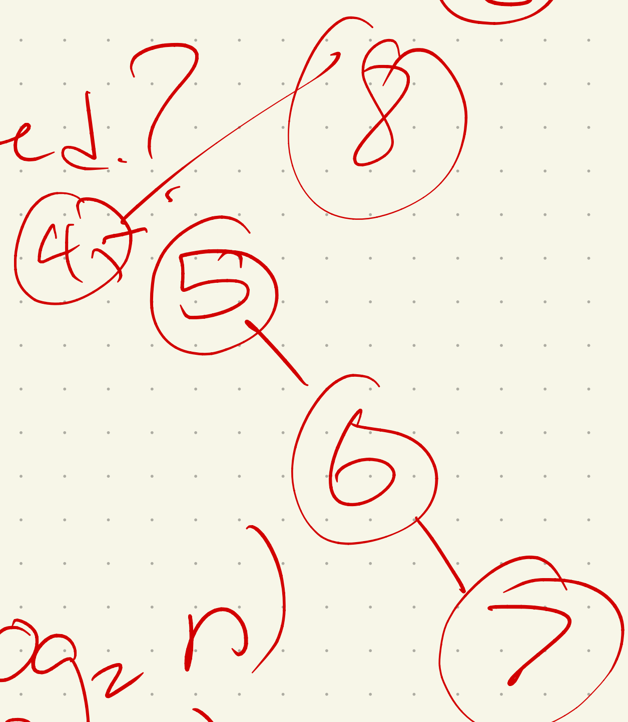
<u>Note</u>: BSTs can do all of this!

How?

insert, delete & find - obvious







<u>Runtime</u>: $O(\log_2 n)$

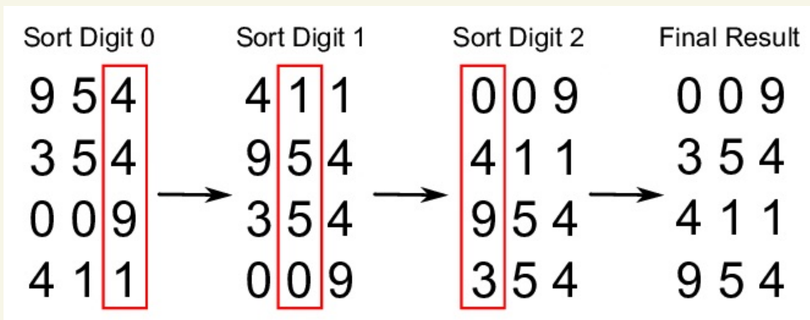
(in balanced BST)

Better approach:

Use bounded set!

Prior example: Radix Sort:

n elements, from 1 to k



0 1 ... 9

Each can be written using $\log k$ bits.

Runtime: $\log_2 k \, (n+k)$

Here: Values $1 .. U$, so size to store is $w =$

Bitvector approach: length U vector of bits

13 $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ U

B: `110110001011101111000100110101011110011011110111 1`

$B[i] = 1$ means i is in set

insert
delete
lookup
$\Big\}$ $O(1)$

min / max: $O(U)$

pred / successor: $O(U)$

`0000000000000000000000000000100000000000000000000000000000`

1

Tiered Bitvector:

Put a summary on top of the vector. u/B

OR the bits

1	0	1	0	1	1	0	0
00100010	00000000	00011000	00000000	00000100	11110111	00000000	00000000

B

How to search/update:

succ: check for next value in x's block

if none, move up + scan upper tier (until 1)

Move down + find min in low block

Runtime: $B + \frac{u}{B} + B$

$= O\left(B + \frac{u}{B}\right)$

How to find "best" value for B?

Calculus!

Minimize $O(\frac{u}{B} + B)$:

$$\frac{d}{dB}(uB^{-1} + B) = 0$$

$$\Rightarrow -uB^{-2} + 1 = 0$$

$$1 = uB^{-2} \Rightarrow B^2 = u$$

Solve for B:

$$B = \sqrt{u} = u^{1/2}$$

<u>Runtime</u>: $O(B + \frac{u}{B})$

$$= O\left(\sqrt{u} + \frac{u}{\sqrt{u}}\right)$$

$$= O(\sqrt{u})$$

What about deleting?

1	0	1	0	~~1~~ 0	1	0	0
001000~~1~~0	00000000	00011000	00000000	00000~~1~~00	11110111	00000000	00000000

~~1~~0 0

- 1 delete in bottom $O(1)$
- Is-empty
  - if empty, delete top $(0 \to 1)$

Runtime:

$O(\sqrt{u})$