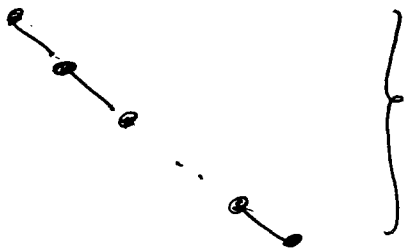
13.1

14.1 RED-BLACK TREES

RECALL THAT IN AN (ALMOST) COMPLETE BINARY TREE $h = \Theta(\lg n)$ WHERE h IS THE HEIGHT AND n IS THE NUMBER OF NODES.

THIS IS FALSE IN AN ARBITRARY BINARY TREE SINCE DIFFERENT PATHS FROM THE ROOT DOWN TO A LEAF CAN HAVE VERY DIFFERENT LENGTHS. WE COULD EVEN HAVE $h = n-1$ AS IN A LINEAR CHAIN:



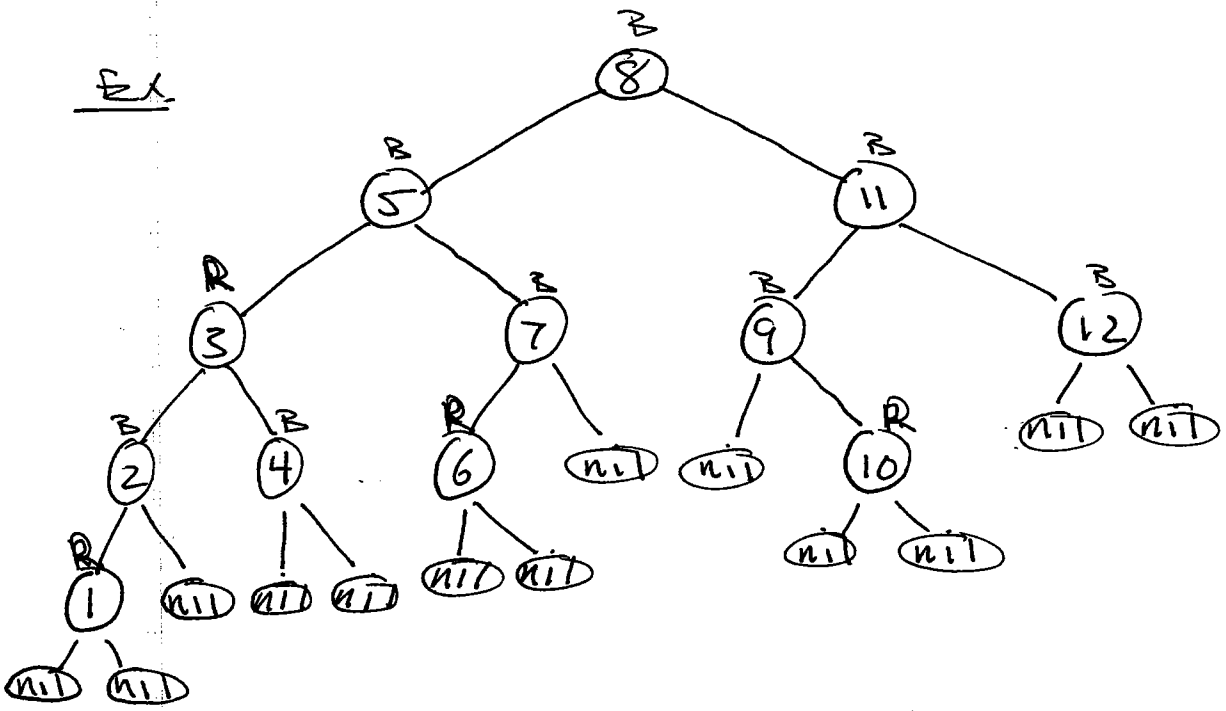
SINCE ALL THE BASIC TREE OPERATIONS IN LAST CHAPTER RAN IN $\Theta(h)$ TIME, WE ARE MOTIVATED TO KEEP THE HEIGHT OF A BST CLOSE TO ITS MINIMUM OF $\lfloor \lg n \rfloor$. (SUCH A TREE IS CALLED BALANCED.)

A <u>RED-BLACK</u> TREE IS A BST WHERE EACH NODE CONTAINS A <u>COLOR</u> FIELD WHICH CAN BE EITHER RED OR BLACK.

IN ADDITION WE CONSTRAIN THE WAY NODES CAN BE COLORED ON A PATH FROM ROOT TO LEAF SO THAT NO SUCH PATH IS MORE THAN TWICE AS LONG AS ANY OTHER.

NOTE IF A NODE'S CHILD IS MISSING, THE CORRESPONDING POINTER IS NIL. WE REGARD THESE NILS AS THE LEAVES OF THE RB TREE, AND THE KEY-BEARING NODES AS INTERNAL.

Ex.



A RED-BLACK TREE MUST SATISFY THE RED-BLACK TREE PROPERTIES:

(IN ADDITION TO THE BST PROPERTIES.)

1.) EACH NODE IS EITHER RED OR BLACK
2.) THE ROOT IS BLACK
3.) EACH LEAF (i.e. nil) IS BLACK
4.) IF A NODE IS RED, THEN BOTH ITS CHILDREN ARE BLACK.
5.) EVERY SIMPLE DOWNWARD PATH FROM A GIVEN NODE TO A DESCENDANT LEAF CONTAINS THE SAME NUMBER OF BLACK NODES.

RULES

- THE # OF BLACK NODES ON A SIMPLE DOWNWARD PATH FROM x TO A LEAF (NOT COUNTING x) IS CALLED THE BLACK HEIGHT OF x, DENOTED $bh(x)$. IT IS WELL DEFINED BY 5.
- ANY SUBTREE OF A RED BLACK TREE IS ALSO A RED BLACK TREE (IF NECESSARY COLOR ROOT BLACK)
- THE BLACK HEIGHT OF A RBT IS THE BLACK HEIGHT OF ITS ROOT.

Ex $bh(T) = bh(root[T]) = 3$
$bh(11) = 2$, $bh(2) = 1$, $bh(\text{Any nil}) = 0$

NOTE: $bh(x) = 0 \iff height(x) = 0$.

THEOREM

A RED BLACK TREE WITH n INTERNAL (i.e. NON-NIL) NODES, AND HEIGHT h SATISFIES

$$h \leq 2 \lg(n+1) .$$

Rmks

- SINCE NECESSARILY $h \geq \lfloor \lg(n) \rfloor$ WE HAVE $h = \Theta(\lg n)$
- THEREFORE RED BLACK TREES MAKE GOOD SEARCH TREES. i.e. Search, Min, Max, Successor, Predecessor ALL TAKE $\Theta(\lg n)$ TIME.
- Insert & Delete from last CHAPTER DO NOT NECESSARILY MAINTAIN THE RED BLACK PROPERTIES. THEY CAN HOWEVER BE MODIFIED TO DO SO AND STILL RUN IN $\Theta(h) = \Theta(\lg n)$ TIME.

LEMMA:

THE SUBTREE ROOTED AT ANY NODE x CONTAINS AT LEAST $2^{bh(x)} - 1$ INTERNAL NODES

LEMMA

IF x IS A NODE IN A RBT, THEN THE SUBTREE ROOTED AT x CONTAINS AT LEAST $2^{bh(x)} - 1$ INTERNAL NODES.

PROOF: INDUCTION ON $height(x)$.

BASE:

IF $height(x) = 0$, THEN x IS A LEAF (i.e. nil), SO $bh(x) = 0$, AND

$$2^{bh(x)} - 1 = 2^0 - 1 = 1 - 1 = 0.$$

IN THIS CASE THE SUBTREE ROOTED AT x HAS 0 INTERNAL NODES, SO THE BASE CASE IS VERIFIED.

INDUCTION

LET $height(x) > 0$ AND ASSUME FOR ANY NODE y WITH $height(y) < height(x)$ THAT THE SUBTREE ROOTED AT y HAS AT LEAST $2^{bh(y)} - 1$ INTERNAL NODES.

SINCE $height(x) > 0$, x IS AN INTERNAL NODE AND NECESSARILY HAS TWO CHILDREN (ONE OR BOTH OF WHICH MAY BE nil.)

IF left[x] is RED THEN bh(left[x]) = bh(x). IF left[x] is BLACK THEN bh(left[x]) = bh(x) − 1. IN ANY CASE

$$bh(Left[x]) \geq bh(x) - 1 .$$

LIKEWISE $bh(right[x]) \geq bh(x) - 1$.

SINCE height(left[x]) < height(x) THE INDUCTION HYPOTHESIS ASSURES US THAT THE SUBTREE ROOTED AT left[x] CONTAINS AT LEAST

$$2^{bh(left[x])} - 1 \geq 2^{bh(x)-1} - 1$$

INTERNAL NODES. LIKEWISE THE SUBTREE ROOTED AT right[x] HAS AT LEAST

$$2^{bh(right[x])} - 1 \geq 2^{bh[x]-1} - 1$$

INTERNAL NODES.

EACH CHILD OF x HAS BLACK-HEIGHT EITHER $bh(x)$ (IF CHILD IS RED), OR $bh(x)-1$ (IF CHILD BLACK.)

THE (ABSOLUTE) HEIGHTS OF THESE CHILDREN ARE LESS THAN height(x), SO APPLYING THE INDUCTION HYPOTHESIS, WE CONCLUDE THAT THE SUBTREES ROOTED AT EACH CHILD HAS AT LEAST

$$2^{bh(x)-1}-1$$

INTERNAL NODES (SINCE THEIR BLACK HEIGHTS ARE AT LEAST $bh(x)-1$.)

THEREFORE, THE SUBTREE ROOTED AT x MUST HAVE AT LEAST

$$\left(2^{bh(x)-1}-1\right)+\left(2^{bh(x)-1}-1\right)+1$$

$$= 2\cdot 2^{bh(x)-1}-1$$

$$= 2^{bh(x)}-1$$

INTERNAL NODES.

///.

THM A RBT WITH n INTERNAL NODES & height h SATISFIES: $h \leq 2\lg(n+1)$.

Proof of Theorem

LET h BE THE HEIGHT OF A RBT T WITH n INTERNAL (i.e. NON-NIL) NODES.

BY PROPERTY 4 (EVERY RED NODE HAS TWO BLACK CHILDREN), AT LEAST HALF THE NODES ON A PATH FROM ROOT TO LEAF (NOT INCLUDING ROOT) MUST BE BLACK. (OTHERWISE THERE WOULD BE TWO REDS IN A ROW IN SUCH A PATH.)

THUS

$$bh(root[T]) \geq \frac{h}{2}$$

BY THE LAST LEMMA

$$n \geq 2^{bh(root)} - 1 \geq 2^{\frac{h}{2}} - 1$$

$$\Rightarrow \quad 2^{h/2} \leq n+1$$

$$\Rightarrow \quad \frac{h}{2} \leq \lg(n+1)$$

$$\Rightarrow \quad h \leq 2\lg(n+1),$$

AS CLAIMED. ///

PROBLEM 13.1-5

LET x_0 BE A NODE IN A RBT. SHOW THAT THE LONGEST PATH FROM x_0 TO A DESCENDENT LEAF HAS LENGTH AT MOST TWICE THAT OF A SHORTEST SUCH PATH.

PROOF:

LET $p = (x_0, x_1, \ldots, x_h)$ BE THE LONGEST DESCENDING PATH FROM x_0 TO A LEAF AND LET $q = (x_0, y_1, y_2, \ldots, y_k)$ BE THE SHORTEST SUCH PATH. WE MUST SHOW $h \leq 2k$.

BY PROPERTY 4, AT LEAST HALF THE NODES IN p (OTHER THAN x_0) ARE BLACK.

$$\therefore \quad bh(x_0) \geq \frac{h}{2}$$

BUT BY PROPERTY 5, THE NUMBER OF BLACK NODES IN q (OTHER THAN x_0) IS THE SAME AS THAT IN p, NAMELY $bh(x_0)$.

$$\therefore \quad k \geq bh(x_0)$$

$$\therefore \quad k \geq \frac{h}{2} \qquad \therefore \; h \leq 2k.$$

///.

13.2 ~~14.2~~ ROTATIONS

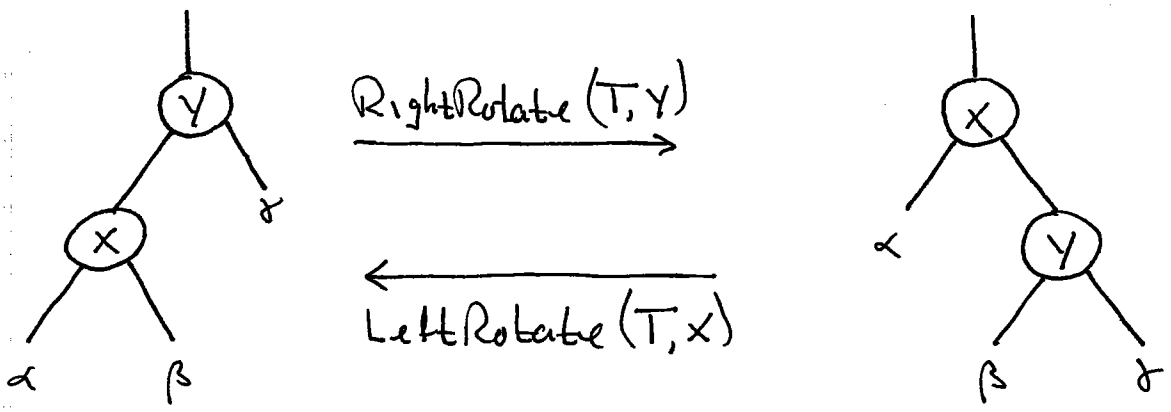
RECALL INSERT & DELETE WITHEN RUN ON A RED-BLACK TREE TAKE $\Theta(h) = \Theta(\lg n)$ TIME, BUT THEY MAY DESTROY THE RED-BLACK PROPERTIES.

IN ORDER TO RESTORE THESE PROPERTIES WE WILL NEED TO CHANGE SOME COLORS & CHANGE THE POINTER STRUCTURE, I.E. MOVE SOME NODES AROUND.

TO CHANGE THE POINTER STRUCTURE WE USE THE ROTATION OPERATION.

THIS IS A LOCAL OPERATION WHICH PRESERVES THE BST-PROPERTIES.

IN PICTURES:



HERE α, β, γ REPRESENT ARBITRARY SUBTREES.

OBSERVE THAT BOTH OPERATIONS PRESERVE THE INEQUALITIES

$$Keys(\alpha) \le key[x] \le Keys(\beta) \le key[y] \le Keys(\gamma)$$

SO THE BST PROPERTIES ARE PRESERVED.

NOTE ALSO RightRotate(T, y) HAS AS A PRECONDITION THAT left$[y] \neq$ NIL, AND LeftRotate(T, x) HAS right$[x] \neq$ NIL.

FIRST WE SUMMARIZE STEPS IN RightRotate.

RightRotate(T, y)

- left$[y] \leftarrow$ right$[x]$ (i.e. switch β)
- $p[$right$[x]] \leftarrow y$ (same)
- $p[x] \leftarrow p[y]$
- $p[y]$'s (left or right) child $\leftarrow x$
- right$[x] \leftarrow y$
- ~~[illegible]~~ $p[y] \leftarrow x$

(LeftRotate is summarized by replacing x by y, left by right.)

MOST OF THE SEEMING COMPLEXITY IN THE ALGORITHM COMES FROM DEALING WITH SPECIAL CASES: e.g. y is THE ROOT SO $p[y] =$ NIL or $\beta = \emptyset$ SO right$[x] =$ NIL.

BOOK DOES LeftRotate, SO WE DO

RightRotate(T, y) (Pre: left[y] ≠ nil)

```
1.) x ← left[y]
2.) left[y] ← right[x]
3.) if right[x] ≠ nil
4.)     p[right[x]] ← y
5.) p[x] ← p[y]
6.) if p[y] = nil
7.)     root[T] ← x
8.) else if y = right[p[y]]
9.)     right[p[y]] ← x
10.) else
11.)    left[p[y]] ← x
12.) right[x] ← y
13.) p[y] ← x
```

NOT NECESSARY IF WE USE A SENTINEL NODE nil[T] TO STAND FOR nil.

REMARKS

- BOTH ROTATIONS RUN IN O(1) TIME
- BOTH PRESERVE BST PROPERTIES.
- BOTH DO NOT PRESERVE RBT PROPERTIES. e.g. DO RightRotate WHEN y, β RED, x, γ BLACK, α ARBITRARY. THEN RBT 4 IS VIOLATED.