CS 315 - Day 27, Red-Black Trees Insertion and Induction

So, what is the running time for Insertion into an RB Tree?

```
RB-INSERT(T, z)
  x = T.root                  // node being compared with z
  y = T.nil                   // y will be parent of z
  while x ≠ T.nil             // descend until reaching the sentinel
      y = x
      if z.key < x.key
          x = x.left
      else x = x.right
  z.p = y                     // found the location—insert z with parent y
  if y == T.nil
      T.root = z              // tree T was empty
  elseif z.key < y.key
      y.left = z
  else y.right = z
  z.left = T.nil              // both of z's children are the sentinel
  z.right = T.nil
  z.color = RED               // the new node starts out red
  RB-INSERT-FIXUP(T, z)       // correct any violations of red-black properties
```

```
RB-INSERT-FIXUP(T, z)
  while z.p.color == RED
      if z.p == z.p.p.left            // is z's parent a left child?
          y = z.p.p.right             // y is z's uncle
          if y.color == RED           // are z's parent and uncle both red?
              z.p.color = BLACK       ⎫
              y.color = BLACK         ⎬ case 1
              z.p.p.color = RED       ⎪
              z = z.p.p               ⎭
          else
              if z == z.p.right
                  z = z.p             ⎫ case 2
                  LEFT-ROTATE(T, z)   ⎭
              z.p.color = BLACK       ⎫
              z.p.p.color = RED       ⎬ case 3
              RIGHT-ROTATE(T, z.p.p)  ⎭
      else (same as then part, but with "right" and "left" exchanged)
  T.root.color = BLACK
```

What is the running time of RB-INSERT? Since the height of a red-black tree on n nodes is $O(\lg n)$, lines 1–16 of RB-INSERT take $O(\lg n)$ time. In RB-INSERT-FIXUP, the **while** loop repeats only if case 1 occurs, and then the pointer z moves two levels up the tree. The total number of times the **while** loop can be executed is therefore $O(\lg n)$. Thus, RB-INSERT takes a total of $O(\lg n)$ time. Moreover, it

<u>Red-Black Properties</u>

1. Every node is either **red** or **black**.
2. The root is **black**.
3. Every leaf (NIL) is **black**.
4. If a node is **red**, then both its children are **black**.
5. For each node, all simple paths from the node to descendant leaves contain the same number of **black** nodes.

In a valid RB Tree, a node x with height h has $bh(x) \geq \frac{h}{2}$

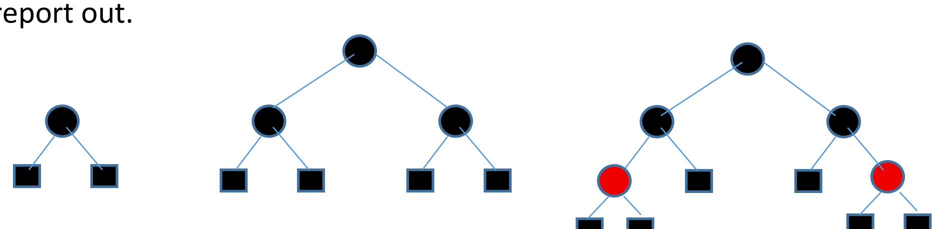
As a convenience, `RB-tree.ipynb` is provided. Try loading it and running it, taking note of the examples provided and the corresponding output.

<u>In-class w/***assigned groups of four (4)***, write 0. and 1. on board as ground rule(s):</u>

0. No devices.
1. Intro: Name, POE, Icebreaker.
2. Pick someone on the group who will report out.

Work together on the following:

3. Are the following valid R-B trees?



(a) ***(b)*** ***(c)***

4. Identify the internal (*i*) and NIL / leaf / external (***NIL***) nodes in each tree above.

5. Give the ***bh*** of each node the in the trees above.

6. How many children does each internal node above have?

7. What's the difference between ***bh*** and ***bh(x)*** ?

8. Ultimately, we want to show that an R-B Tree with ***n*** internal nodes has height ***at most*** $2*\lg(n+1)$ nodes. It's pretty easy to show this if we have an intermediate result, that ***the subtree rooted at** x **in an R-B Tree has at least** $2^{bh(x)}-1$ **internal nodes*** (proof on the next page). ***Starting with that intermediate result, we have:***

$n \geq 2^{bh}-1$

$n \geq 2^{bh}-1 \geq 2 \quad -1$ *(substitution using something above and shown in class)*

$n \geq$ ______________ *(no need for intermediate inequality)*

$n+1 \geq$ ___________ *(algebra)*

(show a few steps)

$2*\lg(n+1) \geq h$

9. Is $\lg(n+1) = O(\lg(n))$? Show your work.

10. *Prove:* The subtree rooted at x in an R-B Tree has at least $2^{bh(x)}-1$ internal nodes.

<u>Basis Step:</u> [x is a NIL / leaf / external node]

"LHS" "RHS"

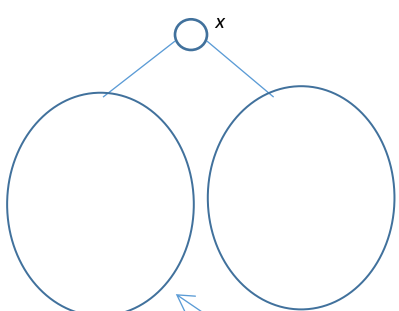
<u>Inductive Step:</u> [x is an internal node]

x is an internal node. We know it has two children. (***Why?*** _____________________________________)

Each child of x has $bh(x)$ or $bh(x)-1$. (***Why?*** ______________________________________)

<u>Assume</u> the subtrees rooted at the ***<u>children</u>*** of x have at least $2^{bh(x)-1}-1$ internal nodes,

and ***<u>show</u>*** the subtree rooted ***<u>at x</u>*** have at least $2^{bh(x)}-1$ internal nodes.



__________ internal nodes in left subtree of *x* **+** __________ internal nodes in right subtree of *x* **+** 1 (node *x*)

[***continue on with algebra***]

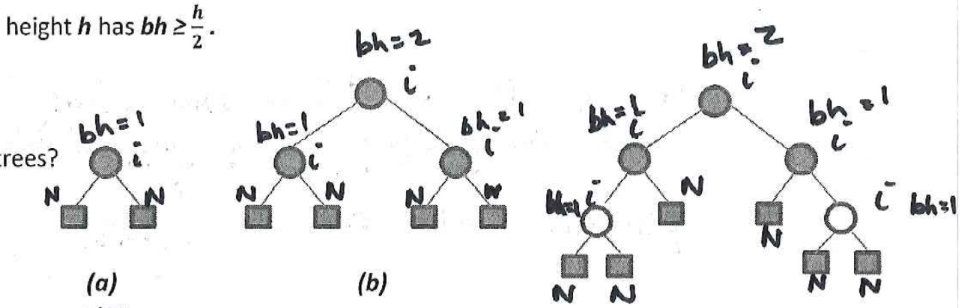
CS 315, Algorithms and Analysis Name: KEY

***R**ed-**B**lack Tree Properties*

1. Every node is R or B
2. Root is always B
3. Every NIL / leaf / external node is B
4. If a node is R, then both its children are B
5. For each node, all simple paths from the node to descendant leaves contains the same number of B nodes.

In a valid R-B tree, a node with height h has $bh \geq \frac{h}{2}$.

1. Are the following valid R-B trees?



(a) yes

(b) yes

(c) yes

2. Identify the internal (*i*) and NIL / leaf / external (***NIL***) nodes in each tree above.

3. Give the ***bh*** of each node the in the trees above. all NIL have bh = 0

4. How many children does each internal node above have? two, always.

bh @ a node labeled x

5. What's the difference between ***bh*** and ***bh(x)*** ?

implied to be bh of entire tree

Ultimately, we want to show that an R-B Tree with ***n*** internal nodes has height ***at most*** $2*\lg(n+1)$ nodes. It's pretty easy to show this if we have an intermediate result, that ***the subtree rooted at x in an R-B Tree has at least*** $2^{bh(x)} - 1$ ***internal nodes*** (proof on the next page). ***Starting with that intermediate result, we have:***

$n \geq 2^{bh} - 1$

$n \geq 2^{bh} - 1 \geq 2^{h/2} - 1$ *(substitution using something above and shown in class)*

$n \geq 2^{h/2} - 1$ *(no need for intermediate inequality)*

$n + 1 \geq 2^{h/2}$ *(algebra)*

$\lg(n+1) \geq \lg(2^{h/2})$

$\lg(n+1) \geq \frac{h}{2} \lg 2$ *(show a few steps)*

$2 \lg(n+1) \geq h$

$2*\lg(n+1) \geq h$

6. Is $\lg(n+1) = O(\lg(n))$? Show your work. Yes

By defn $\lg(n+1) \le c \lg(n)$ For some c + $n \ge n_0$

$$2^{\lg(n+1)} \le 2^{c \cdot \lg(n)} = 2^c \, 2^{\lg(n)}$$

$$\downarrow$$

$$n+1 \le 2^c \cdot n$$

$$n+1 \le K \cdot n$$

let $c = 4$ + $n_0 = 1$

Prove: The subtree rooted at x in an R-B Tree has at least $2^{bh(x)} - 1$ internal nodes.

Basis Step: [x is a NIL / leaf / external node]

"LHS" if x is a NIL node, has ht = 0 and $bh(x) = 0$, and it doesn't have any children (or internal nodes below)

"RHS" plugging into formula $2^{bh(x)} - 1 = 2^0 - 1 = 1 - 1 = 0$ internal nodes

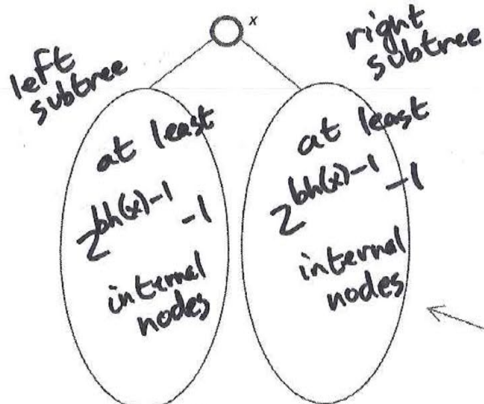
Inductive Step: [x is an internal node]

x is an internal node. We know it has two children. (***Why?*** Front page, all internal nodes have at least 2 NIL nodes

Each child of x has $bh(x)$ or $bh(x) - 1$. (***Why?*** if child of x is B, then its bh ~~is 1 less than x.~~ if child of x is R, then its bh is same as x

Assume the subtrees rooted at the ***children*** of x have at least $2^{bh(x)-1} - 1$ internal nodes,

and ***show*** the subtree rooted ***at x*** have at least $2^{bh(x)} - 1$ internal nodes.



$2^{bh(x)-1} - 1$ internal nodes in left subtree of x + $2^{bh(x)-1} - 1$ internal nodes in right subtree of x + 1 (node x)

[***continue on with algebra***]

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

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

Q.E.D.