Lecture Notes - Red-Black Trees

- Discuss self-balancing.
- AVL Trees + (2,4) trees require many "restructurings."
- Red-Black Trees will only require AVL "restructurings" for balance.

**Definition:**

- Red-Black tree is a binary search tree with nodes colored red and black such that:
  - Root property: The root is black.
  - External property: Every external node is black.
  - Red property: Children of a red node are black.
  - Depth property: All external nodes have the same black depth, defined as the number of black ancestors.

**Example:**

![Red-Black Tree Diagram](image)

Note: relationship to properties.

Then, the height of a red-black tree storing n elements is $O(\log n)$.

**Proof:** Recall the property that the height of a binary tree must be $\geq \log_{2}^{n}$.

Let $h(x)$ be the black nodes in every $x$-to-leaf path. (All the same by Prop 4).

So a subtree at $x$ has at least $2^{\text{height}(x)}$ nodes, by lower bound.

$\text{BH}(\text{root}) \geq \log \frac{n}{2}$ by Prop III, otherwise double red occurs.

$\text{BH}(\text{root}) \geq \log \frac{n}{2}$

$\log \frac{n}{2} \geq \frac{1}{2} \log n$ for $n \geq 2^{\log \frac{n}{2} - 1}$
- Get operation identical to BST.

- Insertion -
  - Begin with standard insertion. If it is the first entry (root) label as black.
  - Otherwise label as red. However, this could violate red property.
  - We call this a double red at the inserted node x.

Example of double red

- Example of regular insert

To remedy double red, we have 2 cases.
(i) sibling s of y is black
(ii) sibling s of y is red

- Case 1 - Sibling s of y is black.
  - Resolution - Tri-node restructuring. After color b = black and a, c as red.

4 possible configurations:

Note - no change in any black depths.
- Case 2 - sibling s of y is red
  - resolution - recolor. y becomes red (unless it's the root)
    y's become black

- Note, black depth is unaffected unless it's the root. In that case it increases by 1.

- However, changing y to red might propagate double red problem higher to the tree. Repeat two cases at z until a recolor eliminates problem, a restructure eliminates problem.

- Complexity
  - $O(\log n)$ search/insert
  - $O(\log n)$ recolorings - half the height of the tree
  - $O(1)$ tri-node restructurings
  - Total: $O(\log n)$

- Activity - Insert the following into a red-black tree:
  30, 40, 24, 58, 48, 26, 11, 13

- Answer
  2 recolors @ 58, 13
  1 restructure @ 48
- Deletion -

- Begin with standard BST removal.

- If the removed node is red, no structural changes or any black depth, nor violate red violations.

- If it was black, it must have had EIHT level, and both children were external as one was a red node with a external children, i.e., we created a black depth deficit. In the second case, recolor as black. In the first, we have.

- Let p be node promoted upon removal. Define let y be sibling of p.

- When p is black, temporarily label as "double black" and must remedy.

- Case 1: Sibling y of p is black and has a red child x

  - Resolution: tri-node restructuring. z will be parent of y.

  - color a, c as black, b gets former value of z.

  - Note - path to p has one more black node now, others are unaffected.

- Example:

```
          Z
          |
          y
          |
         10
         |
         X
         30
          |
         p
          |
         40
```

```
          a
          |
         10
          |
         30
          |
         c
         40
          |
         p
```

```
          Z
          |
          y
          |
         20
          |
         30
          |
         p
          |
         40
```
- **Case 2**: The sibling $y$ of $p$ is black and both children of $y$ are black.
  
  Resolution: Recolor $y$ becomes red. Now consider parent $z$ of $y$.
  
  (because decrease in black depth through $y$)
  
  if $z$ is red, color black and problem resolved.
  
  if $z$ is black, color double-black propagating the problem.

  **Examples**:

  ![Diagram 1](image1.png)

  ![Diagram 2](image2.png)

- **Case 3**: Sibling $y$ of $p$ is red.
  
  Let $z$ be common parent. It must be black because $y$ is red.
  
  Resolution: Adjustment through rotation + recolor.
  
  Rotate about $y$ and $z$ - recolor $y$ black and $z$ red.
  
  After, reconsider problem at $p$: Sibling of $p$ is black so case 1 or 2 applies.
  
  Next application must be last because case 1 is terminal and case 2 is terminal given that parent of $p$ is now red.

  **Example**:

  ![Diagram 3](image3.png)

  **Performance (complexity)**

  - Find - $O(\log n)$
  - Recoloring - $O(\log n)$
  - Restructuring - $O(1)$ - why it's better than AVL + (2,4) trees.
  - Total - $O(\log n)$
- Activity - Start with the following:

```
     30
   /   \
  24   48
 /   / \
11  26  40  58
   /     \
  13     26
```

and remove 30 and then 48 and then 58

- answer
  - remove 30 causes no imbalance
  - remove 48 case 0 promoted node was red with 2 black children
  - remove 50 case 3 rotate + recolor
    then case 2 reoder :

```
     24
   /   \
  11   40
   /     \
  13     26
```