CSE 100: Final Review
Red Black Trees

Four rules:
1. Nodes are either red or black
2. Root is always black
3. No two red nodes in a root to leaf path. Children of red node must be black
4. For every node X, every path from X to a null reference must contain the same number of black nodes

Height is at most $2 \log_2(N+1)$
Can a Red-Black tree have a black node with exactly one black child and no red child?

Answer: No

Property 4 gets violated - Every path from the root to a nil node must contain the same number of black nodes

The root to nil node path for the nodes attached to ‘10’ will have one black node more than the root to nil node path for the nil node attached directly to ‘20’.
Practice Question

The subtree of the root of a red-black tree is always itself a red-black tree. True or False?

Answer: False

Property 2 (The root node of the RB tree should be Black) will not be satisfied for the subtree.

The child of the root node could either be a red child or a black child.

If the child is red, then the subtree rooted at the red child will not be a RB tree because the root of the RB tree must be black.
Insertion

X is the node to be inserted, P is the parent, G is the parent of P (Grandparent of X)

**Step 1:** Color the node X as red and insert it to the Red Black

**Step 2:**

**Case 1: If sibling of parent is RED**
   a. P is left child of G, and X is left child of P
   b. P is left child of G, and X is right child of P
   c. P is right child of G, and X is right child of P
   d. P is right child of G, and X is left child of P

**Case 2: If sibling of parent is BLACK**

**Case 3: if X is root → change it’s color to black and your’re done !**
Insertions

Let's recap different rotations in a BST before diving in

Right rotation: (at z)

```
z   /
 /  
y   T4   x   z
 /    ----->   /   /
T1   T3   T1  T2  T3  T4
 /   /
T2   T1  T2
```

Left rotation: (at z)

```
z   /
 /  
T1  y
 /  ----->   z  x
/    /   /
T2  T1   T1  T2  T3  T4
 /   /
T3  T4
```
Insertion - Case 1a

Right Rotate P, recolor
Insertion - Case 1b

Left Rotate P, NO recoloring

Right Rotate G, recolor
Insertion - Case 1c and 1d

Similar to Case 1a and Case 1b
After this step, now consider G as X and repeat Step 2 (includes either case 1, case 2 or case 3) if there is any property violation.
Ternary Trees

Convince yourself that “CUP” is not in the tree

Following are the 5 fields in a node:

1) The data (a character)
2) isEndOfString bit (0 or 1). It may be 1 for nonleaf nodes (the node with character T)
3) Left Pointer
4) Equal Pointer
5) Right Pointer

Ternary Search Tree for CAT, BUG, CATS, UP
(a) Consider the following ternary search tree. Nodes with double circles have their end bits set to true. Circle all of the words from the list on the right that are in the tree and write in any words that are missing. At the end you should have a complete list of all words found in the tree, and only those words.

Nodes in the tree - a, an, and, as, at, air, all

air, all, a, and, as

(b) Does the height of this tree depend on the order in which the keys have been inserted? Yes

(c) Briefly explain why you would prefer to use a ternary search tree rather than a binary search tree to implement `getAllValidWords()` in PA4.
Run time analysis

3. Run Time Analysis

(a) Consider the following algorithm operating of a graph $G(V, E)$ and a source vertex $s$
   i. Initially, set the visited field of all vertices in $V$ as FALSE
   ii. Start at source vertex, $s$. Set the visited field of $s$ as TRUE, set counter to 1
   iii. Enqueue $s$ into a queue
   iv. While the queue is not empty or counter less than or equal to $|V|$
      A. Dequeue the vertex $v$ from the head of the queue
      B. For each of $v$’s neighbor vertex, $w$ (with visited FALSE):
         • Mark the visited field of $w$ as TRUE
         • Increment counter by 1
         • Enqueue $w$ into the queue
   i. Identify the above graph algorithm.

   BFS

   ii. Assume the graph is implemented as an adjacency list. Write the Big O running time of each step of the algorithm (against each line of above) and derive the tightest overall Big O bound.

   $O(|V| + |E|)$
BFS

What if we use adjacency matrix?

$O(|V|^2)$
Good luck !