# CS 313 - Project 4
# Red Black Tree

March 2, 2023

## 1 Project Overview

For this project, we will be taking what we've learned in class and using it to implement a Red-Black Tree (RBT). In CS, a red-black tree is a special type of binary tree that self-balances. This is an effort to keep the time cost of accesses to strictly $O(lg(n))$ at the cost of a $O(n)$ operation somewhere else. For this project, you will extend your code in project 2 to now include the balancing operations. For your convenience, I implemented most of the RBT operations and provided them for you via the file **p4_starter_code.py**. You simply need to extend the functionality to support balanced insert operations.

A BST is a Red-Black tree if it satisfies the following Red-Black properties:

- Every node is either red or black
- Every leaf node counts as black
- If a node is red, then both of its children are black
- Every simple path from a node to a descendant leaf contains the same number of black nodes
- The root node is always black

## 2 Program Requirements:

You will need to write a RedBlackTree class. To receive any credit, your class must follow the specifications below exactly:

- Class Name: **RedBlackTree**
- Methods:
  - $\_\_init\_\_(< RedBlackTree > self) \rightarrow < Nonetype > None$
    - Complexity: O(1)
    - Instance variables:
      - $< RBNode > \_root$: This variable contains the head of our tree. Note: must be initialized as None.
      - Any other instance variables you want.
  - $\_leftRotate(< RedBlackTree > self, < RBNode > currentNode) \rightarrow None$
    - Complexity: $O(1)$
    - Valid Input: A RBNode item.
    - Error Handling: No error handling is needed.
    - Description: This is a helper method. It performs the left rotation on a binary tree (see Figure 1).

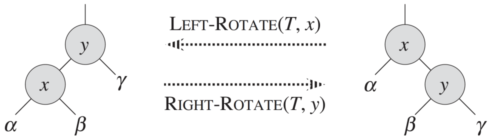

Figure 1: Node rotation on a binary tree

- $\_insertFixup(< RedBlackTree > self, < RBNode > currentNode) \rightarrow None$
  - Complexity: $O(log(n))$
  - Valid Input: A RBNode node.
  - Error Handling: No error handling is needed.
  - Description: This is a helper method. It performs the fix-up following the instruction in the book.

## 3 Submission Requirements:

Submit a single file **p4.py** to canvas.

## 4 Grading:

Your work will be graded along three primary metrics: Correctness, Completeness, and Elegance.

- Correctness: (60 points)
  - You wrote the class methods as specified.
  - Your class methods meet the complexity requirements.
  - You utilize a linked list to build your RBT.
  - You implement the correct algorithms.
  - Your classes are robust, fault-tolerant, and follow the specified behavior on invalid input.
- Completeness (25 points)
  - Program contains a class named: RedBlackTree
  - The class contains methods as defined above.
  - The method signatures were implemented as specified.
- Elegance: (15 points)
  - See the programming guide posted on canvas for information on elegance.

## 5 Remarks:

- The methods listed above are the only methods that will be tested. However, you may add additional methods as you please. We have provided a skeleton code to get you started be sure to read and understand the auxiliary methods provided before you start. The skeleton code provides the methods for find, traverse, an RB-node class, and some helpful extras you may use if you want.
- Pay attention to the complexity bounds mentioned in the method descriptions. Your implementation must run within these bounds.