

## Assignment 3

### RedBlack Trees

#### Introduction

- Most of the BST operations (e.g., search, max, min, insert,) take  $O(h)$  time where  $h$  is the height of the BST.
- The cost of these operations may become  $O(n)$  for a skewed Binary tree.
- If we make sure that the height of the tree remains  $O(\log n)$  after every insertion and deletion, then we can guarantee an upper bound of  $O(\log n)$  for all these operations.
- The height of a Red-Black tree is always  $O(\log n)$  where  $n$  is the number of nodes in the tree.

#### 2. Lab Goal

- This lab assignment focuses on balanced binary search trees and focusing on one of Balanced BST RedBlack trees.

#### 3. Red Black Trees Implementation

##### 3.1 Requirements

**You are required to implement the Red-Black Tree data structure supporting the following operations:**

1. **Search:** Search for a specific element in a Red-Black Tree.
2. **Insertion:** Insert a new node in a Red-Black tree. Tree balance must be maintained via the rotation operations.
3. **Print Tree Height:** Print the height of the Red-Black tree. This is the longest path from the root to a leaf-node.
4. **Print Black Height:** number of black nodes in any path from root to leaf.
5. **Print Tree size:** Print the number of elements in Red-Black tree.

## 4. Application

### 4.1 Introduction

English Dictionary As an application based on your Red-Black Tree implementation.

### 4.2 Requirements

**You are required to implement a simple English dictionary, with a simple text-based user interface, supporting the following functionalities:**

1. **Load Dictionary:** You will be provided with a text file, "dictionary.txt", containing a list of words. Each word will be in a separate line. - You should load the dictionary into a Red-Black Tree data structure to support efficient insertions and search operations.
2. **Insert Word:** Takes a word from the user and inserts it, only if it is not already in the dictionary. Otherwise, print the appropriate error message (e.g. "ERROR: Word already in the dictionary!"). The dictionary file must be updated with the newly inserted word.
3. **Look-up a Word:** Takes a word from the user and prints "YES" or "NO" according to whether it is found or not.

**Note: For validation purposes, you are required to print both the size of the dictionary, height, and the black height of your Red-Black tree after each insertion.**

## 5. Notes

- Implement your algorithms using (Python, Java, or C/C++) Preferably Python
- You should work in groups **of members.**
- Discussion will have higher weight than implementation, so you should understand your implementation well to get discussion marks.
- Late submissions are not allowed unless there is a valid documented excuse.