Assignment 2
Red&Black Trees

Most of the BST operations (e.g., search, max, min, insert, delete.. etc) 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(Logn) after every insertion and deletion, then we can guarantee an upper bound of O(Logn) for all these operations. The height of a Red-Black tree is always O(Logn) where n is the number of nodes in the tree.

1. Lab Goal

This lab assignment focuses on balanced binary search trees, and focusing on one of Balanced BST Red&Black trees.

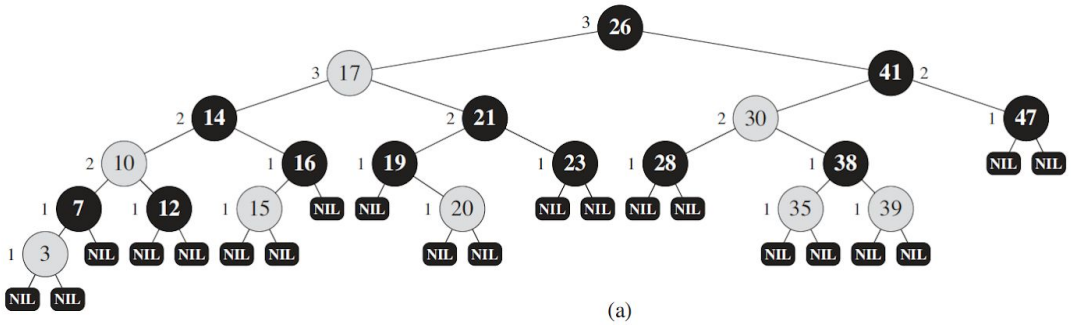
TA: Eng. Sami Mamdouh
Eng. Mohamed Ibrahim
Eng. Reem Hesham
Eng. Omar Aly

Prof Dr. Amr Elmasry

2. Background

2.1 Introduction

A red-black tree is a binary search tree with one extra bit of storage per node: its color, which can be either RED or BLACK. By constraining the node colors on any simple path from the root to a leaf, red-black trees ensure that no such path is more than twice as long as any other, so that the tree is approximately balanced.



(a)

A red-black tree is a binary tree that satisfies the following red-black properties:

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

2.2 Red-Black Tree Implementation

You're required to implement some basic procedures:

2.2.1 Red-Black Tree Implementation

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.

Please refer to the reference below for more implementation details.

3. Application: English Dictionary

As an application based on your Red-Black Tree implementation, 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, deletions and search operations.

2- Print Dictionary Size:

- Prints the current size of your dictionary.

3- 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!").

4- 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 and the height of your Red-Black tree after each insertion

4. References

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein "Introduction to Algorithms 3rd Edition - Thomas H. Cormen, Charles E. Leiserson, R"

Weiss, Mark Allen. "Data structures and algorithm analysis in Java." Addison-Wesley Long-man Publishing Co., Inc., 1998.

5. Notes

- Implement your algorithms using (Java, C/C++ or Python)
- We will submit a URL to upload the source code, your compressed files name should be students ids for example 1234_3456.zip
- You should work in groups **of 2 or 3 members**
- Discussion will have higher weight than implementation, so you should understand your implementation well to get discussion marks.

Good Luck

TA: Eng. Sami Mamdouh
Eng. Mohamed Ibrahim
Eng. Reem Hesham
Eng. Omar Aly

Prof Dr. Amr Elmasry