CSN 261

DATA STRUCTURES LAB

ASSIGNMENT 3

DIVYANSHU SETIA

18114020

B. TECH, CSE

PROBLEM 1:

Given the set of integers, write a C++ program to create a binary search tree (BST) and print all possible paths for it. You are not allowed to use subarray to print the paths. Convert the obtained BST into the corresponding AVL tree for the same input. AVL tree is a selfbalancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property. Convert the obtained BST into the corresponding red-black tree for the same input. Red-Black Tree is a self-balancing Binary Search Tree (BST) where every node follows following rules.

1) Every node has a color either red or black.

2) Root of tree is always black.

3) There are no two adjacent red nodes (A red node cannot have a red parent or red child).

4) Every path from a node (including root) to any of its descendant NULL node has the same number of black nodes. Write a menu driven program as follows:

1. To insert a node in the BST and in the red-black tree

2. To create AVL tree from the inorder traversal of the BST

3. To print the inorder traversal of the BST/AVL/red-black tree

4. To display all the paths in the BST/AVL tree/red-black tree

5. To print the BST/AVL tree/red-black Tree in the terminal using level-wise indentation (print color for red-black tree)

6. Exit

INPUT:

10 20 30 40 50 25

OUTPUT:

```
1) Insert new node in BST and Red-Black-Tree
2) Create AVL tree from the inorder traversal of the BST
3) Print tree data through inorder traversal
4) Print all the paths in the BST/AVL/Red-Black-Tree
5) Print the BST/AVL/Red-Black-Tree using level-wise indentation
6) Exit

5
1) Red Black Tree
2) Binary Search Tree
1
20[2][BLACK]
  10[BLACK]
  40[1][RED]
    30[1][BLACK]
      25[RED]
    50[BLACK]
```

```
5
1) Red Black Tree
2) Binary Search Tree
2
10[4]
  20[3]
    30[1]
      25
      40[1]
        50
```

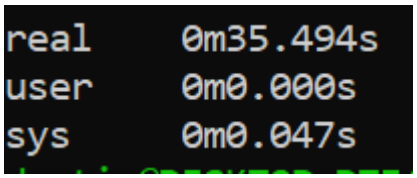
```
5
1) Red Black Tree
2) Binary Search Tree
2
30[1]
  20
    10
    25
  40[1]
    50
```

On converting to AVL

ALGORITHMS AND DATA STRUCTURES:

- A BST and a Red Black Tree are implemented
- Recursive functions are used for insertion and printing of data in trees
- In order traversal is used to get sorted array of data which is used to build the AVL tree
- Vectors are used to store paths to leaf nodes (used for printing the paths)
- Left and right rotations about a node are used for restoring the properties of the Red Black Tree upon an insertion

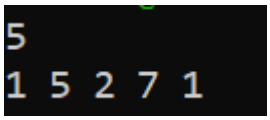
TIME:



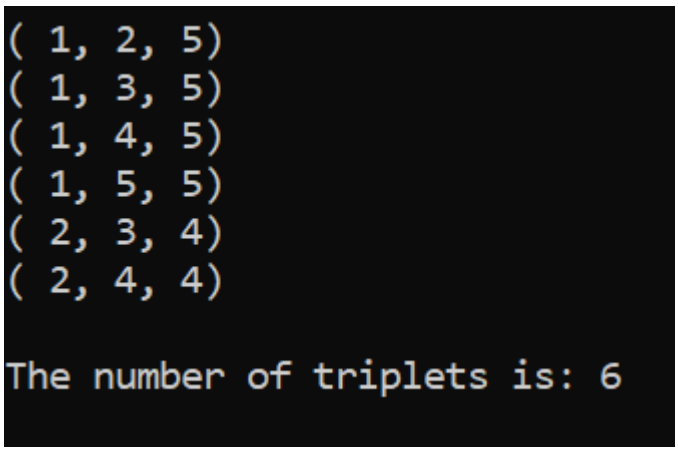
PROBLEM 2:

For a given sequence of positive integers A1, A2, …, AN in decimal, find the triples (i, j, k), such that $1 \le i < j \le k \le N$ and $A_i \oplus A_{i+1} \oplus \ldots \oplus A_{j-1} = A_j \oplus A_{j+1} \oplus \ldots \oplus A_k$, where $\oplus$ denotes bitwise XOR. This problem should be solved using dynamic programming approach and linked list data structures. Input: (a) Number of positive integers N. (b) N space-separated integers A1, A2, …, AN. Output: Print the number (count) of triples and list all the triplets in lexicographic order (each triplet in a new line).

INPUT:



OUTPUT:



ALGORITHMS AND DATA STRUCTURES:

- A linked list and arrays are used to store data
- We use the properties that a^0 = a, a^a = 0

TIME: