IIT Roorkee
CSN 261:Data Structures Laboratory

Assingment 3

Name:Ritesh Singh

Enrollment Number:18114067

Email: rsingh1@cs.iitr.ac.in

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

Example:

Input: 10 20 30 40 50 25

Output:

BST: 10 [4]

20 [3]

25

30 [2]

40 [1]

50

Inorder traversal (This output is for BST Tree): 10 20 25 30 40 50

AVL Tree: 30 [0]

20 [0]

10

25

40 [1]

50

Inorder traversal (This output is for AVL Tree): 10 20 25 30 40 50

Red Black Tree: 20 [2] [BLACK]

10 [BLACK]

40 [1] [RED]

30 [1] [BLACK]

25 [RED]

50 [BLACK]

Inorder traversal (This output is for Red Black Tree): 10 20 25 30 40 50

Paths (This output is for AVL tree):

30->20->10

20->10

10

30->20->25

20->25

25

30->40->50

40->50

50

Algorithm Used:

We have used the Binary Search Algorithm to create the Binary Search Tree.
We have use recurrsion to traverse across the trees to print the path and level wise indentation.
We have used the left rotation and the right rotation to balance the AVL Tree
Similarly, we have used rotations in Red Black Tree so that the tree donot violates the property of Red Black Tree

Data-Structures Used:

1.Srtucture for BST Tree
2.Structure for AVL Tree
3.Structure for RED BLACK Tree
4.Arrays
5.Linked List

Snapshots:

Starting Program-

```
ritesh@kaneki:~/Desktop/DS_lab_3$ g++ problem1.cpp -o a
ritesh@kaneki:~/Desktop/DS_lab_3$ ./a
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
6. Exit
```

Print Inorder Traversal-

```
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
6. Exit

3

The inorder traversal of BST tree is:
10      20      25      30      40      50
The inorder traversal of AVL tree is:
10      20      25      30      40      50
The inorder traversal of Red Black tree is:
10      20      25      30      40      50
```

Display All the Paths-

```
4

The complete path of BST Tree:
10->20->30->25
20->30->25
30->25
25

10->20->30->40->50
20->30->40->50
30->40->50
40->50
50

The complete path of AVL Tree
30->20->10
20->10
10

30->20->25
20->25
25

30->40->50
40->50
50

The complete path of RED BLACK Tree:
20->10
10

20->40->30->25
40->30->25
30->25
25

20->40->50
40->50
50
```

Display Level-wise Indentation-

```
5

The level-wise indentation of BST Tree:
10[4]
        20[3]
                30[2]
                        25[0]
                        40[1]
                                50[0]
The level-wise indentation of AVL Tree:
30[0]
        20[0]
                10[0]
                25[0]
        40[1]
                50[0]
The level-wise indentation of RED BLACK Tree:
20[BLACK]
        10[BLACK]
        40[RED]
                30[BLACK]
                        25[RED]
                50[BLACK]
```

Time-

```
ritesh@kaneki:~/Desktop/DS_lab_3$ time ./a
```

```
real    0m17.392s
user    0m0.001s
sys     0m0.007s
ritesh@kaneki:~/Desktop/DS_lab_3$
```

Problem 2-

For a given sequence of positive integers A1, A2, …, AN in decimal, find the triples (i, j, k), such that $1 \leq i < j \leq k \leq N$ and $Ai \oplus Ai+1 \oplus \ldots \oplus Aj-1 = Aj \oplus Aj+1 \oplus \ldots \oplus Ak$, 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).

Example: Input:

N = 3 5 2 7

Output: 2

(1, 2, 3)

(1, 3, 3)

Algorithm Used:

1.If XOR of a subarray of length L is 0, then there are $L-1$ valid triplets.

Realize that $Ai \oplus Ai+1 \oplus ....\oplus Aj-1 = Aj \oplus Aj+1 \oplus ....\oplus Ak$ on re-arranging the terms becomes $Ai \oplus Ai+1 \oplus ....\oplus Aj-1 \oplus Aj \oplus Aj+1 \oplus ....\oplus Ak = 0$, i.e.

the XOR of subarray in range $[i,k]$ must be 0.

2.We have calculated the pre_xor i.e a[0], a[0]^a[1], a[0]^a[1}^a[2} ,..... ,a[0]^.....a[n-1].

And stored that in a linked list.

Data-Structures Used:

1.Linked List

2.Arrays

3.Unordered_Map

Snapshots:

Starting the Program-

```
ritesh@kaneki:~/Desktop/DS_lab_3$ g++ problem2.cpp -o a
ritesh@kaneki:~/Desktop/DS_lab_3$ ./a
Enter the value of n:
```

Output of Program-

```
ritesh@kaneki:~/Desktop/DS_lab_3$ g++ problem2.cpp -o a
ritesh@kaneki:~/Desktop/DS_lab_3$ ./a
Enter the value of n:
3
Enter n spaced integers:
5 2 7
The total number of triplets are:
2
The list of triplets are:
(1,2,3)
(1,3,3)
ritesh@kaneki:~/Desktop/DS_lab_3$
```

Time-

```
ritesh@kaneki:~/Desktop/DS_lab_3$ time ./a
Enter the value of n:
3
Enter n spaced integers:
5 2 7
The total number of triplets are:
2
The list of triplets are:
(1,2,3)
(1,3,3)

real    0m3.504s
user    0m0.005s
sys     0m0.000s
```