CSN-261: Data Structures Laboratory

Assignment 3

Name – Kavya Barnwal

Class- B.Tech CSE

Sub Batch- O2

Enrollment No- 18114039

Email Id - kbarnwal@cs.iitr.ac.in

****

Problem Statement 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

****

Algorithms and Data Structures Used

Data Structures Used-

→ Various forms of tree such as Binary Search Tree, AVL Tree, and Red Black Tree.

→ Nodes for the implementation of these trees.

Algorithm Used-

→ Used seperate classes for the implementation of AVL tree and Red Black Tree.

→ Class AVLTree is used for the implementation of complete AVL Tree whose data members are such as root node which is the instance of each class.

→ Class RBTree is used for the implementation of complete Red Black Tree whose data members are such as root node which is the instance of each class.

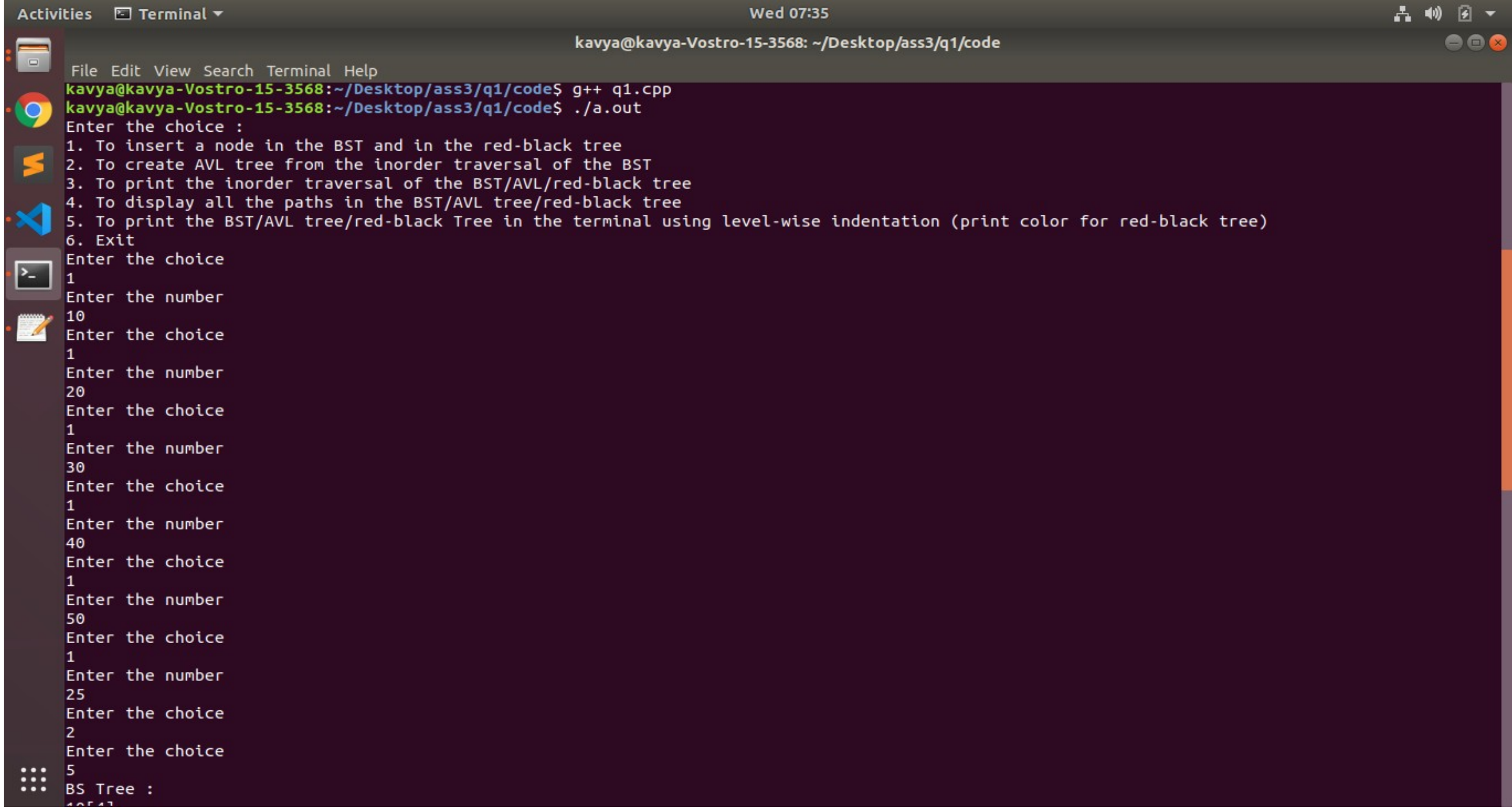
→ It also have different methods for Rotations which is used for insertion operation as per the requirements of different conditions.

→ Binary Search tree is also implemented using Nodes and all its necessary operations are implemented with it.

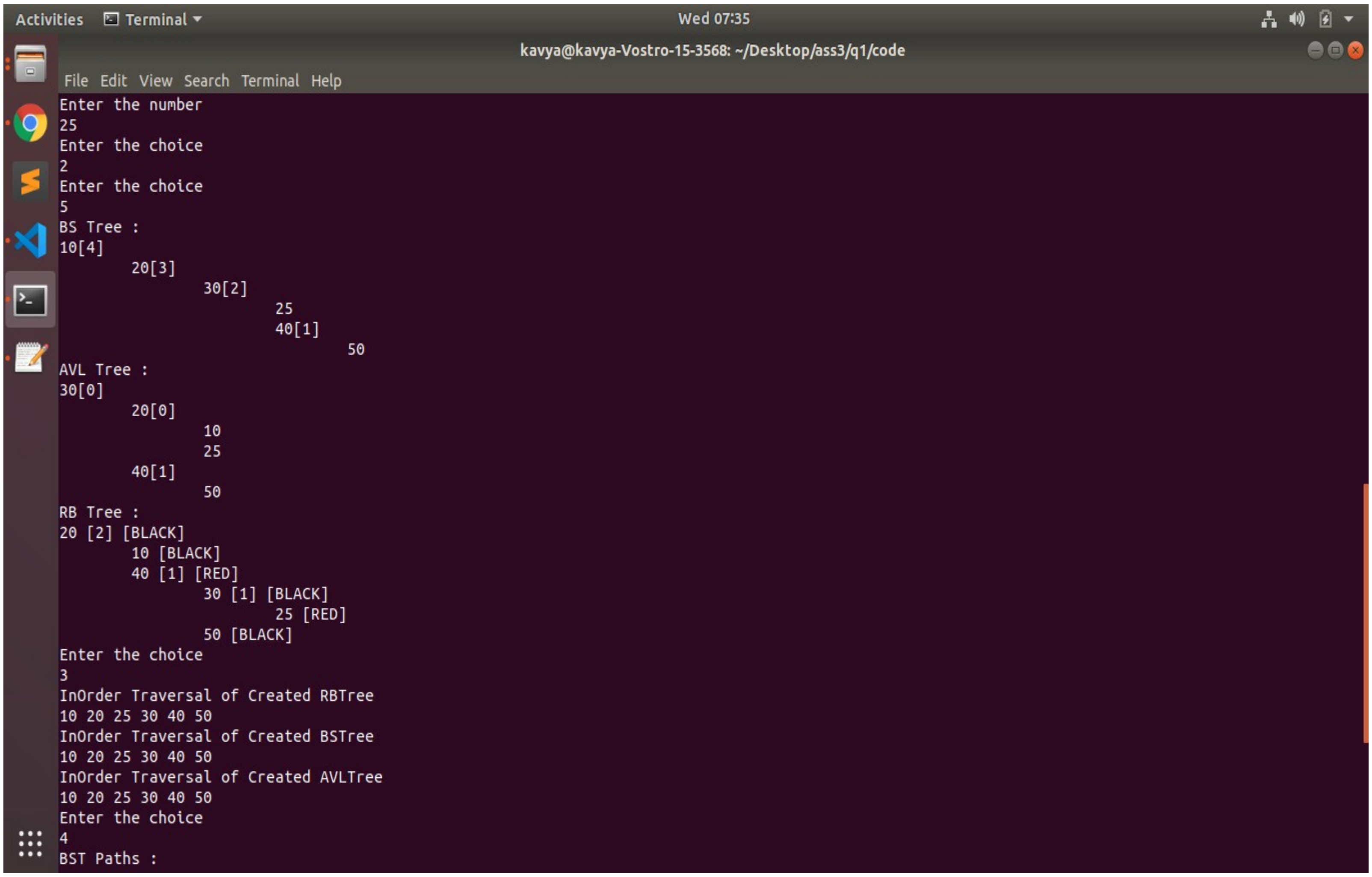
→ created seperate method for each utility such as to retrieve all the paths of different trees, and printing

the trees with proper level-wise-indentation and also
for in-order traversal in each tree.

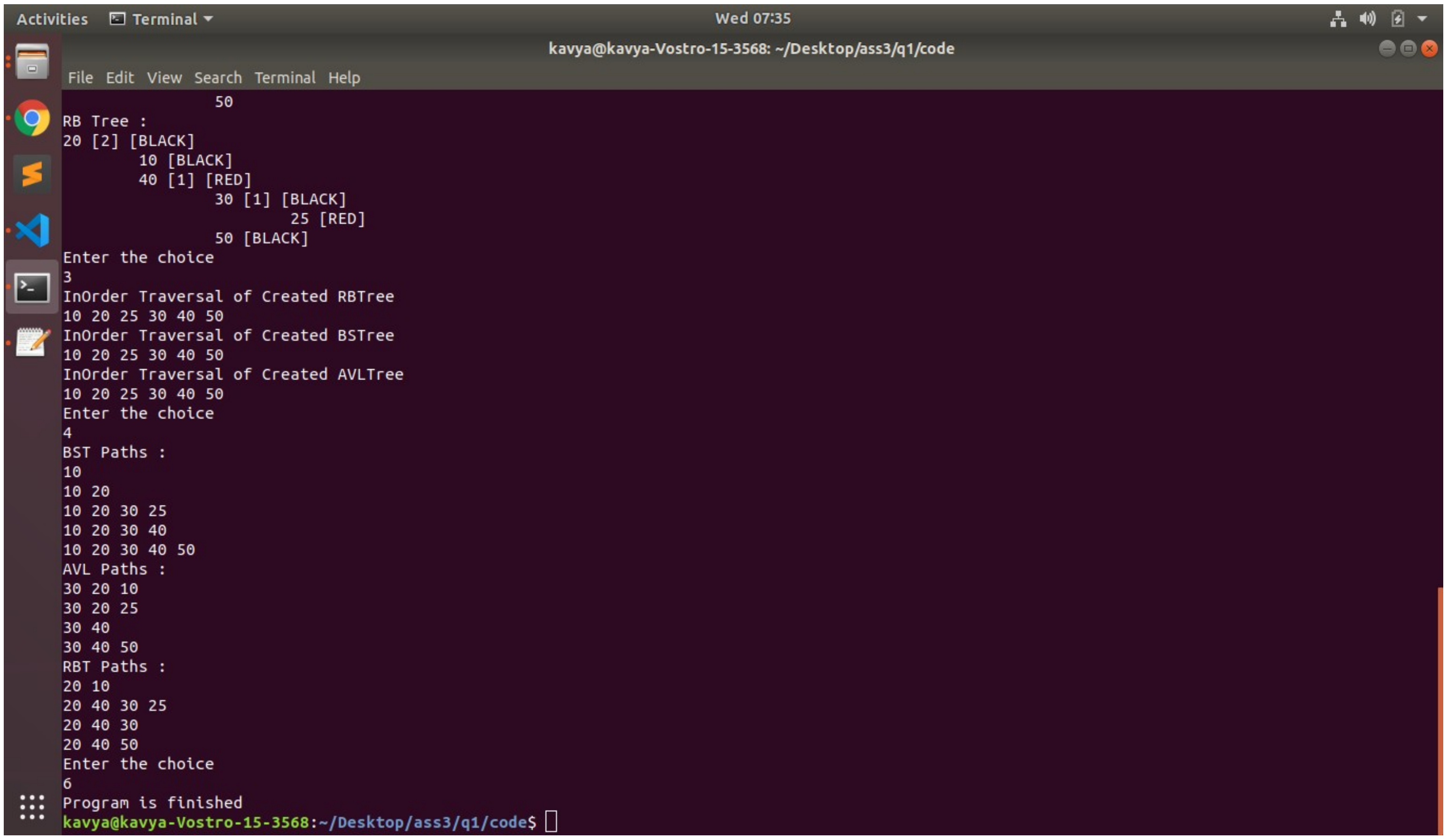
→ Similar things have been implemented for red balck
tree and AVL trees.
and return it as a string.



```
kavya@kavya-Vostro-15-3568:~/Desktop/ass3/q1/code$ g++ q1.cpp
kavya@kavya-Vostro-15-3568:~/Desktop/ass3/q1/code$ ./a.out
Enter the choice :
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
Enter the choice
1
Enter the number
10
Enter the choice
1
Enter the number
20
Enter the choice
1
Enter the number
30
Enter the choice
1
Enter the number
40
Enter the choice
1
Enter the number
50
Enter the choice
1
Enter the number
25
Enter the choice
2
Enter the choice
5
BS Tree :
```



```
Enter the number
25
Enter the choice
2
Enter the choice
5
BS Tree :
10[4]
        20[3]
                30[2]
                        25
                        40[1]
                                50
AVL Tree :
30[0]
        20[0]
                10
                25
        40[1]
                50
RB Tree :
20 [2] [BLACK]
        10 [BLACK]
        40 [1] [RED]
                30 [1] [BLACK]
                        25 [RED]
                50 [BLACK]
Enter the choice
3
InOrder Traversal of Created RBTree
10 20 25 30 40 50
InOrder Traversal of Created BSTree
10 20 25 30 40 50
InOrder Traversal of Created AVLTree
10 20 25 30 40 50
Enter the choice
4
BST Paths :
```


Activities
Terminal
Wed 07:35
kavya@kavya-Vostro-15-3568: ~/Desktop/ass3/q1/code
File Edit View Search Terminal Help

```
                50
RB Tree :
20 [2] [BLACK]
        10 [BLACK]
        40 [1] [RED]
                30 [1] [BLACK]
                        25 [RED]
                50 [BLACK]
Enter the choice
3
InOrder Traversal of Created RBTree
10 20 25 30 40 50
InOrder Traversal of Created BSTree
10 20 25 30 40 50
InOrder Traversal of Created AVLTree
10 20 25 30 40 50
Enter the choice
4
BST Paths :
10
10 20
10 20 30 25
10 20 30 40
10 20 30 40 50
AVL Paths :
30 20 10
30 20 25
30 40
30 40 50
RBT Paths :
20 10
20 40 30 25
20 40 30
20 40 50
Enter the choice
6
Program is finished
kavya@kavya-Vostro-15-3568:~/Desktop/ass3/q1/code$ 
```

****

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

Ai⊕Ai+1⊕…⊕Aj−1 = Aj⊕Aj+1⊕…⊕Ak,

where ⊕ 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)

****

Algorithms and Data Structures Used

Data Structures Used -

→ Doubly Linked List for storing the values given as input

→ Linked list to store the output values (I , j , k)

→ 2-d Arrays to store the XOR values of all input sequences.

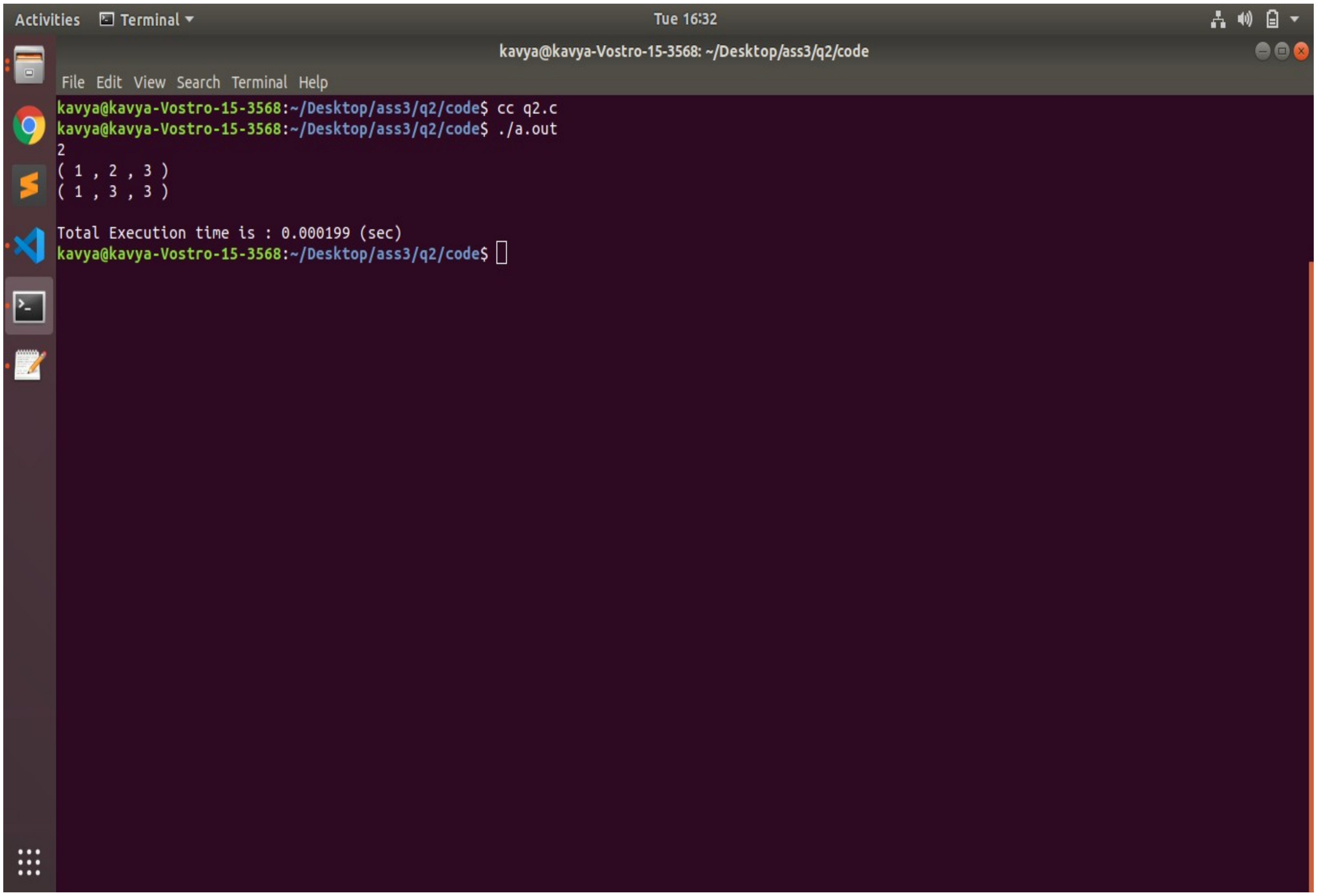
Algorithm Used-

→ 2-d Arrays used to store XOR values of different continuous sequences of given inputs, used as a look-up table.

→ XOR[i][j] dentoes the XOR of A(i) to A(j)

→ first we're checking for XOR[i][k] == 0, then searching for XOR[i][j-1]==XOR[j][k],
In this way we got our triplets (I,j,k) and store them into the linked list.

→ Used different methods for different operations such as countTriplets() which counts the required triplets for us, and XORi2l() to create the 2-d XOR Array.


Activities
Terminal
Tue 16:32
kavya@kavya-Vostro-15-3568: ~/Desktop/ass3/q2/code
File Edit View Search Terminal Help

```
kavya@kavya-Vostro-15-3568:~/Desktop/ass3/q2/code$ cc q2.c
kavya@kavya-Vostro-15-3568:~/Desktop/ass3/q2/code$ ./a.out
2
( 1 , 2 , 3 )
( 1 , 3 , 3 )

Total Execution time is : 0.000199 (sec)
kavya@kavya-Vostro-15-3568:~/Desktop/ass3/q2/code$ 
```

Total Execution Time is around 0.000119 seconds