Assignment 3 Splay Trees Test Document

Danish Sharma, Sami Ali, Tushar Dhiman
0623392, 0791752, 0757538
COIS2020 Fall 2023
December 6th, 2023

Insert

The following screenshot shows all kinds of rotations. Since the numbers inserted have a wide range the rotations are covered by this set of program executions.

Insertions: 5, 10, 70, 60, 55, 50, 2, 1

```
}

// Summary: Program class that will create the SplayTree and test
//      methods that were implemented by our team
0 references
class Program
{
    0 references
    public static void Main(string[] args)
    {
        // Creating a splay tree object and then inserting some
        //      numbers in the tree. Test documents have further
        //      testing screenshots
        SplayTree<int> tree = new SplayTree<int>();

        Console.WriteLine("INSERTING 5, 10, 70, 60, 55, 50, 2, 1");

        Console.WriteLine("\nInserted 5: " + tree.Insert(5));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 10: " + tree.Insert(10));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 70: " + tree.Insert(70));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 60: " + tree.Insert(60));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 55: " + tree.Insert(55));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 50: " + tree.Insert(50));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 2: " + tree.Insert(2));
        tree.Print();
        Console.WriteLine();

        Console.WriteLine("\nInserted 1: " + tree.Insert(1));
        tree.Print();
        Console.WriteLine();
```

```
INSERTING 5, 10, 70, 60, 55, 50, 2, 1

Inserted 5: True

InorderTraversal: 5
PreorderTraversal: 5
PostorderTraversal: 5

Inserted 10: True

InorderTraversal: 5 10
PreorderTraversal: 10 5
PostorderTraversal: 5 10

Inserted 70: True

InorderTraversal: 5 10 70
PreorderTraversal: 70 10 5
PostorderTraversal: 5 10 70

Inserted 60: True

InorderTraversal: 5 10 60 70
PreorderTraversal: 60 10 5 70
PostorderTraversal: 5 10 70 60

Inserted 55: True

InorderTraversal: 5 10 55 60 70
PreorderTraversal: 55 10 5 60 70
PostorderTraversal: 5 10 70 60 55

Inserted 50: True

InorderTraversal: 5 10 50 55 60 70
PreorderTraversal: 50 10 5 55 60 70
PostorderTraversal: 5 10 70 60 55 50

Inserted 2: True

InorderTraversal: 2 5 10 50 55 60 70
PreorderTraversal: 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2

Inserted 1: True

InorderTraversal: 1 2 5 10 50 55 60 70
PreorderTraversal: 1 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2 1
```

Remove

Using the same insertions as the Insert Method, the Remove Method works perfectly. As you can see from the screenshot after removing, the last accessed item is splayed to the root using rotations in the Splay Method and the path from the Access Method. The return values are boolean, if the removal is a success then it is true otherwise false.

Insertions: 5, 10, 70, 60, 55, 50, 2, 1

Program.cs | Program(14).cs

Assignment 3 | Program

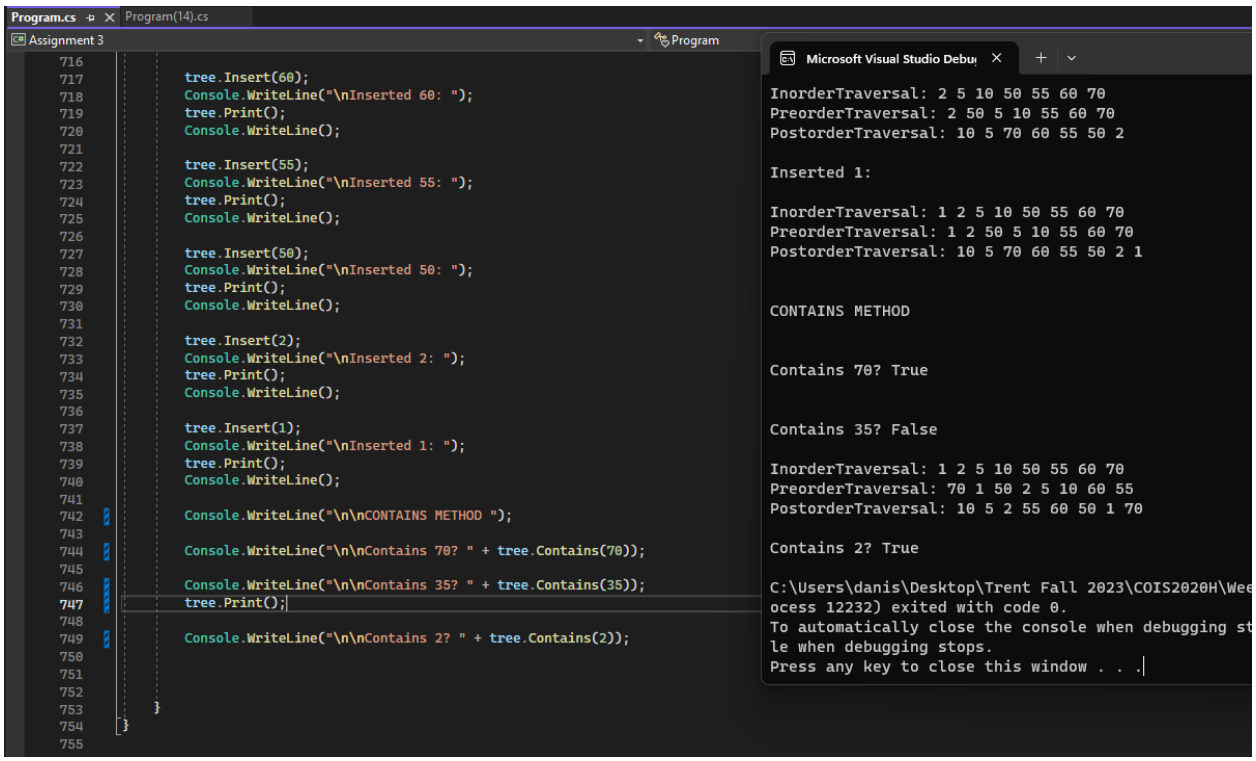
```
703            tree.Print();
704            Console.WriteLine();
705
706            tree.Insert(10);
707            Console.WriteLine("\nInserted 10: ");
708            tree.Print();
709            Console.WriteLine();
710
711            tree.Insert(70);
712            Console.WriteLine("\nInserted 70: ");
713            tree.Print();
714            Console.WriteLine();
715
716            tree.Insert(60);
717            Console.WriteLine("\nInserted 60: ");
718            tree.Print();
719            Console.WriteLine();
720
721            tree.Insert(55);
722            Console.WriteLine("\nInserted 55: ");
723            tree.Print();
724            Console.WriteLine();
725
726            tree.Insert(50);
727            Console.WriteLine("\nInserted 50: ");
728            tree.Print();
729            Console.WriteLine();
730
731            tree.Insert(2);
732            Console.WriteLine("\nInserted 2: ");
733            tree.Print();
734            Console.WriteLine();
735
736            tree.Insert(1);
737            Console.WriteLine("\nInserted 1: ");
738            tree.Print();
739            Console.WriteLine();
740
741            Console.WriteLine("\n\nREMOVING MULTIPLE NUMBERS FROM THE TREE");
742
743            Console.WriteLine("\n\nREMOVE 70? " + tree.Remove(70));
744            tree.Print();
745
746            Console.WriteLine("\n\nREMOVE 35? " + tree.Remove(35));
747            tree.Print();
748
749            Console.WriteLine("\n\nREMOVE 1? " + tree.Remove(1));
750            tree.Print();
751        }
```

```
PreorderTraversal: 55 10 5 60 70
PostorderTraversal: 5 10 70 60 55

Inserted 50:

InorderTraversal: 5 10 50 55 60 70
PreorderTraversal: 50 10 5 55 60 70
PostorderTraversal: 5 10 70 60 55 50

Inserted 2:

InorderTraversal: 2 5 10 50 55 60 70
PreorderTraversal: 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2

Inserted 1:

InorderTraversal: 1 2 5 10 50 55 60 70
PreorderTraversal: 1 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2 1


REMOVING MULTIPLE NUMBERS FROM THE TREE


REMOVE 70? True

InorderTraversal: 1 2 5 10 50 55 60
PreorderTraversal: 60 50 1 2 5 10 55
PostorderTraversal: 10 5 2 1 55 50 60

REMOVE 35? False

InorderTraversal: 1 2 5 10 50 55 60
PreorderTraversal: 10 1 5 2 60 50 55
PostorderTraversal: 2 5 1 55 50 60 10

REMOVE 1? True

InorderTraversal: 2 5 10 50 55 60
PreorderTraversal: 10 5 2 60 50 55
PostorderTraversal: 2 5 55 50 60 10
C:\Users\danis\Desktop\Trent Fall 2023\COIS2020H\Week
ocess 18136) exited with code 0.
To automatically close the console when debugging stop
le when debugging stops.
```

Contains

Using the same insertions as the Insert Method, the Contains Method checks within the tree if the item is in any of the nodes. If the item is found it is brought back to the root and the return is true. If the item is not found then the last accessed item is at the root node and the return is false.

Insertions: 5, 10, 70, 60, 55, 50, 2, 1



Splay

The Splay method is used to splay the desired node towards the root using many different types of operations. The Splay method is used in Insert, Remove, Contains, and Undo methods.

Access

The Access method is used to get the access path of a node within the tree. It uses a Stack of objects that shows which nodes were accessed to get to our desired node (target node). The Access method is used in Insert, Remove, Contains, and Undo methods.

Clone & Equals

Using the previous insertions, we made a clone of the splay tree and compared it with the original tree. As per the screenshot, the order of the trees are same as well as the Equal method result comes out to be true.

Program.cs | Program(14).cs

Assignment 3 | Program

```
706
707         tree.Insert(70);
708         Console.WriteLine("\nInserted 70: ");
709         tree.Print();
710         Console.WriteLine();
711
712         tree.Insert(60);
713         Console.WriteLine("\nInserted 60: ");
714         tree.Print();
715         Console.WriteLine();
716
717         tree.Insert(55);
718         Console.WriteLine("\nInserted 55: ");
719         tree.Print();
720         Console.WriteLine();
721
722         tree.Insert(50);
723         Console.WriteLine("\nInserted 50: ");
724         tree.Print();
725         Console.WriteLine();
726
727         tree.Insert(2);
728         Console.WriteLine("\nInserted 2: ");
729         tree.Print();
730         Console.WriteLine();
731
732         tree.Insert(1);
733         Console.WriteLine("\nInserted 1: ");
734         tree.Print();
735         Console.WriteLine();
736
737         Console.WriteLine("\n\nCLONING THE TREE AND COMPARING IT WITH EQUAL ME
738
739         SplayTree<int> clonedTree = (SplayTree<int>)tree.Clone();
740
741         Console.WriteLine("CLONED TREE ORDER: ");
742         clonedTree.Print();
743
744         Console.WriteLine("\n\nThe Cloned tree equals Original Tree? Answer:"
745
746
747
748     }
749 }
750
751
752
753
```

```
InorderTraversal: 5 10 70
PreorderTraversal: 70 10 5
PostorderTraversal: 5 10 70

Inserted 60:

InorderTraversal: 5 10 60 70
PreorderTraversal: 60 10 5 70
PostorderTraversal: 5 10 70 60

Inserted 55:

InorderTraversal: 5 10 55 60 70
PreorderTraversal: 55 10 5 60 70
PostorderTraversal: 5 10 70 60 55

Inserted 50:

InorderTraversal: 5 10 50 55 60 70
PreorderTraversal: 50 10 5 55 60 70
PostorderTraversal: 5 10 70 60 55 50

Inserted 2:

InorderTraversal: 2 5 10 50 55 60 70
PreorderTraversal: 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2

Inserted 1:

InorderTraversal: 1 2 5 10 50 55 60 70
PreorderTraversal: 1 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2 1


CLONING THE TREE AND COMPARING IT WITH EQUAL METHOD
CLONED TREE ORDER:

InorderTraversal: 1 2 5 10 50 55 60 70
PreorderTraversal: 1 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2 1

The Cloned tree equals Original Tree? Answer:True
```

Undo

The Undo() method is a complex structure that requires us to perform reverse rotations on the root node to get it to the bottom of the tree aka the leaf node. The following screenshots show the insertion of numbers, undoing a single insertion and then re-inserting the number. Using the Equals method, it can be verified that the trees are the same.

Insertion: 10, 20, 5, 30, 40, 50

```
// Summary: Program class that will create the SplayTree and test
//      methods that were implemented by our team
0 references
class Program
{
    0 references
    public static void Main(string[] args)
    {
        SplayTree<int> tree = new SplayTree<int>();

        Console.WriteLine("INSERTING 10, 20, 5, 30, 40, 50");

        tree.Insert(10);
        tree.Insert(20);
        tree.Insert(5);
        tree.Insert(30);
        tree.Insert(40);
        tree.Insert(50);

        SplayTree<int> clonedTree = (SplayTree<int>)tree.Clone();

        tree.Print();

        Console.WriteLine("\n\nUNDO 50 AND THEN RE-INSERTING 50 BACK IN, RESULT: ");

        tree.Undo();

        tree.Insert(50);

        tree.Print();

        Console.WriteLine("\n\nIs the Undo method working properly? " + tree.Equals(clonedTree));

    }
}
```

```
Microsoft Visual Studio Debu

INSERTING 10, 20, 5, 30, 40, 50

InorderTraversal: 5 10 20 30 40 50
PreorderTraversal: 50 40 30 5 20 10
PostorderTraversal: 10 20 5 30 40 50

UNDO 50 AND THEN RE-INSERTING 50 BACK IN, RESULT:

InorderTraversal: 5 10 20 30 40 50
PreorderTraversal: 50 40 30 5 20 10
PostorderTraversal: 10 20 5 30 40 50

Is the Undo method working properly? True

C:\Users\danis\Desktop\Trent Fall 2023\COIS2020H\We
ocess 24100) exited with code 0.
To automatically close the console when debugging s
le when debugging stops.
Press any key to close this window . . .
```

Insertion: 5, 10, 70, 60, 55, 50, 2, 1

```
// Summary: Program class that will create the SplayTree and test
//      methods that were implemented by our team
0 references
class Program
{
    0 references
    public static void Main(string[] args)
    {
        SplayTree<int> tree = new SplayTree<int>();

        Console.WriteLine("INSERTING 5, 10, 70, 60, 55, 50, 2, 1");

        tree.Insert(5);
        tree.Insert(10);
        tree.Insert(70);
        tree.Insert(60);
        tree.Insert(55);
        tree.Insert(50);
        tree.Insert(2);
        tree.Insert(1);

        SplayTree<int> clonedTree = (SplayTree<int>)tree.Clone();

        tree.Print();

        Console.WriteLine("\n\nUNDO 1 AND THEN RE-INSERTING 50 BACK IN, RESULT: ");

        tree.Undo();

        Console.WriteLine("\n\nTREE AFTER UNDO: ");
        tree.Print();

        Console.WriteLine("\n\nRE-INSERTING 50 BACK IN, RESULT: ");

        tree.Insert(1);

        tree.Print();

        Console.WriteLine("\n\nIs the Undo method working properly? " + tree.Equals(clonedTree));

    }
}
```

```
Microsoft Visual Studio Debu

INSERTING 5, 10, 70, 60, 55, 50, 2, 1

InorderTraversal: 1 2 5 10 50 55 60 70
PreorderTraversal: 1 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2 1

UNDO 1 AND THEN RE-INSERTING 50 BACK IN, RESULT:


TREE AFTER UNDO:

InorderTraversal: 2 5 10 50 55 60 70
PreorderTraversal: 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2

RE-INSERTING 50 BACK IN, RESULT:

InorderTraversal: 1 2 5 10 50 55 60 70
PreorderTraversal: 1 2 50 5 10 55 60 70
PostorderTraversal: 10 5 70 60 55 50 2 1

Is the Undo method working properly? True

C:\Users\danis\Desktop\Trent Fall 2023\COIS2020H\Week 12\Assi
ocess 29872) exited with code 0.
To automatically close the console when debugging stops, enab
le when debugging stops.
Press any key to close this window . . .
```