Lam Dang
Lab 2: Splay Tree
Writeup

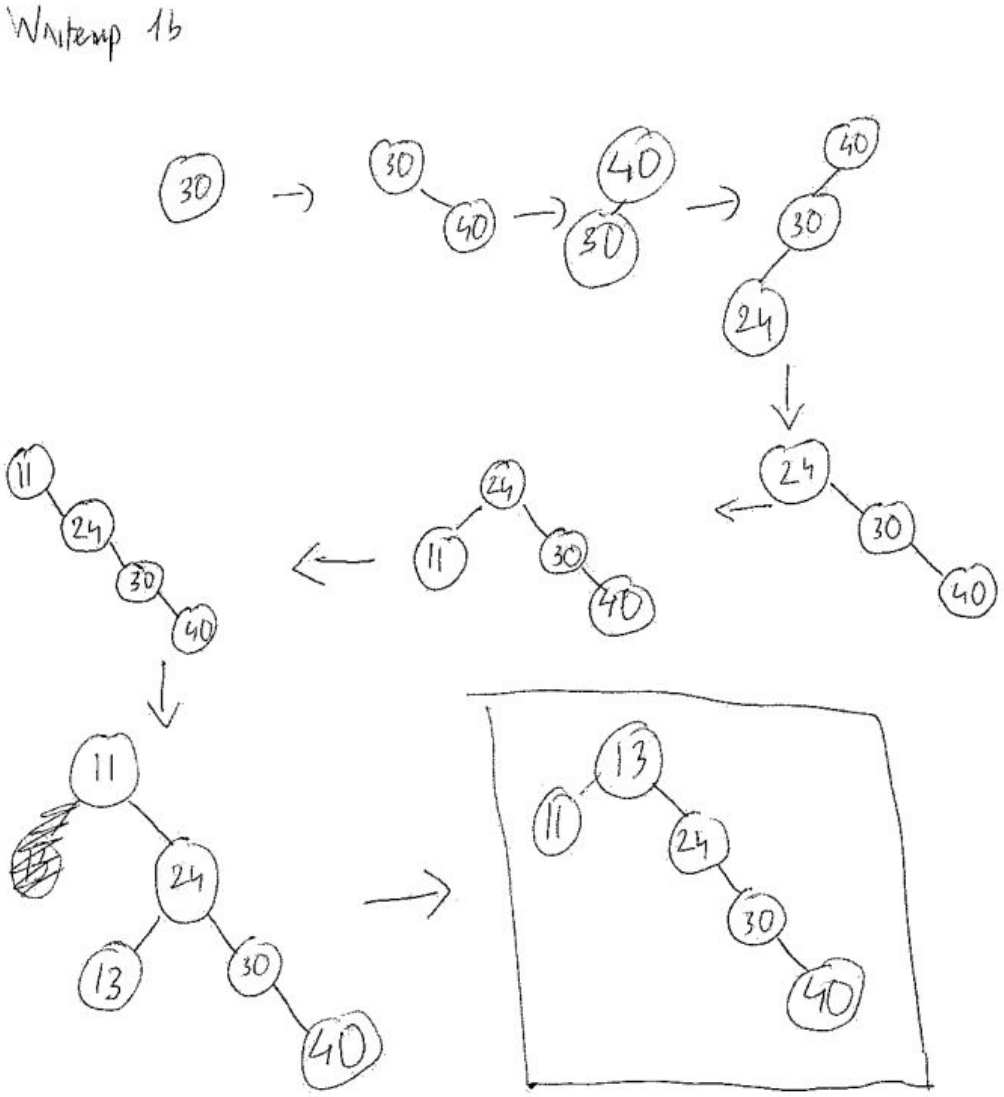
Part 1:
Writeup #1a:
Splay Tree is a self-balancing Binary Search Tree with rotation methods that allow quick access to recently-used nodes.

Splaying is a rotation method that uses a series of rotation (Zig and Zag) to rotate accessed nodes to the root of the tree. For every method of insertion, deletion and search, we use splay to rotate the nodes.

Splay tree has the worst case of Amortized O(logN) while BST has the worst case of O(N). Also, repeated access in a Splay tree will promise a much faster access time than of a Binary Search Tree.

Splay Tree is not always better than AVL tree because AVL tree access time has a worse case of O(logN) and Splay Tree access time has the worse case of O(n)

Writeup #1b



Part 2:

In the example, you can see that Splay Tree has the worse insertion time. This is because of the multiple rotations they have to do while splaying. BST has the best Insertion because it is the only case where rotation is not used

However, The Splay Tree is the best in look-up time. This is because of multiple operations that were operated on the Lam node, causing it to be pushed to the top, which is easier and faster to access.

Part 3:

A network router: The network router receives a lot of requests of IP addresses. If an IP address was used recently, it is highly likely that it will be used over and over again in the short future. So it makes sense to keep them on top of the splay tree for easy access.

Cache: Cache is a component to store data so that for future requests, data will be fetched and loaded faster and more efficiently. Since resources that were used before are likely to be accessed again in the future, we use a splay tree to store, access and delete these resources.

For Unittest Purpose:

Figure 1.1. SetUp Tree

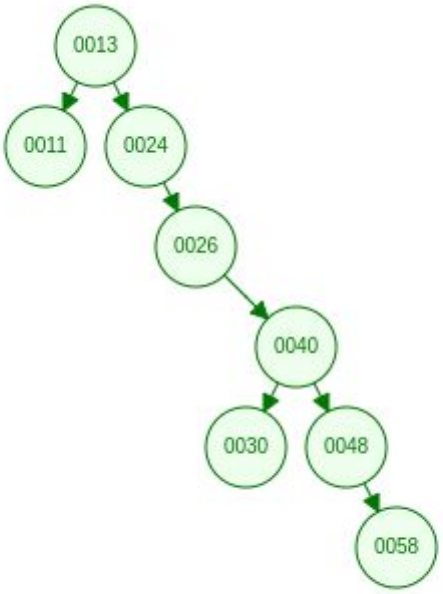
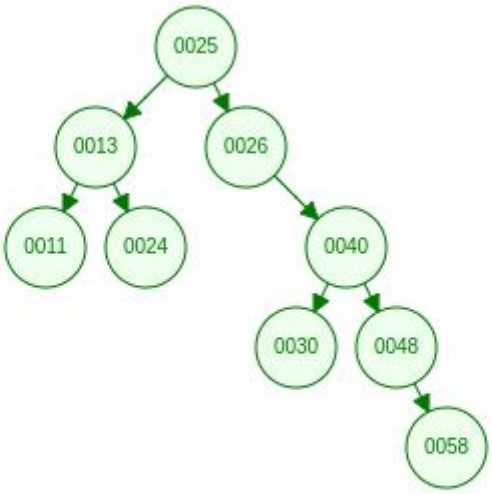


Figure 1.2. Insertion of 25


0025
0013
0026
0011
0024
0040
0030
0048
0058