CMSC 420: Coding Project 4
Splay Trees

1 Due Date and Time

Due to Gradescope by Sunday 12 November at 11:59pm. You can submit as many times as you wish before that.

2 Get Your Hands Dirty!

This document is intentionally brief and much of what is written here will be more clear once you start looking at the provided files and submitting.

3 Assignment

We have provided the template `splay.py` which you will need to complete. More specifically you will fill in the code details to manage insertion, deletion, and search for a splay tree.

As with the B-tree project we have implemented a `SplayTree` class as well as a `Node` class. The `SplayTree` class stores only a pointer to the root `Node`. The `insert`, `delete`, and `search` functions are then implemented as methods of the `SplayTree` class.

Note that the `Node` class includes a parent pointer which you will need to maintain and the `dump` method prints out the parent key (except for the root).

Please look at this file as soon as possible.

4 Details

The functions should do the following:

- `def insert(self, key: int):`

  Insert the key into the tree using the first method discussed in class. The key is guaranteed not to be in the tree.

- `def delete(self, key: int):`

  Delete the key from the tree using the first method discussed in class and using the right subtree if neither subtree is empty. The key is guaranteed to be in the tree.

- `def search(self, key: int):`

  Search for the key in the tree using the method discussed in class. This will splay the tree but does not return or print anything. The key may or may not be in the tree.

5 Additional Functions

You will probably want some helper functions as well as `SplayTree` class methods. Up to you.

6 Suggestion

I suggest first copying your code which inserts and deletes as with a standard BST, this way you can build some predictable trees to test.

Then I suggest copying your rotation functions from the AVL tree project and modifying them so they are entirely self-contained in terms of modifying the tree. By this I mean they should modify all the correct pointers and should not (need to) return anything. This will make the splay operation significantly easier to write and check.

Once your rotations are working, then write the splay-specific code.

7 What to Submit

You should only submit your completed `splay.py` code to Gradescope for grading. We suggest that you begin by uploading it as-is (it will run!), before you make any changes, just to see how the autograder works and what the tests look like. Please submit this file as soon as possible.

8 Testing

This is tested via the construction and processing of tracefiles.

- Each non-final line in a tracefile is either `insert,key` or `delete,key`, or `splay,key`. All together these lines result in the creation of a Splay Tree.
- The final line will always be `dump`.

You can see some examples by submitting the `splay.py` file as-is.

9 Local Testing

We have provided the testing file `test_splay.py` which you can use to test your code locally. Simply put the lines from a tracefile (either from the autograder or just make one up) into a file `whatever` and then run:

```
python3 test_splay.py -tf whatever
```