CMSC 420: Coding Project 3
Generalized Scapegoat Trees

1 Due Date and Time

Due to Gradescope by Sunday 24 March 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 `scapegoat.py` which you will need to complete. More specifically you will fill in the code details to manage various aspects of generalized scapegoat trees.

Note: The standard scapegoat tree uses $\alpha = 2/3$ but the generalized version may be initialized with $\alpha = a/b$ where $1/2 < a/b < 1$. This α value impacts the situation several ways:

- Definition of a scapegoat.
- Depth check for insertion.
- Restructure check for deletion.

As with the B-tree project we have implemented a `SGtree` class as well as a `Node` class. The `SGtree` class stores the a and b values as well as a pointer to the root `Node`. The various functions are then implemented as methods of the `SGtree` class.

As is noted in the `scapegoat.py` template you are welcome to make minor augmentations to the `Node` class. For example we have included a `parent` pointer but you can delete this if you don't use it.

Please look at this file as soon as possible.

4 Details

The functions should do the following:

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

  Insert the key,value pair into the tree and rebalance as per scapegoat tree rules. The key is guaranteed not to be in the tree.

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

  Delete the key,value pair from the tree and rebalance as per scapegoat tree rules. The key is guaranteed to be in the tree.

- `def search(self, search_key: int) -> str:`

  Calculate the list of values on the path from the root to the search key, including the value associated to the search key. Return the json stringified list. The key is guaranteed to be in the tree.

5 Additional Functions

You will probably want some helper functions as well as `SGtree` class methods to handle the necessary operations.

6 What to Submit

You should only submit your completed `scapegoat.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.

7 Testing

This is tested via the construction and processing of tracefiles.

- The first line in the tracefile is `initialize,a,b` which should initialize an instance of the `SGtree` class with $\alpha = a/b$ value using `a` and `b` and with root node `None`.
- Each remaining non-final line in a tracefile is either `insert,key,value` or `delete,key`. All together these lines result in the creation of a scapegoat tree.
- The final line is either `dump` or `search,key`.

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

8 Local Testing

We have provided the testing file `test_scapegoat.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_scapegoat.py -tf whatever
```