CMSC 420: Coding Project 5
Skip Lists

1 Due Date and Time

Due to Gradescope by Sunday 28 April 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 `skiplist.py` which you will need to complete. More specifically you will fill in the code details to manage various aspects of a **slight variation** of a skip list. More details are given below.

4 Details

The class methods should do the following:

- `def insert(self,key,value,toplevel):`

  Create a node with `key` and `value` into the skip list. The node should have `toplevel` as its top level. The key is guaranteed not to be in the skip list.

  You are to do this slight variation: After insertion if the expected maximum level for the number of nodes in the skip list is greater than the skip list's maximum enforced level then rebuild the entire skip list as follows:

  Double the skip list's maximum enforced level and rebuild it as a perfect skip list. For perfection we assume that the nodes (not counting the header node) are 1-indexed and then:

  - Every node certainly reaches level 0.
  - Nodes 2,4,6,8,... also reach level 1.
  - Nodes 4,8,12,16,... also reach level 2.
  - Nodes 8,16,24,32,... also reach level 3.
  - Etc. with the added condition that no node goes above the new maximum enforced level.

- `def delete(self,key):`

  Delete the node with `key` from the skip list. The key is guaranteed to be in the skip list.

- `def search(self,key) -> str:`

  Search for the node with given `key`. Return a `json.dumps` list consisting of all keys in all nodes visited during the search and with the associated `value` appended at the end. The key is guaranteed to be in the skip list.

5 Additional Functions

You might want some additional functions as well as helper functions to handle the necessary operations. However this project is straightforward and shouldn't need any.

6 What to Submit

You should only submit your completed `skiplist.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 as with the previous projects.

- The first line in the tracefile is `initialize,m`. This initializes an instance of the `skiplist` class, setting up head and tail nodes with maximum enforced level `m`. Note that this initialization function has been written for you.
- Each remaining non-final line in a tracefile is either `insert,key,value,toplevel` or `delete,key`.
- The final line is either `dump`, which dumps the skip list, or `search,key`.

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

8 Local Testing

We have provided the testing file `test_skiplist.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_skiplist.py -tf whatever
```