I&C SCI 46 Fall 2022 Project 2

Due October 19, 2022 at 7:30 AM. This project is eligible for late submissions.

Introduction

In lecture, we discussed Skip Lists. In this project, we're going to implement one. You will discover that there are some details you uncover as you work on this that aren't apparent when you analyze the concept of a Skip List.

Requirements

- Fill in SkipList.hpp
  - Your implementation must be a linked structure.
  - Your implementation must fit the interface given.
  - Your implementation must be templated
    - An implementation of SkipList that does not properly template may cause you to get a zero on this assignment. **Do not assume that working for unsigned is sufficient, despite those being the only cases in the provided test cases.**
    - Your project needs to be able to handle Key items of any type that have the < operator defined, a default constructor, an assignment operator, and for which the flipCoin() function is defined. For purposes of the assignment, **unsigned integers** and **std::string** are the only two such types you need to worry about
    - **No, really; pay attention to that warning. Students may end up with zeroes for this project if their code is not properly templated, regardless of other considerations.**
  - When you insert a key/value pair into your Skip List and flip a coin to determine higher levels, use the function **flipCoin**. This is described at the top of SkipList.hpp This returns true if the coin flip returned heads (meaning add a layer). The "bubbling up" procedure should be halted if the addition of a new layer would violate any of the following conditions:
    - For a skip list with **16 or fewer** elements (including the element that was just added in the bottom-most lane) the maximum number of layers is 13. This includes the "fast" lane (S_h, where h is the height of the skip list) and the "base" lane S_0, see section 9.4 in the textbook for further elaboration on this distinction. See `SimpleHeightsTest` in `SkipTests.cpp` for a concrete example.
    - For a skip list with **more than 16** elements the maximum number of layers is `3 * ceil(log2(n)) + 1`. In other words, the skip list should have a *height* of `3 * ceil(log2(n))` in the worst case. The library `cmath` has a function for logarithms and ceiling, and you may use this

for the assignment. See `Capacity17Test` in `SkipTests.cpp` for a concrete example of this.

- The functions `height()`, `nextKey()`, `previousKey()`, `find()`, `isSmallestKey()`, and `isLargestKey()` should throw a `RunimeException` (provided in `runtimeexcept.hpp`) if the skip list does not contain the passed key *k*. More details are provided in `SkipList.hpp` for the implementations of these functions.
- **Please note:** the project will not build successfully until you have at least stubbed code for the public member functions.

- Your implementation does not have to be the most efficient thing ever, but it cannot be "too slow." In general, any test case that takes over a minute on the grader's computer may be deemed a wrong answer, even if it will later return a correct one.

You are prohibited from using any standard library container classes. Using a standard library container, including `std::vector`, in implementing your SkipList will be treated as a serious academic integrity violation. The *only* exception to this is in the function allKeysInOrder(). However, in that function, you must create the vector during your implementation.

Can I submit after the deadline?
Yes, it is possible, subject to the late work policy for this course, which is described in the section titled Late work in the course reference and syllabus.