Program 12 B-tree

2024/6/17

40 Points Possible

Attempt 1

In Progress

NEXT UP: Submit Assignment

Add Comment

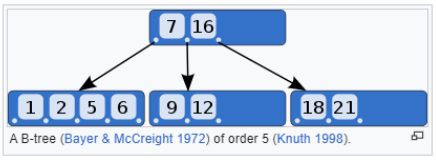
Unlimited Attempts Allowed

2024/6/10

Details

Overview

B-Tree (https://en.wikipedia.org/wiki/B-tree), simplified to zyBooks 2-3-4 Tree, just using int data type.



In this lab, the Tree234 class is extended to support iteration with a range-based for loop. Iteration support is provided via the implementation of an iterator that can iterate through the tree's keys in ascending order.

An iterator is an object that maintains a pointer to a specific element in a collection and can move to the next element. Ex: A Tree234 iterator points to the tree's minimum key upon construction. The iterator can then move to the second to minimum key, then the third to minimum, and so on. After moving past the tree's last key, the iterator can move no further.

Overview of Iterable objects in C++

This lab requires implementation of a simplified iterator that:

- Implements the dereference operator (*) to return the key that the iterator currently points to
- Implements the pre-increment operator (++) to advance to the next key
- Supports copy construction and copy assignment
- Supports equality and inequality comparison

Range-based for loops work on any class that implements the begin() and end() member functions such that each returns an iterator. So Tree234 implements begin() and end(). Tree234's begin() member function returns an instance of Tree234Iterator object representing the inclusive starting point of iteration: the tree's minimum key. Tree234's end() member function returns an instance of Tree234Iterator object representing the exclusive ending point of iteration: one beyond the tree's maximum key.

Step 1: Inspect the **Node234.h (https://bc.instructure.com/courses/2442359/files/249137559?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137559/download?download_frd=1)** file and the **Tree234.h (https://bc.instructure.com/courses/2442359/files/249137562?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137562/download?download_frd=1)** file

Node234.h has a complete implementation of a Node234 class for a 2-3-4 tree node. Member variables are protected and so must be accessed through the provided getter and setter functions.

Step 2: Inspect the **Tree234Iterator.h (https://bc.instructure.com/courses/2442359/files/249137564?wrap=1)**

(https://bc.instructure.com/courses/2442359/files/249137564/download?download_frd=1)

file

The Tree234Iterator class is declared, but required member functions are not implemented. The implementation would optimally satisfy the following requirements:

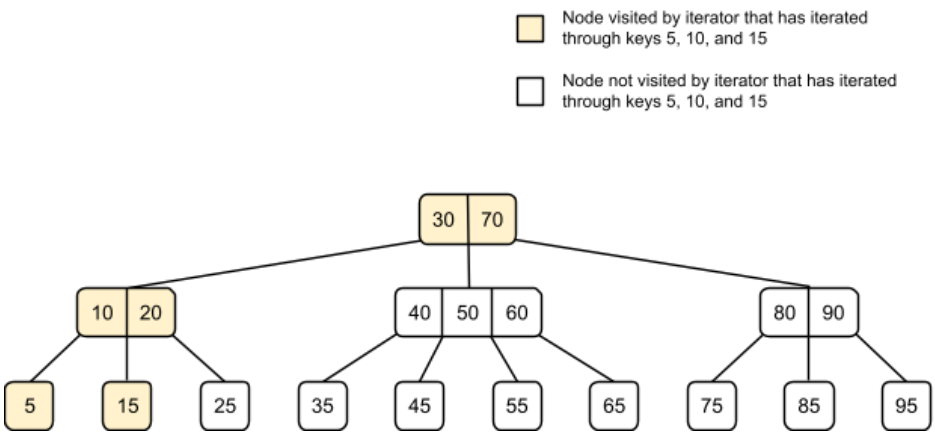
- Iteration never changes the tree in any way.
- Iteration starts at the tree's minimum key and ends at the maximum.
- Construction occurs in worst-case O(log N) time.
- Dereferencing executes in worst-case O(1) time.
- Incrementing executes in worst-case O(log N) time.
- Space complexity is worst-case O(log N).

For simplicity, assume the tree is not changed by an outside source during the iterator's lifetime.

Step 3: Understand requirement implications

To satisfy the requirements, the iterator must maintain a collection of node pointers. A node exists in the collection only if that node must be revisited at some point in time.

The iterator must visit only the necessary nodes to deliver a key when the iterator is dereferenced. "Visiting" a node means calling any of that node's member functions. Ex: Suppose an iterator is built for the tree below. Then the iterator's * operator is invoked to return key 5, then ++ is invoked to advance to key 10, then * is invoked to return key 10, then ++ is invoked to advance to key 15, and then * is invoked to return key 15. The iterator should have only visited the highlighted nodes.



Step 4: Implement the Tree234Iterator class

Implement the Tree234Iterator to satisfy the complexity requirements mentioned above. Code in **main.cpp (https://bc.instructure.com/courses/2442359/files/249137566?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137566/download?download_frd=1)** adds random keys to a Tree234 object, then tests that the iterator properly iterates through all keys in ascending order. But time and space complexity aren't tested by code in main.cpp. Rather, main.cpp only ensures that the iterator properly iterates through all keys.

Tests will fail if the iterator does not properly iterate through all the tree's keys in the correct order. So run code and ensure that the test in main.cpp passes before submitting code.

Starter code:

- **main.cpp (https://bc.instructure.com/courses/2442359/files/249137566?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137566/download?download_frd=1)**
- **Node234.h (https://bc.instructure.com/courses/2442359/files/249137559?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137559/download?download_frd=1)**
- **Tree234.h (https://bc.instructure.com/courses/2442359/files/249137562?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137562/download?download_frd=1)**
- **Tree234Iterator.h (https://bc.instructure.com/courses/2442359/files/249137564?wrap=1)** **(https://bc.instructure.com/courses/2442359/files/249137564/download?download_frd=1)**

SUBMIT: Tree234Iterator.h

Do not modify the other 3 files (main, Node234, Tree234) as I will use those as provided for final grading.

Choose a submission type

Upload | More

Choose a file to upload

File permitted: H

or

Canvas Files

(https://bc.instructure.com/courses/2442359/modules/items/83921175)

(https://bc.instructure.com/courses/2442359/modules/items/8