Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Лабораторна робота №2

Виконав: Олексишин Олексій

Група К-28

Кафедра інтелектуальних програмних систем

**2020**

**Предметна область**

Вариант 3

Предметная область Отдел кадров Объекты Подразделения, Сотрудники Примечание Имеется множество подразделений предприятия. В каждом подразделении работает множество сотрудников.

Для того щоб дану структуру даних можна було помістити в дерево я перевантажив operator< (та інші оператори порівняння), для виведення в консоль operator<<.

**Завдання**

Реалізуйте дерево порядкової статистики на основі червоно-чорного дерева.

**Алгоритм і теорія**

**1) Червоно-чорне дерево**

Червоно-чорне дерево – балансоване бінарне дерево пошуку, для якого виконуються настпуні умови:

* кожна вершина або червона, або чорна
* корінь дерева — чорний
* кожний лист (nullptr) — чорний
* якщо вершина червона, обидві її дочірні вершини чорні (інакше, батько червоної вершини — чорний)
* усі прості шляхи від будь-якої вершини до листів мають однакову кількість чорних вершин

Всі операції з червоно чорним деревом аналогічні до операцій з звичайним бінарним деревом пошуку, але після кожної вставки/видалення дерево потребує додаткових змін, які зберігають виконання вищевказаних умов. Це реалізується за допомогою процедур remove\_fix, insert\_fix та поворотів rotate\_left, rotate\_right.

**2) Дерево порядкової статистики**

Дерево порядкової статистики – дерево в якого можна отримати k-ий найбільший елемент за O(log(n)) часу. Для цього в кожну вершину червоно-чорного дерева треба додати ще один атрибут – розмір піддерева, утвореного цією вершиною. Для підтримки валідності цього поля, в поворотах треба додати зміну цього атрибута для вершин, які ми повертаємо.

Тепер, коли ми знаємо розмір піддерева, знайти k-ий елемент можна так: беремо вершину, дізнаємося розмір лівого піддерева – left->size. Якщо left->size > k - 1 повторюємо рекурсивно дану операцію з лівим сином і цим самим k. Якщо left->size == k – 1, то шуканий елемент лежить у вершині дерева, повертаємо його. Інакше повторюємо рекурсивно наш алгоритм для правого піддерева і k’ = k – left→size – 1.

**Складність алгоритму**

Операції вставки, пошуку елемента, видалення та балансування дерева виконуються за O(log(n)) часу.

Операція пошуку k-ого по порядку елемента O(log(n)) часу – ми рекурсивно пройдемо щонайбільше від вершини до листа, тобто O(h) = O(log(n)), оскільки дерево балансоване.

**Реалізація (мова С++)**

Програма реалізує класи RedBlackTree та RedBlackTreeNode, які мають методи вказані в описі алгоритму (insert, remove, search, operator[] – реалізує пошук k-ого елемента, delete\_fix, insert\_fix, left\_rotate, right\_rotate).

**Основні модулі програми**

У файлі RedBlackTree.hpp показаний інтерфейс вказаних класів, а в файлах RedBlackTree.tpp, RedBlackTreeNode.tpp реалізація методів класів для роботи з ними. Файл struct.hpp містить реалізацію класів предметної області. В файлі main.cpp міститься інтерфейс користувача та приклад роботи з реалізованою структурою даних.

Код можна знайти в моєму репозиторії на github за посиланням <https://github.com/ooleksyshyn/semester4/tree/master/algorithms/lab2>

**Інтерфейс користувача та приклади**

Програма дозволяє взаємодіяти з класом через простий інтерфейс – вставка, видалення, пошук елемента за ключем та за індексом. В коді є наведений приклад використання програми.

**Літературні джерела**

* [https://uk.wikipedia.org/wiki](https://uk.wikipedia.org/wiki/Червоно-чорне_дерево)/Червоно-чорне\_дерево
* <https://ru.wikipedia.org/wiki/Порядковая_статистика>
* <https://github.com/ooleksyshyn/semester4/tree/master/algorithms/lab2>