Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Лабораторна робота №5

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 персистентну множину 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**

Персистентна множина – множина яка зберігає в собі всі свої попередні стани після змін. Для цього можна було б просто зберігати кожну версію як копію і при змінах в множині, просто створювати нову копію, робити зміни з нею та зберігати. Проте, такий підхід використовує O(n) додаткової пам’яті для кожної версії, що дуже неефективно.

Нова ідея – реалізувати персистентну множину на основі дерева, для кожної нової версії зберігати новий корінь та копіювати тільки ті вершини, піддерево яких було змінене. Оскільки за одну операцію вставки/видалення ми змнінюємо 1 елемент, то для створення нової версії дерева нам треба буде копіювати всіх предків зміненої вершини, а таких не більше ніж log(n) (n – кількість елементів в множині), оскільки дерево балансоване.

Отже, для реалізації персистентної множини на основі червоно чорного дерева достатньо просто копіювати потрібні елементи під час змін.

**Складність алгоритму**

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

Оскільки ми копіюємо не більше, ніж log(n) вершин, то затрати в пам’яті будуть O(k \* log(n)), де k – кількість версій, тобто кільксть разів коли ми щось додавали чи видаляли з дерева, n – максимальна кількість елементів в множині з цих k версій.

**Реалізація (мова С++)**

Програма реалізує клас PersistentSet, що містить в собі адаптовану реалізацію червоно-чорного дерева, методи для роботи з множиною та звичайний інтерфейс для роботи з множиною (insert, remove, search) та процедури для перебудови дерева.

**Основні модулі програми**

У файлі PersistentSet.hpp показаний інтерфейс вказаних класів, та реалізовані методи класів та процедури для роботи множиною та червоно-чорним деревом. 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/lab5>

**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://en.wikipedia.org/wiki/Red-black\_tree](https://en.wikipedia.org/wiki/Red–black_tree)