Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

**Лабораторна робота №2**

**Завдання №2.2**

**Дерево порядкової статистики**

**(на основі червоно-чорного дерева)**

**Тип даних – Т9 (Дійсні числа)**

Виконав:

студент групи ІПС-23

факультету комп’ютерних наук

та кібернетики

Кирпотенко Юрій Володимирович

Київ 2023

**Зміст**

|  |  |
| --- | --- |
| Теоретичні відомості | 3 |
| Алгоритм | 3 |
| Складність | 4 |
| Мова програмування | 4 |
| Модулі програми | 5 |
| Інтерфейс користувача | 6 |
| Приклади | 6 |
| Висновок | 7 |
| Література | 7 |
|  |  |

**Теоретичні відомості**

Дерево порядкової статистики - це дерево, яке забезпечує можливість ефективного пошуку k-го елементу в масиві відсортованих елементів. Воно реалізується на основі червоно-чорного дерева, яке має певні властивості балансування та швидкого доступу до елементів.

Кожен вузол червоно-чорного дерева містить дані про елемент масиву, який зберігається в цьому вузлі, а також додаткову інформацію про розмір піддерева, що коренем є даний вузол. Дерево порядкової статистики забезпечує реалізацію таких методів, як вставка елемента, пошук порядкової статистики, пошук найменшого та найбільшого елементів, видалення елемента тощо.

**Алгоритм**

1. Починаємо з кореня дерева.
2. Запам'ятовуємо розмір піддерева, що є коренем є поточного вузла (включаючи корінь).
3. Якщо номер порядкової статистики, яку ми шукаємо, менший за розмір лівого піддерева поточного вузла, то переходимо до лівого сина поточного вузла і повторюємо кроки 2 і 3.
4. Якщо номер порядкової статистики, яку ми шукаємо, дорівнює розміру лівого піддерева поточного вузла, то повертаємо значення, що зберігається в поточному вузлі.
5. Якщо номер порядкової статистики, яку ми шукаємо, більший за розмір лівого піддерева поточного вузла, то віднімаємо розмір лівого піддерева від номера порядкової статистики і переходимо до правого сина поточного вузла, повторюючи кроки 2 і 3.
6. Якщо поточний вузол є листком, то повертаємо значення, що зберігається в цьому вузлі.

**Складність**

Вставка елемента: складність вставки в дерево порядкової статистики на основі червоно-чорного дерева є O(log n), де n - кількість елементів у дереві. Це завдяки тому, що червоно-чорне дерево гарантує балансування дерева, що забезпечує глибину дерева не більше O(log n).

Фіксація дерева після вставки: фіксація дерева може знадобитися після кожної вставки, щоб забезпечити, що дерево залишається червоно-чорним. Складність фіксації дерева також є O(log n), оскільки максимальна глибина дерева дорівнює O(log n).

Пошук порядкової статистики: складність пошуку k-го елемента в порядку зростання є O(log n), де n - кількість елементів у дереві. Це завдяки тому, що ми обходимо дерево вгору та/або вниз, поки не знайдемо k-ий елемент.

Обертання вліво та вправо: обертання вліво та вправо, які використовуються для фіксації дерева після вставки, також мають складність O(log n), оскільки глибина дерева є O(log n).

**Мова програмування**

С# 11.0

Cередовище - Visual Studio 2022

**Модулі програми**

***Class Program – точка входу в програму***

\* Оскільки цей клас відповідає за запуск алгоритму, у ньому ми отримуємо вхідні дані, створюємо, після чого виводимо результат.

static void Main(string[] args) – точка входу програми

static bool FillCollection(out RedBlackTree? tree) – розбиття вводу на дійсні числа, заповнення дерева

***Class Node – вузол дерева***

\* Клас-утримувач вузла дерева

Не має у собі методів, та тільки тримає в собі інформацію щодо вузла у наступних змінних:

public float data;

public Color color;

public Node? left;

public Node? right;

public Node? parent;

***Class RedBlackTree – червоно-чорне дерево***

\* Власне клас нашого дерева.

public void Insert(float data) - метод, який додає новий елемент до дерева. Цей метод виконує два основних кроки: спочатку він виконує стандартну вставку елемента до червоно-чорного дерева, а потім фіксує дерево, щоб забезпечити, що воно залишається червоно-чорним.

private void FixTreeAfterInsert(Node node) - метод, який фіксує дерево після вставки нового елемента. Цей метод використовує два основних операції: обертання вліво та вправо, щоб відновити властивості червоно-чорного дерева після вставки.

public float OrderStatistic(int k) - метод, який повертає k-ий елемент дерева порядкової статистики в порядку зростання. Цей метод обходить дерево вгору та/або вниз, знаходячи k-ий елемент.

private void RotateRight(Node node) - обертання вправо, використовується для фіксації дерева після вставки.

private void RotateLeft(Node node) - обертання вліво, використовується для фіксації дерева після вставки.

**Інтерфейс користувача**

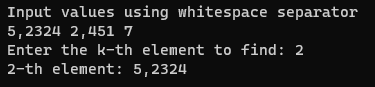
Вхідні данні задаються у консолі, при запуску програми. Результат також виводиться у консоль.

**Приклади**

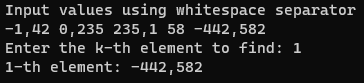
**Приклад використання**

1. **Запустіть програму**
2. **Введіть кількість дійсні числа для вставки у дерево**
3. **Натисніть “Enter”**
4. **Введіть к-тий елемент.**
5. **Натисніть “Enter”**
6. **Отримайте результат - значення к-го елемента в порядку зростання.**

**Приклад 1**



**Приклад 2**



**Висновок**

За допомогою дерева порядкової статистики можна ефективно знаходити k-ий елемент у відсортованому масиві за O(log n) часу, де n - кількість елементів у масиві. Це зроблено за рахунок використання властивостей червоно-чорного дерева, таких як балансування та швидкий доступ до елементів.

**Література**

<https://ru.wikipedia.org/wiki/Порядковая_статистика>

<https://uk.wikipedia.org/wiki/Червоно-чорне_дерево>