ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»
Университет ИТМО

Отчёт по лабораторной работе № 1
«Рекомендации для красно-чёрного сайта»

Выполнил работу:

Бабич Александр Петрович

Академическая группа: №J3112

Санкт-Петербург 2025

1. Цель и задачи

1.1. Цель работы

Разработка эффективной структуры данных на основе красно-чёрного дерева для хранения информации о фильмах по жанрам, обеспечивающей быстрый поиск рекомендаций на основе близости к таргетному рейтингу

1.2. Задачи работы

- Спроектировать структуру узла красно-чёрного дерева, включающую название фильма, среднюю оценку, сумму оценок и их количество.
- Реализовать операции вставки фильмов в дерево с автоматическим пересчётом средней оценки и поддержанием свойств красно-чёрного дерева.
- Разработать алгоритм поиска фильмов, наиболее близких к таргетному рейтингу,

2. Разбор задачи

2.1. Структура данных

Для начала разберемся с самой предоставленной структурой красно-черного дерева

- Для каждого жанра создаётся отдельное красно-чёрное дерево
- Узел дерева содержит:
  - Название фильма
  - Текущая средняя оценка
  - Сумма всех оценок
  - Количество оценок
- Дерево упорядочено по названиям фильмов (лексикографически)

2.2. Ключевые операции

Вставка/Обновление фильма (функция insert):

- При получении новой оценки:
  - Если фильм существует: обновить сумму и количество оценок, пересчитать среднее
  - Если фильма нет: создать новый узел с начальными значениями
- Поддержание свойств красно-чёрного дерева после вставки (с помощью функции fixInsert)

Поиск рекомендаций: В нашей лабе поиск рекомендованного фильма происходит с помощью функции search и функции recommend:

- Найти фильм с оценкой, наиболее близкой к целевому значению
- При одинаковой разнице: выбрать фильм с более высоким рейтингом
- Сложность: O(log n)

Приоритеты сравнения:

1. Минимальная абсолютная разница с таргетным рейтингом
2. Максимальный средний рейтинг (при равной разнице рейтингов)
3. Лексикографический порядок названий (при полном совпадении рейтингов)

3. Выделенные кейсы для автоматического тестирования

1. **Базовая вставка и проверка рейтингов** *Цель*: Проверить корректность добавления фильмов и вычисления начального среднего рейтинга. *Сценарий*:

   - Добавление 3 фильмов с разными рейтингами
   - Проверка существования добавленных фильмов в дереве
   - Верификация точности вычисления средних оценок

   Проверяемые аспекты: Корректность операций вставки, поиска и инициализации данных.

2. **Обновление средней оценки**
   Цель: Убедиться в правильности пересчёта рейтинга при добавлении новых оценок.
   Сценарий:

   - Добавление новой оценки для существующего фильма
   - Проверка обновления суммы и количества оценок
   - Верификация нового среднего значения

   Проверяемые аспекты: Механизм обновления данных и математическая точность вычислений.

3. **Проверка системы рекомендаций**
   Цель: Тестирование алгоритма поиска ближайшего значения.
   Сценарий:

   - Поиск рекомендации для целевого рейтинга 8.5
   - Проверка выбора фильма с минимальной разницей
   - Тест приоритета более высокого рейтинга при равной разнице

   Проверяемые аспекты: Корректность алгоритма рекомендаций и приоритетов выбора.

4. **Проверка свойств красно-чёрного дерева**
   Цель: Гарантировать соблюдение основных свойств структуры.
   Сценарий:

   - Рекурсивная проверка цветов узлов
   - Подсчёт чёрных узлов на всех путях
   - Проверка отсутствия двух красных узлов подряд

   Проверяемые аспекты: Соответствие структуры требованиям красно-чёрного дерева.

5. **Обработка граничных случаев**
   Цель: Проверить устойчивость системы к экстремальным значениям.
   Сценарий:

   - Добавление фильма с максимальным возможным рейтингом
   - Работа с отрицательными оценками
   - Рекомендации для пустого дерева

   Проверяемые аспекты: Обработка специальных случаев и устойчивость к переполнениям.

4. Сложности, возникшие в результате написания основного алгоритма

Разобраться как именно должна работать функция поиска и рекомендации фильма по таргетному рейтингу

5. Результаты запуска алгоритма

На тестовых случаях алгоритм отработал, на входной точке отработал (Простите, setlocate тоже не справился с табличкой :(((((( ) Вывел фильмы, их средний рейтинг и рекомендованные фильмы с таргетным рейтингом - 8.5

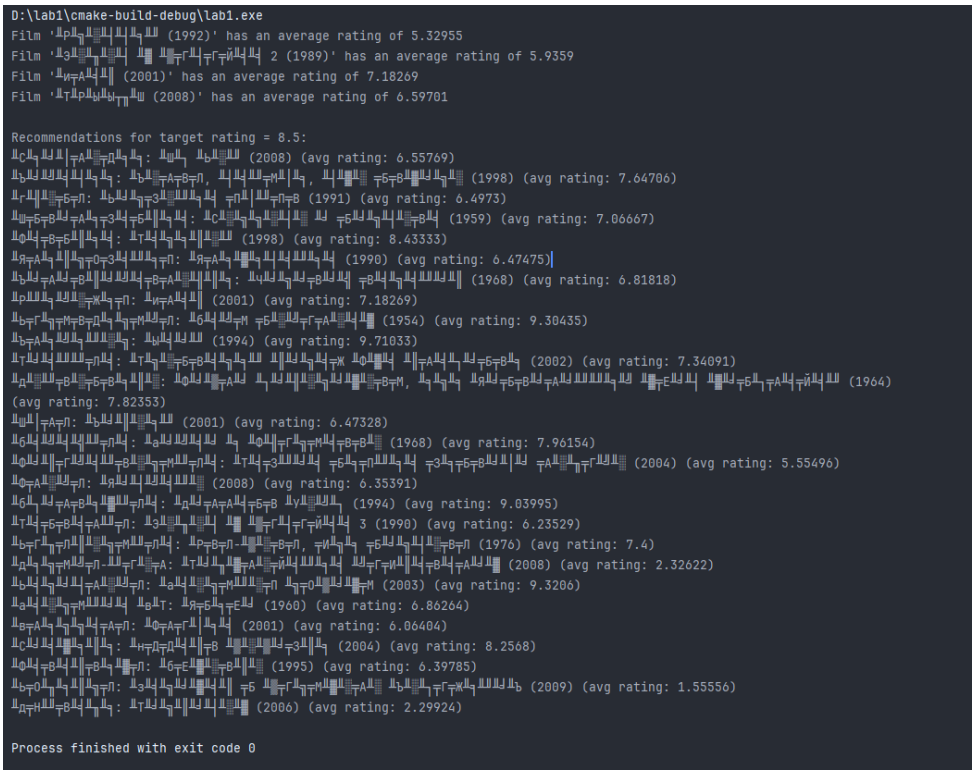

Рис. 1: Результаты main.cpp

6. Выводы

В ходе работы была успешно реализована система рекомендаций фильмов на основе красно-чёрных деревьев. Красно-черное дерево - эффективная структура данных, обеспечивающая быстрый поиск и вставку данных за (O(log n))

Полный код доступен по ссылке: https://github.com/aiAlgo25/lab1/pull/47

7. Реализация

7.1. Функция insert()

```
void insert(const std::string &film_name, double rating) {
        Node *x = root;
        Node *y = nullptr;
        Node *z = new Node(film_name, rating);
        while (x) {
            y = x;
            if (x->film_name == film_name) {

                x->updateRating(rating);
                delete z;
                return;
            }
            if (film_name < x->film_name) {

                x = x->left;
            } else {
                x = x->right;
            }
        }
        z->parent = y;
        if (!y) {

            root = z;
        } else if (y->film_name < film_name) {

            y->right = z;
        } else {
            y->left = z;
        }
        fixInsert(z);
    }
```

Листинг 1: Реализация вставки

7.2. Функция fixInsert()

```
void fixInsert(Node *node) {

        while (node->parent && node->parent->color == RED) {
            if (node->parent == node->parent->parent->left) {

                Node *y = node->parent->parent->right;
                if (y && y->color == RED) {

                    y->color = BLACK;
                    node->parent->color = BLACK;
                    node->parent->parent->color = RED;
                    node = node->parent->parent;
                } else {

                    if (node == node->parent->right) {
                        node = node->parent;
                        leftRotate(node);
                    }

                    node->parent->color = BLACK;
                    node->parent->parent->color = RED;
                    rightRotate(node->parent->parent);
                }
            } else {

                Node *y = node->parent->parent->left;
                if (y && y->color == RED) {

                    y->color = BLACK;
                    node->parent->color = BLACK;
                    node->parent->parent->color = RED;
                    node = node->parent->parent;
                } else {

                    if (node == node->parent->left) {
                        node = node->parent;
                        rightRotate(node);
                    }

                    node->parent->color = BLACK;
                    node->parent->parent->color = RED;
                    leftRotate(node->parent->parent);
                }
            }
        }

        root->color = BLACK;
    }
```

Листинг 2: Балансировка дерева

7.3. Функция search()

```
Node *search(const std::string &film_name) {
       Node *current = root;
       while (current) {
           if (film_name == current->film_name) return current;

           if (film_name < current->film_name)
               current = current->left;
           else
               current = current->right;
       }
       return nullptr;
   }
```

Листинг 3: Поиск

7.4. Функция recommend()

```
Node *recommend(double target_rating) {
       Node *bestMatch = nullptr;
       double bestDiff = INFINITY;
       inOrderRecommend(root, target_rating, bestMatch, bestDiff);
       return bestMatch;
   }
```

Листинг 4: Рекомендации