Контест

Алгоритмы и структуры данных. ПИиКТ. Осень 2025

10 окт 2025, 18:10:43
старт: 3 окт 2025, 20:49:29
начало: 27 мар 2024, 12:49:22

C. Тындекс.Экспресс

Не решена

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод
Вывод	стандартный вывод

Легенда

В мобильном клиенте интернет-магазина "Тындекс.Экспресс" необходимо реализовать ленту лучших товаров. Лента представляет собой бесконечный список карточек товаров, который можно прокручивать. На каждой карточке представлена информация о товаре.

С каждым товаром связан уникальный идентификатор, а еще его рейтинг. Уникальный идентификатор - положительное целое число (`SERIAL` из `PostgreSQL`), рейтинг - положительное целое число длины 16 бит (такое значение им приходит от какой-то модели).

В списке товары отсортированы по возрастанию рейтинга, а если рейтинг совпадает, то выше будет товар с наименьшим идентификатором. Получается, что тем меньше рейтинг, тем выше (лучше) товар в списке. Пользователь должен иметь возможность быстро находить в списке товар по идентификатору или порядковому номеру в ленте лучших товаров. То есть при заполнении некоторой формы, осуществляется переход на нужную карточку в списке.

Конечно же, товаров в системе может быть достаточно много, потому загружать информацию о всех товарах в оперативную память устройства может быть не очень удобно, поэтому пришлось реализовать некоторый аналог пагинации, только для бесконечно прокручивающихся списков.

Кроме этого, множество товаров на маркетплейсе "Тындекс.Экспресс" постоянно изменяется - там постоянно появляются новые, исчезают старые товары, а также меняется рейтинг товаров.

Инженеры из команды мобильной разработки сообщили, что им для реализации обсуждаемой функциональности будет достаточно API метода, возвращающего список товаров заданной длины, начинающийся с товара с заданным идентификатором или позицией в топе.

Естественно, хотим, чтобы все работало быстро - как методы получения, так и методы обновления данных. Ожидаемая нагрузка: редкие удаления, добавления, частые обновления, очень частые запросы на получение. Однако, нужно быть готовым к атакам с иным паттерном нагрузки.

Наши тестировщики уже разработали модульные тесты для нашего нового компонента. Осталось лишь вам реализовать интерфейс! Подразумевается, что входные данные корректны.

Особенность

В данной задаче вам предстоит реализовать только класс, определенный в заголовочном файле. Чтение и вывод данных мы берем на себя. В поле ввода решения скопировать содержимое заголовочного файла, мейн присылать не нужно.

Полная строка компиляции выглядит следующим образом.

```
CXXFLAGS=-std=c++20
CXXFLAGS+=-O3 -DNDEBUG
CXXFLAGS+=-Wall -Wextra -Wpedantic -Werror
CXXFLAGS+=-g -fsanitize=address,undefined,leak
g++ $CXXFLAGS ./Driver.cpp -o b.out
```

- Шаблон заголовочного файла
- Драйвер для запуска тестов

Формат ввода

Первая строка входных данных содержит целое число n ($0 \le n \le 1000000$) — количество операций. Следующие n строк описывают операции.

Строки имеют вид

- `i <id> <score>`. Добавление товара с уникальным идентификатором id и рейтингом $score$. Гарантируется, уникальность идентификатора, а также $1 \le id \le 10000000, 1 \le score \le 65535$.
- `u <id> <score>`. Обновление товара. Гарантируется, что товар с заданным идентификатором был добавлен в коллекцию.
- `r <id>`. Удаление товара. Гарантируется, что товар с заданным идентификатором был добавлен в коллекцию и после удаления больше никогда не появится.
- `p <position> <limit>`. Получение $limit$ товаров, начиная с позиции в ленте $position$. Гарантируется, что $0 \le position < length(feed)$, $1 \le limit \le 16$.

- `g <id> <limit>`. Получение $limit$ товаров, начиная с товара, имеющего уникальный идентификатор id. Гарантируется, что товар есть в коллекции.

Формат вывода

Сперва в отдельной строке выведите `Started`.

Далее для каждой из n операций в отдельной строке выведите

- `Added` после добавления.
- `Updated` после обновления.
- `Removed` после удаления.
- `AtPos: [(item_id ',' ' ')+]` после получения товаров по позиции в списке.
- `AtId: [(item_id ',' ' ' )+]` после получения товаров по идентификатору товара.

Пример 1

Ввод	Вывод
0	Started

Пример 2

Ввод	Вывод
3 i 1 10 p 0 1 g 1 1	Started Added AtPos: [1,] AtId: [1,]

Пример 3

Ввод	Вывод
5 i 1 10 i 2 20 p 0 2 r 1 p 0 1	Started Added Added AtPos: [1, 2,] Removed AtPos: [2,]

Пример 4

Ввод	Вывод
15 i 11 1 i 22 2 i 33 3 i 44 4 i 55 5 p 0 1 p 1 1 p 2 1 p 3 1 p 4 1 g 11 1 g 22 1 g 33 1 g 44 1 g 55 1	Started Added Added Added Added Added AtPos: [11,] AtPos: [22,] AtPos: [33,] AtPos: [44,] AtPos: [55,] AtId: [11,] AtId: [22,] AtId: [33,] AtId: [44,] AtId: [55,]

Пример 5

Ввод	Вывод
17 i 1 1 i 2 2 i 3 3 i 4 4	Started Added Added Added Added

Ввод	Вывод
i 5 5	Added
p 0 1	AtPos: [1,]
p 0 2	AtPos: [1, 2,]
p 0 3	AtPos: [1, 2, 3,]
p 0 5	AtPos: [1, 2, 3, 4, 5,]
p 0 6	AtPos: [1, 2, 3, 4, 5,]
p 0 16	AtPos: [1, 2, 3, 4, 5,]
p 0 1024	AtPos: [1, 2, 3, 4, 5,]
g 1 1	AtId: [1,]
g 1 2	AtId: [1, 2,]
g 1 3	AtId: [1, 2, 3,]
g 1 5	AtId: [1, 2, 3, 4, 5,]
g 1 6	AtId: [1, 2, 3, 4, 5,]

Пример 6

Ввод	Вывод
10	Started
i 1 10	Added
i 2 40	Added
i 3 20	Added
i 4 30	Added
p 0 4	AtPos: [1, 3, 4, 2,]
g 1 4	AtId: [1, 3, 4, 2,]
g 3 2	AtId: [3, 4,]
g 2 2	AtId: [2,]
r 3	Removed
p 1 2	AtPos: [4, 2,]

Ответ

Язык(Make) Clang 17.0.1 C++20

```
#pragma once

#include <cassert>
#include <cstddef>
#include <cstdint>
#include <cstdlib>
#include <vector>

namespace youndex::express {

struct Item {
  std::uint64_t id;
  std::uint16_t score;
};

/// NB: Is not thread safe.
class ItemFeed final {
public:
  [[nodiscard]] std::vector<std::uint64_t> GetAtPosition(std::size_t position, std::size_t limit)
      const {
    assert(position < Size());
    assert(1 <= limit && limit <= 16);

    (void)position;
    (void)limit;
```

Отправить Осталось 102 попытки Предыдущая Следующая

Посылок нет