ВШЭ АиСД 2021. Краскал, фильтр Блума

6 дек 2021, 16:22:05
старт: 28 ноя 2021, 16:00:00
финиш: 8 дек 2021, 23:59:00
до финиша: 2д. 7ч.
начало: 28 ноя 2021, 16:00:00
конец: 8 дек 2021, 23:59:00
длительность: 10д. 7ч.

C. Фильтр Блума (0.5)

Ограничение времени	1 секунда
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод
Вывод	стандартный вывод

Для решения этой задачи вам предстоит реализовать **Фильтр Блума**, который работает со строковыми объектами.

Он поддерживает:

1. Добавление строки в множество.
2. Тест принадлежности строки к множеству объектов.

Помимо этого вам требуется добавить поддержку расчета доли ложно-положительных срабатываний, то есть значения, которое показывает отношение количества тестов принадлежности, которые дали ложно-положительный результат, к общему числу поступивших тестов принадлежности.

Реализуемый Вами класс `BloomFilter` должен содержать следующие методы и конструкторы:

1. Конструктор, параметризуемый **числом хеш-функций фильтра** и **числом ячеек фильтра**.
2. Деструктор.
3. Метод `add`, который вставляет информацию о строке, с использованием хэш-функций, в множество. Принимает на вход значение строкового типа.
4. Метод `verify`, который проверяет, существует ли строка в Фильтре Блума. Принимает на вход значение типа **std::string**, возвращает значение типа **bool**. В случае, если Фильтр Блума показывает, что строка в нём находится, но при этом она не добавлялась - необходимо инкрементировать счетчик ложно-положительных значений на единицу.
5. Метод `getFPRate`, который возвращает отношение количества ложно-положительных срабатываний к сумме всех запросов к функции `verify`. Тип возвращаемого значения - *double*.
6. Константный метод `numberOfHashFunctions`, который возвращает количество хеш-функций для данного фильтра.
7. Константный метод `numberOfBits`, который возвращает количество ячеек в данном фильтре.

Поскольку фильтр содержит информацию о количестве k используемых хеш-функций, применяется следующий вариант получения k хеш-функций:

Используется объект стандартной библиотеки **std::hash<std::string>**, а при подсчете хеша для **n-ого хешера**, где **n > 0**, добавляется некоторое подобие «соли», в качестве которой выступает номер хеш-функции, приведенный к строке и добавленный в конец исходной строки. Таким образом, например, вычисление 4 хеш-функций от строки «abcd» будет выглядеть следующим образом:

1. hash0(«abcd») = std::hash<std::string>{}(«abcd»);
2. hash1(«abcd») = std::hash<std::string>{}(«abcd1»);
3. hash2(«abcd») = std::hash<std::string>{}(«abcd2»);
4. hash3(«abcd») = std::hash<std::string>{}(«abcd3»);

Примечания

Вы должны прислать код, содержащий определение вашего класса. Для удобства рекомендуется использовать публичный интерфейс из данного примера: ссылка

Ввиду того, что требуется рассчитывать долю ложно-положительных срабатываний, необходимо организовать хранение «действительных» копий строк, информация о которых была добавлена в фильтр. Вам предстоит самостоятельно выбрать способ хранения строк, с которыми вы работаете. Разрешено использование только **собственных структур**.

От выбора оптимального способа хранения строк будет зависеть оценка. Например, может быть выбрана одна из возможных реализаций префиксного дерева

Набрать здесь | Отправить файл

```cpp
#include <iostream>
#include <string>

class Vector {
private:
    std::string *array_;
    int64_t size_;
    int64_t capacity_;

public:
    Vector() {
        size_ = 0;
        capacity_ = 1;
        array_ = new std::string[1];
    }

    void pushBack(std::string data) {
        if (size_ == capacity_) {
            std::string *temp = new std::string[2 * capacity_];

            for (int i = 0; i < capacity_; ++i) {
                temp[i] = array_[i];
            }

            delete[] array_;
            capacity_ *= 2;
            array_ = temp;
        }

        array_[size_] = data;
        ++size_;
    }

    bool contains(std::string data) {
        for (int i = 0; i < size_; ++i) {
            if (array_[i] == data) {
                return true;
            }
```

Отправить

Предыдущая