Фильтр Блума

(Автор: Даниил Плющенко)

Научный руководитель: Антон Коробейников

Суть задачи

Требуется реализовать фильтр Блума для следующей задачи:

Имеется большое (10^8 - 10^9) запросов insert(x), erase(x), count(x)

(т.е. добавить объект x в структуру, удалить из неё и посчитать количество),

где x - объект произвольного типа.

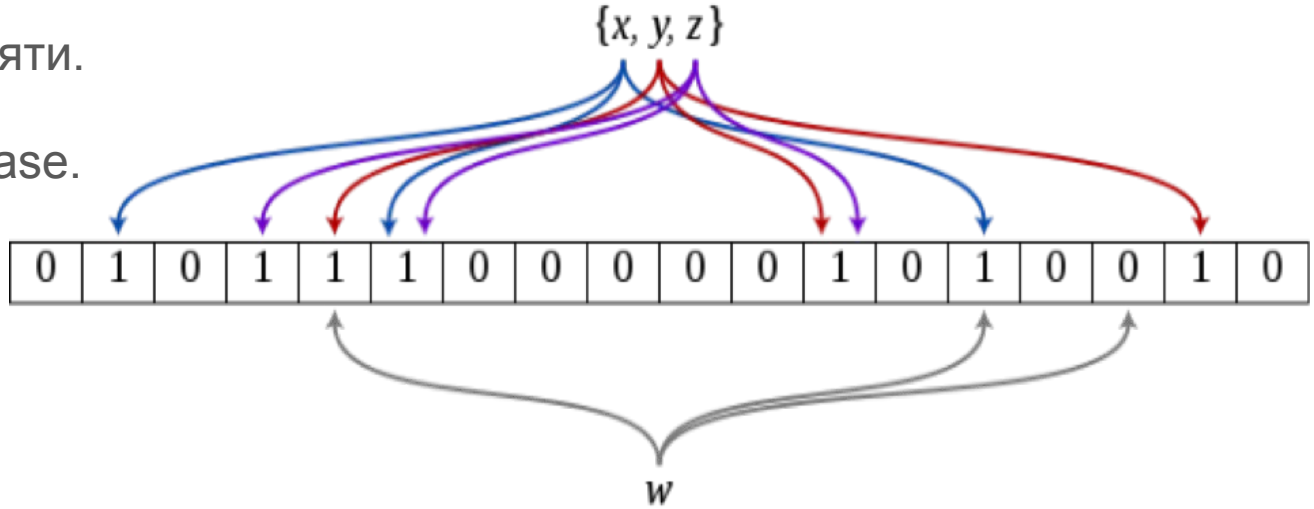
Что это?

Вероятностная структура данных. Опишем самый простой вариант:

Битовый массив и k хэш-функций. Поддерживает insert и find (Yes/No).

Плюс: экономия памяти.

Явный минус: нет erase.



Модификации

Вместо битов храним счётчики (например, 4 или 8 бит).

Тем самым получили возможность делать erase и count.

Требования к реализации

1. Многопоточность (возможность распараллелить)
2. lock-free (неблокирующая) реализация

В C++11 достигается за счёт использования

1. std::atomic (атомарные операции)
2. compare-and-swap primitive

Как хэшировать любые объекты?

Пользователь может предоставить:

1. Семейство хэш-функций
2. Класс-адаптер

   “Отображение” из объекта в пару {void *pointer, size_t length}.

   Тогда будет применяться хэширование Google CityHash

3. Ничего не предоставлять

   В этом случае (при наличии) используется std::hash

Результаты работы

Написал lock-free counting Bloom filter

Познакомился с моделью памяти в C++11 (std::atomic && compare-and-swap)

Встроил стороннюю библиотеку (Google CityHash)

Недостатки

Не было возможности потестировать на реальных данных (большой объём и много запросов) и встроить в какой-либо существующий проект

Большое спасибо!

https://github.com/Plyushchenko/BloomFilter