Звіт

Лабораторна робота №2

Шевченко Максим, МІ-3

Варіант 1

Фільтр Блума

***Умова***

Реалізувати фільтр Блума із сімома хеш-функціями.

***Теоретичні відомості***

Фільтр Блума (англ. Bloom filter) — заощадлива до пам'яті ймовірнісна структура даних, призначена для перевірки приналежності елементів до множини. Запропонована Бартоном Говардом Блумом (англ. Burton Howard Bloom) в 1970 році. Допускає помилкові спрацювання, але пропуск події неможливий, тому фільтр Блума має 100% потужність. Алгоритм дозволяє перевірити або «ймовірну приналежність до множини», або «точну не приналежність». Можна додавати нові елементи до множини, але не можна їх видаляти (хоча цю проблему можна вирішити «рахуючим» фільтром). Чим більше елементів у множині, тим вища ймовірність помилкового спрацювання.

Порожній фільтр Блума — це масив m біт, що мають значення 0. Також мають бути визначені k хеш-функцій, кожна з яких відображає або хешує елемент множини на одну з m позицій в масиві з рівномірним розподілом ймовірностей.

Аби додати елемент, слід обробити його k хеш-функціями і обчислити k позицій в масиві. Встановити значення біт в цих позиціях рівними 1.

Для перевірки присутності елемента в множині, його слід обробити кожною з k хеш-функцій та отримати k позицій в масиві. Якщо будь-який з біт на цих позиціях дорівнює 0, то елемент гарантовано не перебуває в множині. Якби елемент належав множині, то всі біти отримали б значення 1 при додаванні елемента до множини. Якщо всі біти мають значення 1, тоді або елемент належить множині, або ж біти отримали значення 1 при додаванні інших елементів, що призведе до помилкового спрацювання. Простий фільтр Блума не дозволяє уникнути помилкових спрацювань.

***Аналіз***

Нехай розмір масиву бітів дорівнює m і задано k хеш-функцій. Припустимо, що множина хеш-функцій вибирається випадково, і для будь-якого елемента x кожна хеш-функція hi призначає йому одне з місць у бітовому масиві з однаковою ймовірністю: Pr(hi(x) = p) = , p=1…m, і, крім того, значення hi(x) є незалежними в сукупності випадковими величинами (для спрощення подальшого аналізу). Тоді ймовірність того, що в певний p-й біт не буде записана одиниця під час операції вставки чергового елемента дорівнює . А ймовірність того, що j-й біт буде дорівнювати нулю після вставки n різних елементів в пустий фільтр . Хибнопозитивні спрацювання полягають в тому, що для деякого елемента y, не рівного жодному зі вставлених, усі k бітів з номерами hi(y) виявляться ненульовими, і фільтр Блума помилково відповість, що y входить у множину вставлених елементів. Імовірність такої події приблизно дорівнює . Очевидно, що ця ймовірність зменшується з ростом m (розміру бітового масиву) і збільшується з ростом n (числа вставлених елементів). Для фіксованих m і n оптимальне число k (число хеш-функцій), які мінімізують її, рівне (звідси при k=7 m=10098865). При цьому сама ймовірність хибного спрацювання рівна 0,0078.

***Результати реалізації***

Результатом є чудова програмна реалізація мовою С# не тільки звичайного фільтру Блума, але ще й «рахуючого» фільтру Блума. Крім того, ці фільтри були імплементовані в попередню роботу по хешуванню, що дає можливість перевіряти правильність дій із фільтрами порівняно з гарантовано правильно робочою хеш-таблицею.

Для звичайного фільтру Блума обраний тип даних BitArray – масив бітів. Для «рахуючого» – byte[] – масив цілих невід’ємних чисел від 0 до 255. Більше максимальне значення лічильника не потрібне, адже середнє його значення рівне . На практиці необхідності в більшому розмірі також не виникало, хоча для підстрахування повідомлення про помилку в разі перевищення лічильника буде виводитися в консоль.

Заповнені фільтри виводяться у файли “outBloom.txt” і “outBloomCount.txt”. Константи поліноміальних хеш-функцій фільтрів Блума рі = { 31, 47, 97, 113, 211, 71, 179 }. Як можна помітити, ці хеш-функції забезпечують досить рівномірне хешування, адже значення лічильників «рахуючого» фільтру не перевищує кількість потраплянь однакового слова в множину (файли “hist.txt” і “output.txt” містять однакові слова).

Файл “error.txt” містить помилки виконання запитів. Із можливих помилок (додавання, пошук, видалення) у ньому присутні тільки помилки пошуку у звичайному фільтрі Блума (хеш-таблиця й «рахуючий» фільтр не містять елемент, який є у звичайному фільтрі). Зрозуміло, що це пояснюється дозволом на видалення елементів. Якщо залишити тільки операції додавання й пошуку, то помилок не буде. Це зайвий раз свідчить про правильну реалізацію й виконання поставленого умовою роботи завдання.