КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра математичної
інформатики

Дисципліна «Проблеми кодування та захисту інформації»

ЗВІТ

з виконання лабораторної роботи №2

на тему:

“Write a bloom filter”

Виконав аспірант

Волохович Ігор

ЗМІСТ

1. Мета роботи

Метою даної лабораторної роботи є вивчення принципів роботи ймовірнісної структури даних Bloom Filter, реалізація її в середовищі C# з використанням бібліотеки Spectre.Console для інтерфейсу користувача, а також розробка модульних тестів за допомогою фреймворку xUnit для перевірки коректності реалізації.

2. Теоретичні відомості

2.1. Поняття Bloom Filter

Bloom Filter — це ймовірнісна структура даних, розроблена для ефективної перевірки належності елемента множині. Основними перевагами Bloom Filter є:

- Економія пам'яті порівняно з традиційними структурами даних
- Константний час операцій додавання та перевірки належності
- Відсутність можливості видаляти елементи з множини

Bloom Filter може давати хибнопозитивні результати (помилково вказувати, що елемент є у множині, коли насправді його там немає), але ніколи не дає хибнонегативних результатів.

2.2. Основні принципи роботи

Bloom Filter складається з:

1. Бітового масиву фіксованого розміру (m бітів)
2. Набору хеш-функцій (k функцій)

Операції:

- **Додавання елемента**: елемент хешується k разів, і встановлюються відповідні біти в масиві.
- **Перевірка належності**: елемент хешується k разів, і перевіряється, чи встановлені всі відповідні біти в масиві.

2.3. Оптимальні параметри

Для заданої кількості елементів n та бажаної ймовірності хибнопозитивного результату p:

- Оптимальний розмір бітового масиву: $m = -n * \ln(p) / (\ln(2)^2)$
- Оптимальна кількість хеш-функцій: $k = (m/n) * \ln(2)$

3. Опис реалізації

3.1. Структура програми

Програма складається з таких компонентів:

1. Клас BloomFilter, що реалізує ймовірнісну структуру даних
2. Консольний інтерфейс користувача, реалізований з використанням Spectre.Console
3. Набір модульних тестів, створених за допомогою xUnit

3.2. Реалізація класу BloomFilter

Клас BloomFilter містить:

- Бітовий масив для зберігання даних
- Методи для додавання елементів та перевірки їх належності

- Методи для обчислення хеш-функцій
- Властивості для відстеження статистики фільтра

```
public class BloomFilter
{
    // Бітовий масив для зберігання даних
    private readonly bool[] _bits;
    // Кількість хеш-функцій
    private readonly int _hashFunctionCount;
    // Кількість доданих елементів (орієнтовна)
    private int _elementsAdded;
    // Властивості та методи реалізації...
}
```

3.3. Алгоритм хешування

Для ефективного хешування використовується алгоритм MurmurHash3 разом з технікою Кірша-Міценмахера для генерації декількох хеш-функцій з двох базових хешів:

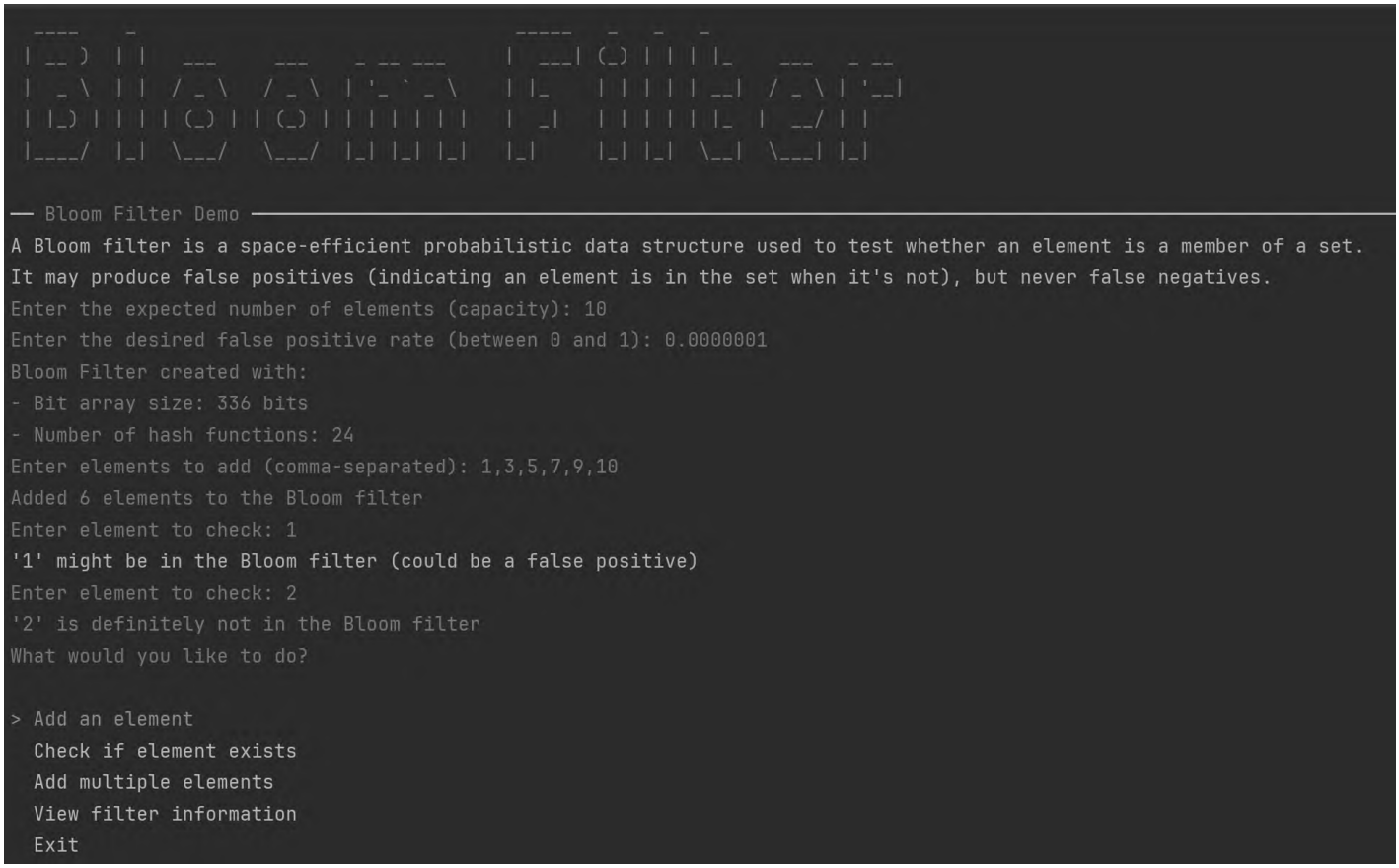
```
private IEnumerable<int> GetHashIndexes(string element)
{
    // Використовуємо дві хеш-функції для створення k різних хеш-функцій
    byte[] data = Encoding.UTF8.GetBytes(element);

    // Обчислюємо два незалежних хеш-значення
    var hash1 = MurmurHash3(data, 0);
    var hash2 = MurmurHash3(data, hash1);

    // Генеруємо k хеш-функцій за формулою: h_i(x) = (hash1 + i * hash2) % m
    for (int i = 0; i < _hashFunctionCount; i++)
    {
        uint combinedHash = (uint)((hash1 + ((long)i * hash2)) % (uint)int.MaxValue);
        yield return (int)(combinedHash % (uint)Size);
    }
}
```

3.4. Користувацький інтерфейс

Інтерфейс користувача реалізований за допомогою бібліотеки Spectre.Console, що

дозволяє створити зручний та інтерактивний досвід використання консольного додатку:

- Інтерактивне меню для вибору дій
- Кольорове форматування виводу
- Зручні форми вводу для користувача



4. Тестування

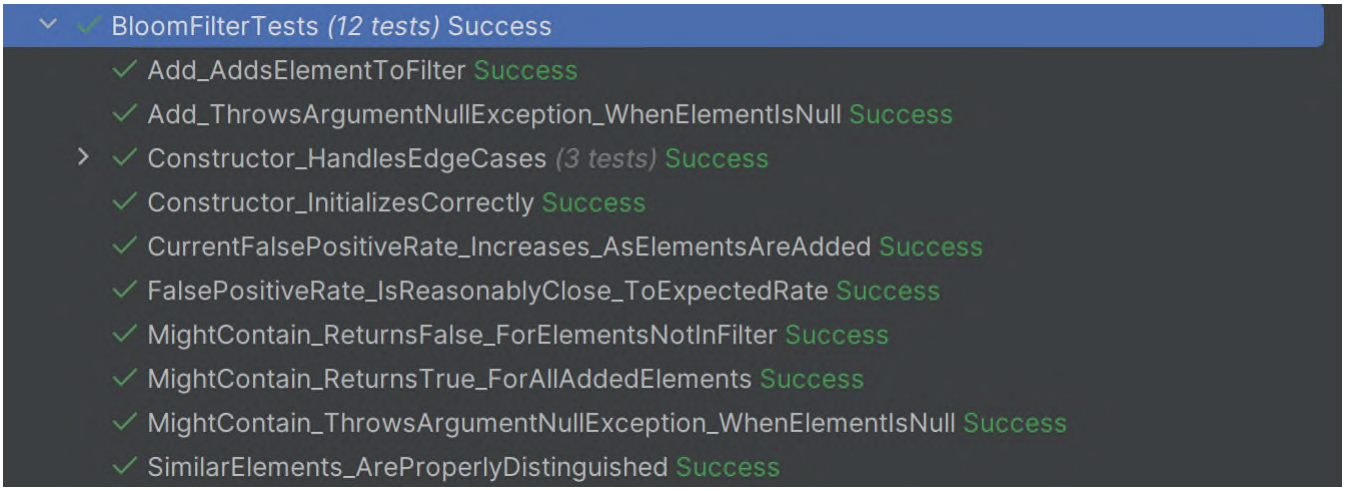
4.1. Методика тестування

Для тестування реалізації Bloom Filter використовується фреймворк xUnit. Тести включають:

1. Тестування правильної ініціалізації конструктора
2. Перевірка крайових випадків
3. Тестування функціональності додавання та перевірки елементів
4. Перевірка коректності обробки помилок
5. Вимірювання частоти хибнопозитивних результатів

4.2. Результати тестування

Всі тести були успішно пройдені, що підтверджує коректність реалізації Bloom Filter. Частота хибнопозитивних результатів відповідає теоретично розрахованим значенням в межах статистичної похибки.



5. Висновки

У ході виконання лабораторної роботи було:

1. Вивчено теоретичні основи структури даних Bloom Filter
2. Реалізовано ефективну версію Bloom Filter на мові C#
3. Створено інтерактивний користувацький інтерфейс за допомогою Spectre.Console
4. Розроблено комплексний набір модульних тестів
5. Підтверджено коректність реалізації через тестування

Bloom Filter є потужною структурою даних, яка дозволяє значно економити пам'ять в задачах, де допустимі хибнопозитивні результати, але не допустимі хибнонегативні. Практичне застосування цієї структури даних може бути знайдено в різних областях, включаючи кешування, фільтрацію спаму, перевірку паролів, тощо.