Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет прикладной информатики

**Лабораторная работа №5**

Выполнили:

Мануковская Д. М.

Сакулин И. М.

Сафронов И. С.

Проверил

Мусаев А. А.

Санкт-Петербург,

2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc182322648)

[Хэширование сложением. 4](#_Toc182322649)

[Хэширование умножением и делением 5](#_Toc182322650)

[Задание 2 6](#_Toc182322651)

[ЗАКЛЮЧЕНИЕ 8](#_Toc182322652)

[СПИСОК ЛИТЕРАТУРЫ 9](#_Toc182322653)

ВВЕДЕНИЕ

В данной лабораторной работе необходимо выполнить 2 задания. В первом задача изучить базовые алгоритмы хэширования (сложения, умножения и деления). Второе заключается в написании программы для хэширования вводимого текста по алгоритму CRC-8.

Цель работы: изучение различных алгоритмов хэширования.

Хэширование сложением.

Создадим функцию для реализации хэширования сложением. Листинг представлен на Рисунке 1.

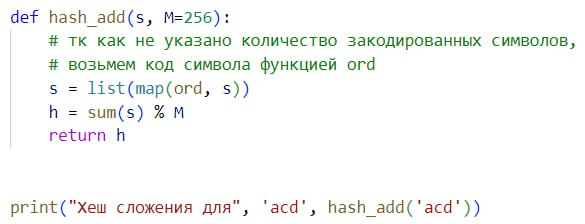


Рисунок 1 – Хэширование сложением

Из недостатков можно отметить большое количество коллизий (разные последовательности имеют одинаковый хэш, перестановка символов не влияет на хэш).

Причины коллизий: при хэшировании не учитывается индекс кодированного символа, сохраняется только остаток от деления, поэтому если хэши отличаются на целое М, то они будут иметь одинаковый хэш.

Хэширование умножением и делением

На Рисунке 2 представлены функции hash\_division, hash\_multiply для хэширования умножением и делением.

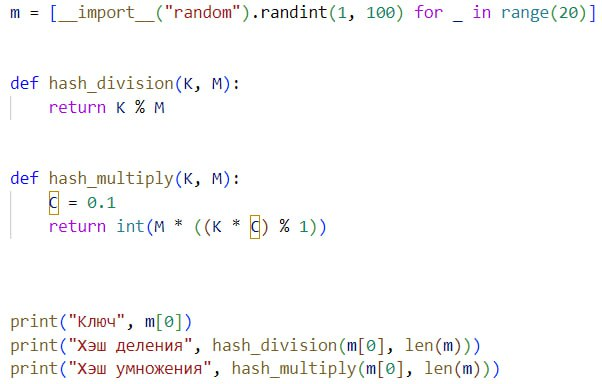


Рисунок 2 – Хэширование умножением и делением

Алгоритм умножения основан на умножении ключа на некоторую константу и последующем извлечении некоторых битов для получения значения хэша.

Недостатки:

-Неэффективен для некоторых ключей, так как процесс умножения может привести к большим значением хэша, что потребует больше памяти для хранения.

-Возможны коллизии, особенно если ключи содержат схожие цифры или символы.

Алгоритм деления основан на делении ключа на некоторое число и использовании остатка от деления в качестве хэш-значения.

Недостатки:

-Неэффективен для некоторых значений ключей, так как деление может занимать больше времени, особенно для больших чисел.

-Возможны коллизии, особенно если ключи делятся на одно и то же число.

Причины коллизий: ключи, с похожими цифрами.

Способы улучшения: подбор константы, комбинирование 3-х простых алгоритмов.

Задание 2

Для хэширования алгоритмом CRC-8 создали функцию, листинг которой предоставлен на рисунке 3.

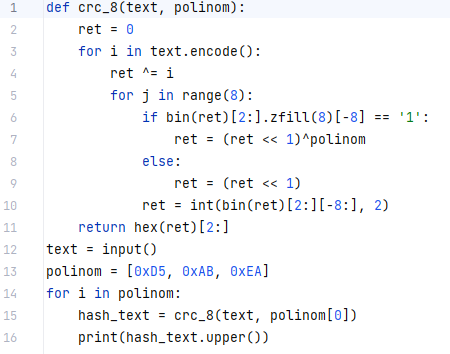


Рисунок 3 – CRC-8

CRC-8 — это алгоритм циклического избыточного кода (CRC) для вычисления 8-битной контрольной суммы. CRC-8 используется для обнаружения ошибок в данных, передаваемых по каналу связи или хранимых в файле.

Как работает CRC-8:

1. Начальное значение: Алгоритм начинается с начального значения (обычно 0).

2. Побитовая обработка: Каждое байт данных обрабатывается побитово.

3. Исключающее ИЛИ: Каждый бит данных XOR'ится с соответствующим битом полинома CRC-8.

4. Сдвиг: Результат сдвигается на один бит влево.

5. Проверка старшего бита: если старший бит результата равен 1, выполняется XOR'ирование результата с полиномом CRC-8.

6. Повторение: Шаги 3-5 повторяются для всех байтов данных.

7. Финальное значение: Финальное значение результата — это контрольная сумма CRC-8.

Полином CRC-8:

Существует несколько различных полиномов CRC-8, но наиболее распространенным является полином:

• x^8 + x^2 + x + 1 (в двоичной форме: 100000111)

Применение CRC-8:

CRC-8 используется во многих различных приложениях, включая:

• Серийные коммуникации: CRC-8 используется для обнаружения ошибок в данных, передаваемых по серийным портам, например, UART.

• Хранение данных: CRC-8 может использоваться для обнаружения ошибок в файлах, хранящихся на жестком диске или в памяти.

• Сетевые протоколы: CRC-8 используется в некоторых сетевых протоколах, например, в CAN bus.

Преимущества CRC-8:

• Простая реализация: CRC-8 относительно легко реализовать в программах и на аппаратном уровне.

• Небольшая контрольная сумма: 8-битный CRC-8 не требует много памяти.

• Хорошая детекция ошибок: CRC-8 может обнаружить многие распространенные ошибки, такие как ошибки одиночного бита и ошибки нескольких битов.

ЗАКЛЮЧЕНИЕ

Цель работы достигнута. Изучили и реализовали базовые алгоритмы хэширования: сложением, умножением, делением. Выявили их недостатки и причины коллизий. Написали функцию для хэширования вводимого текста по алгоритму CRC-8.

СПИСОК ЛИТЕРАТУРЫ