Лабораторная работа №

Колобаев Дмитрий
Группа Б01-903

8 ноября 2019 г.

Содержание

1 Аннотация

В данной работе проводится сравнительный анализ хеш-функций на прдемет равномерности распределения хешей.

2 Теория

Хеш-функция, или функция свёртки — функция, осуществляющая преобразование массива входных данных произвольной длины в (выходную) битовую строку установленной длины, выполняемое определённым алгоритмом. Преобразование, производимое хеш-функцией, называется хешированием. Исходные данные называются входным массивом, «ключом» или «сообщением». Результат преобразования (выходные данные) называется «хешем», «хеш-кодом», «хеш-суммой», «сводкой сообщения».

Требования к хорошей хеш функции: иньективность соответствия хеша ключу (отсутствие коллизий), скорость работы, забавное название (желательно связанное с котиками)

3 Методика и оборудование

Рис. 1: Лабораторная установка

Рис. 2: Блок питания

Рис. 3: Устройство для исправления ошибок

В работе используется:

1. ЭВМ 1 для построения хеш таблиц, на основе данных хеш-функций. После чего строится график зависимости размера списка от индекса в хеш-таблице. По полученному графику определяются пики коллизий функции. По равномерности распределения делается вывод о качестве хеширования.

2. Чайник для заваривания кофе. 2

3. Бубен для дебага. 3

4 Результаты и обработка

Пояснения к графикам

В качестве выборки использовася английский словарь на 350000 слов.

4.1 ASCII код первой буквы

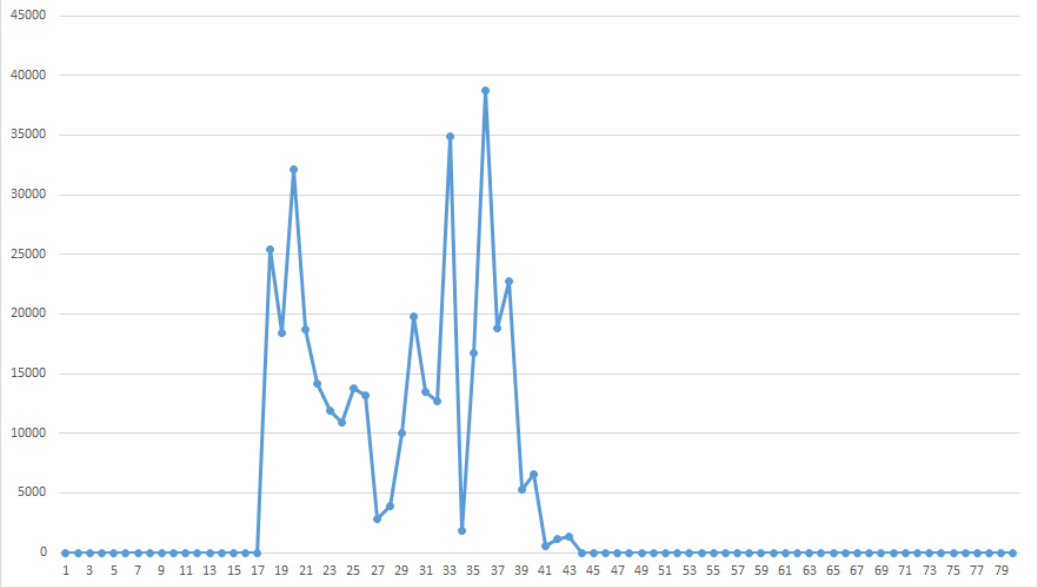

Рис. 4: Хеш равен ASCII коду первой буквы

График 4. Можно заметить, что мощность множества значений хеша равна количеству букв в английском алфавите. А пики значений соответсвуют кодам наиболее часто встречающихся буквю. Но в целом хеш-функция плохая.

4.2 Длина строки

График 5. Видно, что длины слов удовлетворяют нормальному распределению, а это плохо для хеш-функции.

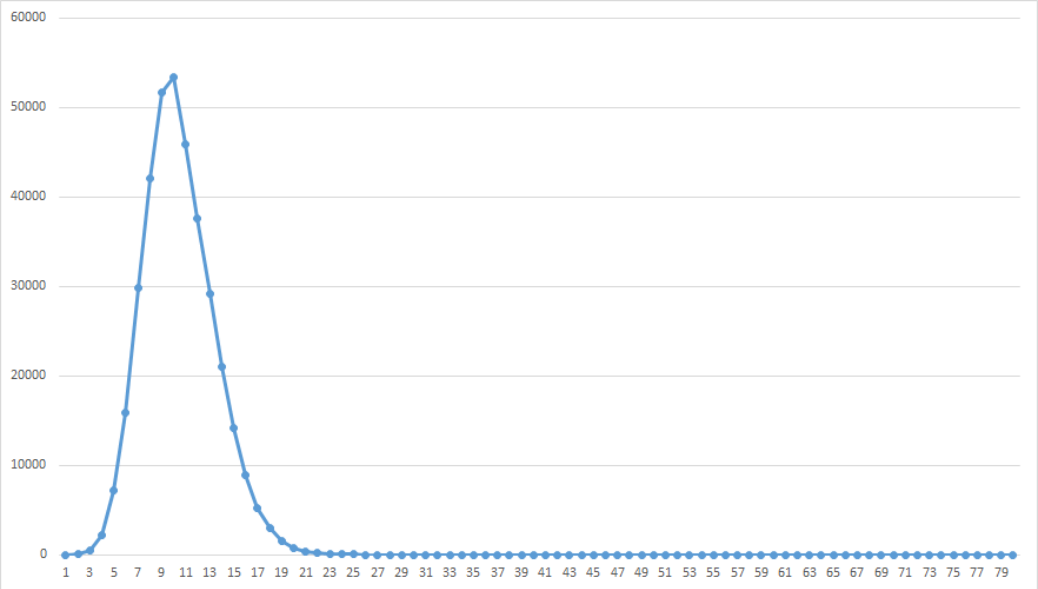

Рис. 5: Хеш равен длине строки

4.3 Сумма ASCII кодов символов строки

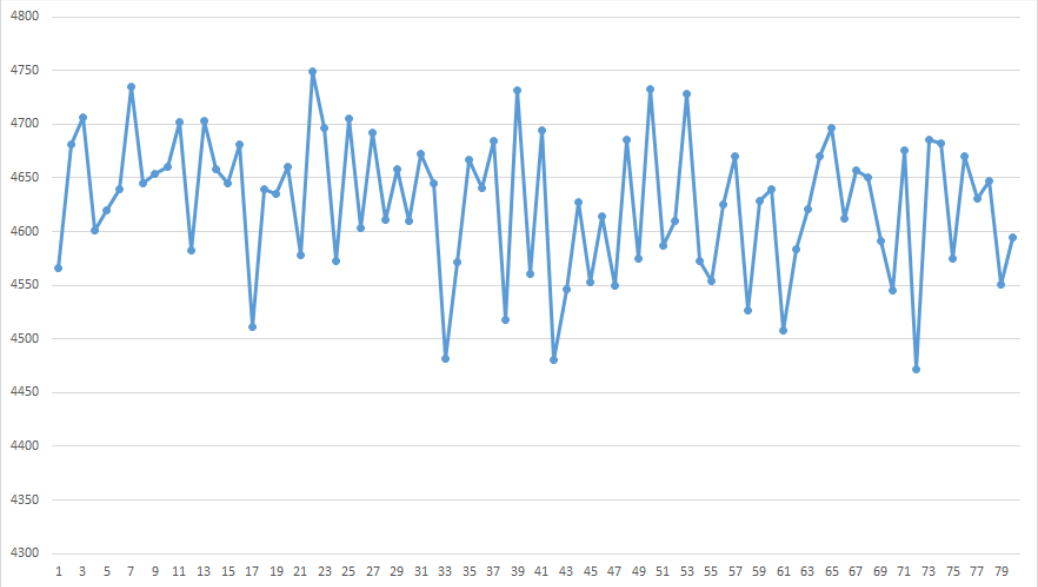

Рис. 6: Хеш равен сумме ASCII кодов

График 6. Этот алгоритм оказался хорошей хеш-функцией, но это скорее всего связано с особенностями входных данных.

4.4 ROT hash - хеш через циклический сдвиг и xor

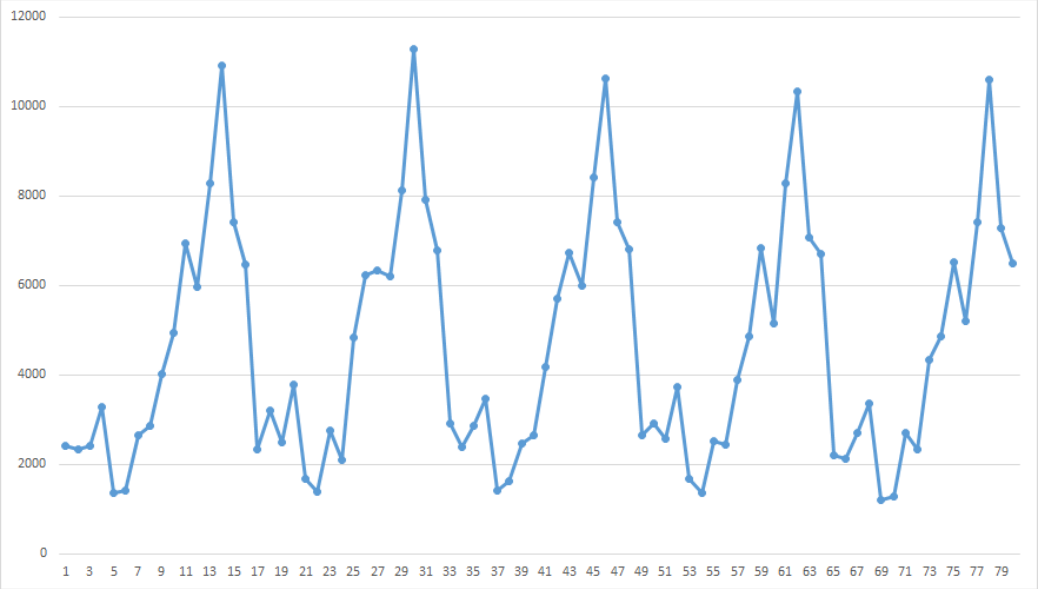

Рис. 7: Рекурсивный rot hash

График 7. Функция имеет много коллизий, повторяющихсся периодически.

4.5 murmur hash

График 8. Ну шо тут сказать, отличная функция.

4.6 GNU hash

График 9. Ну шо тут сказать, отличная функция.

4.7 ELF hash

График 10. Функция имеет периодическое распределение. Видимо алгоритм был предназначен для хэширования конкретного типа входных данных, на котором давал бы равномерное распредление.

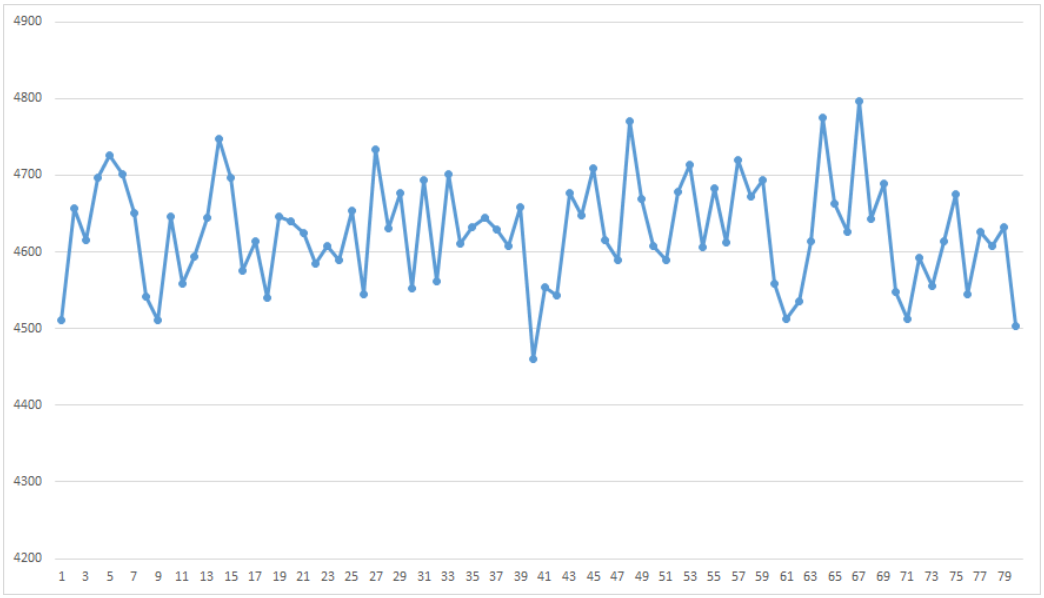

Рис. 8: murmur hash

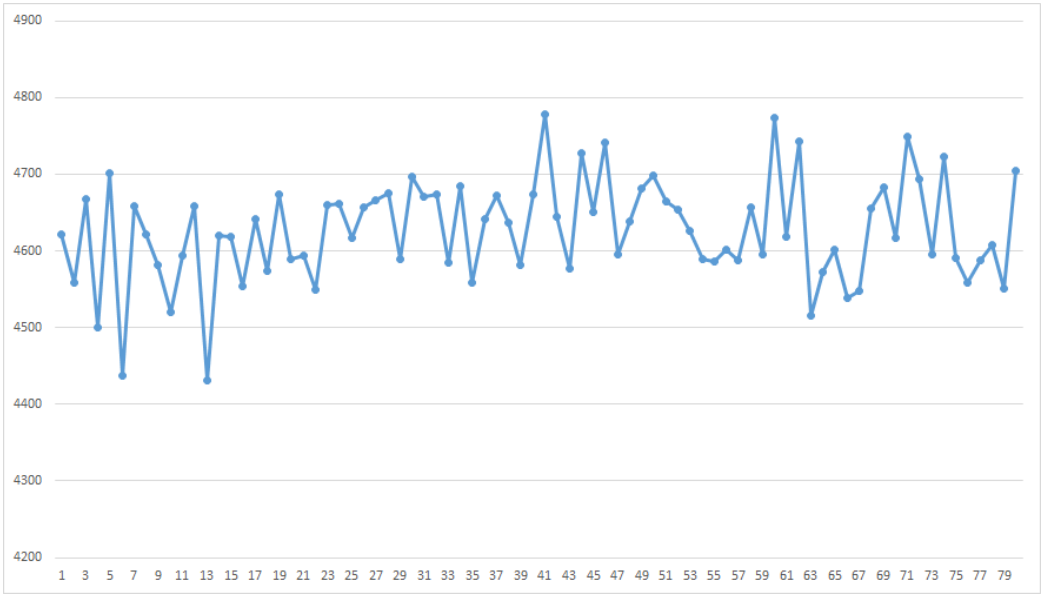

Рис. 9: GNU hash

4.8 JS hash

График 11. Ну шо тут сказать, отличная функция.

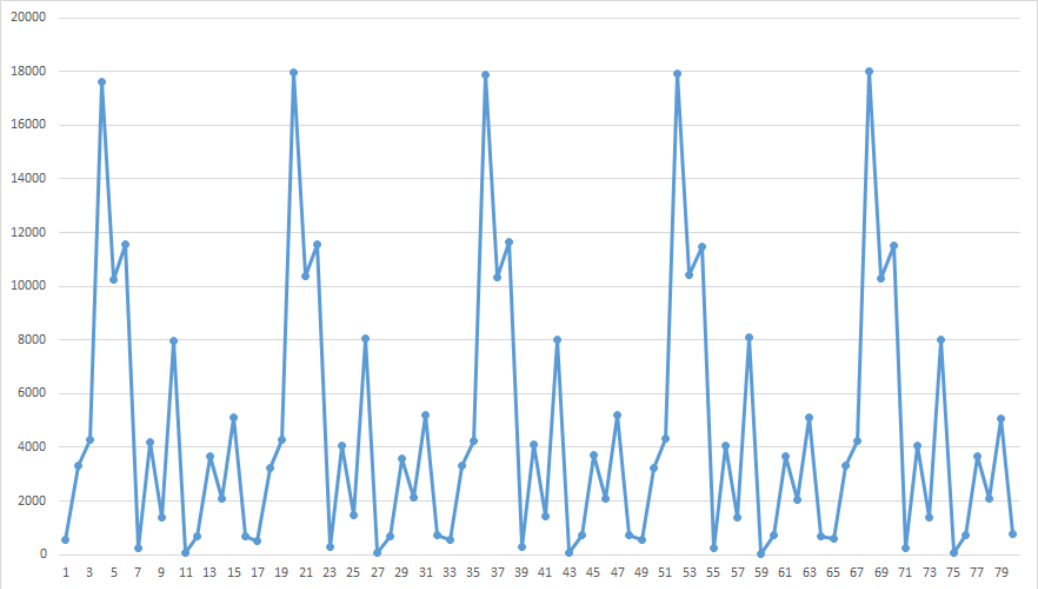

Рис. 10: ELF hash

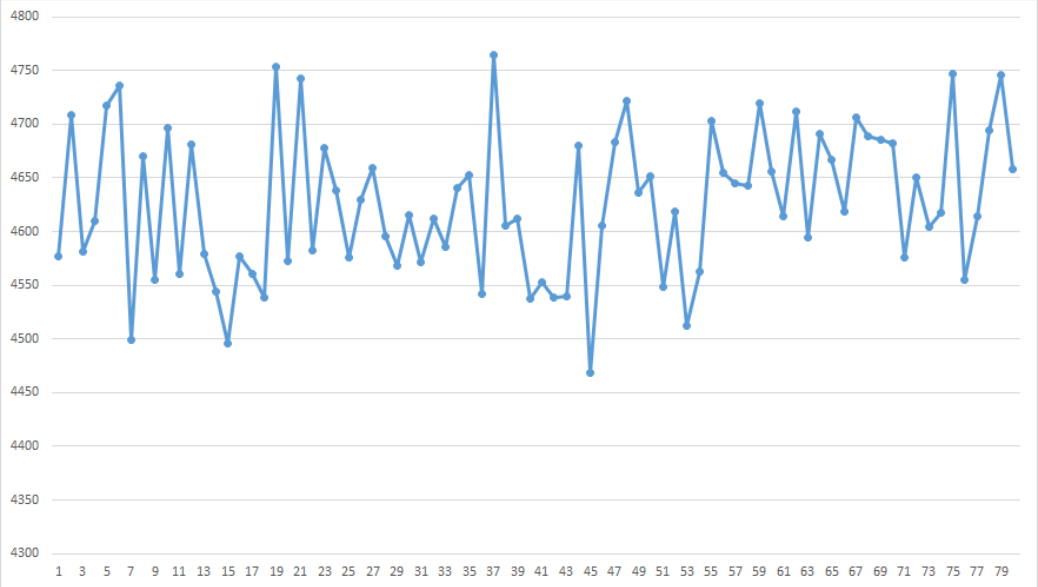

Рис. 11: JS hash

4.9 ROT13 hash

График 12. Ну шо тут сказать отличная функция.

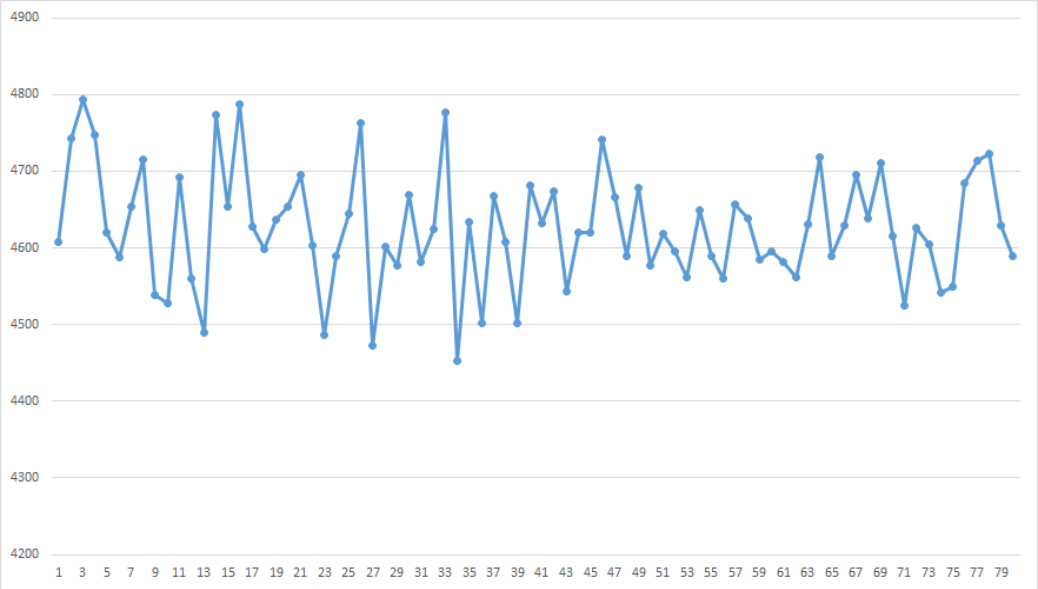

Рис. 12: ROT13 hash

5 Обсуждение результатов

5.1 Сравнение алгоритмов

Для сравнения алгоритмов поместим их на общий график 13. Видно, что по сравнению с другими распредление JS, GNU, murmur, ROT13 и суммы ASCII кодов представляет собой почти прямую.

5.2 Погрешности измерений

Погрешность измерений достаточно мала, экспериментальные занчения совпадают с теоретическими в пределах погрешности, основной вклад вносят случайные погрешности измерений, погрешности, погрешности, отклонение, МНК, погрешности...

6 Заключение

В данной работе проводился сравнительный анализ хеш-функций на предмет равномерности распределения хешей. МурМур лучше всех, по объективному критерию.
?Приведение типов с typedef() типами не работает?

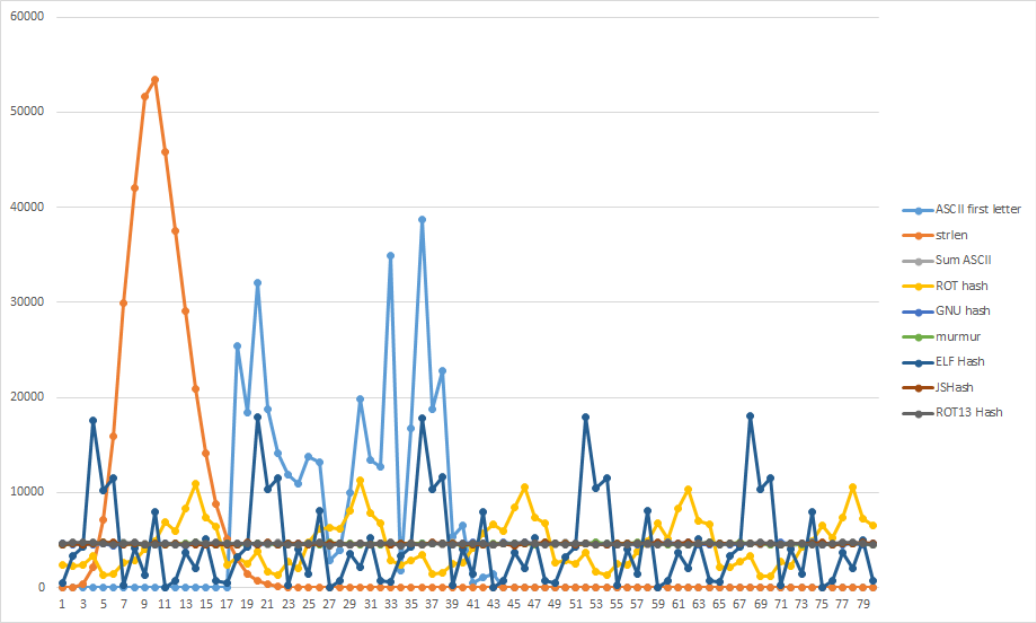

Рис. 13: Сравнение хеш-функций

?Для хэш таблицы нужен более абстрактный лист - не достаточно просто typedef, нужен обобщенный тип <T> с интерфейсом Comparable для ключей, иначе неясно как сравнивать экземпляры между собой(для строк == не работает?
На большой выборке отличить вечный цикл от долго работающей программы сложно, поэтому для начала стоит потестить на маленькой выборке.