РЕШЕНИЕ ЗАДАЧ, СВЯЗАННЫХ С ОЦЕНКОЙ НАДЁЖНОСТИ ХЕШИРОВНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Саскевич А. В, Шидловский О. В., Марчук М. С.

Стройникова Е. Д. – ассистент кафедры информатики

Уровень сложности новых алгоритмов непрерывно растёт. На сегодняшний день одной из наиболее актуальных проблем в области программирования является сжатие и быстрая обработка данных посредством хеширования.

Хеширование представляет собой преобразование по детерминированному алгоритму входного массива данных произвольной длины в выходную битовую строку фиксированной длины. Такие преобразования также называются хеш-функциями или функциями свёртки, а их результаты называют хешем, хеш-кодом или сводкой сообщения. Использование хеширования для сжатия информации позволяет значительно ускорить процесс выполнения поставленных задач. Однако, в связи с тем, что алгоритм хеширования ориентирован, в первую очередь, на максимальное сжатие данных, он может приводить к возникновению коллизий, то есть совпадению функции хеширования для различных начальных значений. Сейчас проблема разработки алгоритма сжатия данных, не допускающего коллизии, является наиболее важной в данной области.

На сегодняшний день существует множество способов хеширования данных. Целесообразно их разделение на 2 вида:

- быстровычисляемые (применяются для работы с большими объёмами данных);
- минимизирующие количество коллизий (применяются для работы с малыми объёмами данных и большим количеством различных строк).

Различные хеш-функции могут быть охарактеризованы по следующим параметрам:

1) скорость вычисления функции;
2) вероятность возникновения коллизии;
3) наличие мощного лавинного эффекта.

Методы быстрого вычисления хеш-функции

В процессе работы авторы рассмотрели некоторые стандартные алгоритмы хеширования. Также при проведении исследования была выявлена функция, наиболее подходящая для быстрого хеширования:

$$f(s) = (s[0]+s[1]*p+s[2]*p\text{^}2....s[n]*p\text{^}n) \bmod M, \quad (1)$$

где p – простое число, примерно равное мощности алфавита, M – модуль функции.

В результате сложность вычисления хеша оказывается равной $O(n)$. Однако при малых значениях длины строки вероятность появления коллизии возрастает.

Ниже приведены результаты экспериментальной апробации функции для 50000 разных строк:

Таблица 1. Количество коллизий

Длина строки	Количество коллизий	Процент коллизий
8	49744	0.9949
16	48478	0.9696
24	42634	0.8527
32	23380	0.4676
40	6696	0.1339
48	1456	0.291
64	303	0.061
104	0	0
128	0	0

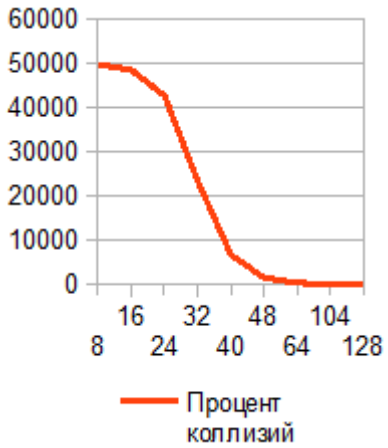

Рисунок 1. График числа коллизий

При замене константы p более высоким значением количество коллизий резко уменьшается. Можно заметить, что такая функция равномерно распределяет значение хеш-функциии по её области определения как для больших объёмов данных, так и для относительно небольшого количества различных строк. Однако при значительном увеличении количества анализируемых строк без повышения модуля функции количество коллизий резко возрастает.

Следует также отметить, что использование принципа двойного хеширования, по двум независимым функциям $h1(s)$ и $h2(s)$, при грамотном выборе хеш-функций в среднем повышает эффективность

алгоритма, в связи со снижением вероятности одновременного совпадения значений двух независимы функций при различных входных данных.

Методы, минимизирующие количество коллизий

Можно повысить эффективность использования функции (1) путём сохранения значений. Таким образом, в случае совпадения значения хеш-функции будут сохранены обе строки с подходящими значениями. Данный подход позволяет избавиться от коллизий, однако значительно увеличивает время обработки строки.

Рассмотрим методы, позволяющие наиболее равномерно распределить хеш. На данный момент существует множество способов хеширования. Наиболее популярные из них: MD5, Adler32, FNV, MurMur2, PJW. Эти способы осуществляют применение эвристических методов для равномерного распределения. К примеру, MD5 медленно распределяет хеш для строки малой длины. В то же время для больших строк он распределяет хеш менее точно, так как вероятность коллизий в этом случае значительно ниже.

Таблица 2. Количество коллизий при применении MD5:

Длина строки	Количество коллизий	Процент коллизий
8	49744	0.9949
16	14901	0.298
24	81	0.16
32	0	0

Данный подход позволяет вычислять хеш для любых наборов данных с наименьшей вероятностью коллизии.

Можно сделать вывод о том, что быстрое вычисление хеш-функции приводит к увеличению вероятности возникновения коллизий. Таким образом, в случае работы со строками малой длины целесообразно затратить больше времени для распределения хеш-функции.

Список использованных источников:

1. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер — М.: «Триумф», 2002. — 816 с.: ил.
2. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт — М.: «Мир», 1989. — 360 с.: ил.