**Алгоритмы хеширования данных**

**Хэширование** или **хеширование** — преобразование массива входных данных произвольной длины в (выходную) битовую строку фиксированной длины, выполняемое определённым алгоритмом. Функция, реализующая алгоритм и выполняющая преобразование, называется «***хеш-функцией»***.

Появился термин «хеширование» в середине прошлого века среди людей занимающихся обработках массивов данных. Хеш-функция позволяла **привести любой массив данных к числу заданной длины**. Хэш-функции – это функции, предназначенные для «сжатия» произвольного сообщения или набора данных, записанных, как правило, в двоичном алфавите, в некоторую битовую комбинацию фиксированной длины, называемую сверткой. Хэш-функции имеют разнообразные применения при создании ассоциативных массивов, при тестировании логических устройств, при проверке целостности записей в базах данных и при построении алгоритмов быстрого поиска(таких как хеш-таблицы).

Идеальная хеш-функция должна легко вычисляться и быть похожей на случайную функцию: для любых аргументов результаты в некотором смысле должны быть равновероятными.

Приведу в пример несколько простых алгоритмов хеширования.

1.Построение хеш-функции методом деления с остатком (division method)(слайд 4). Этот метод состоит в том, что ключу k ставится в соответствие остаток от деления k на m, где m— число возможных хеш-значений. Но такой алгоритм хеширования создает множество коллизий. Об этой проблеме, я думаю нам расскажут в теме «Хеш-таблицы».

2.Так же можно реализовать построение хеш-функции методом умножения (multiplication method)(слайд 5). Он состоит в следующем: пусть количество хеш-значений равно m. Зафиксируем константу А в интервале 0 < А < 1, и положим в

***h(k) = [m(kA mod 1)]***

где (kA mod 1) – дробная часть kA.

Метод умножения работает при любом выборе константы А, но некоторые значения А могут быть лучше других. Оптимальный выбор зависит от того, какого рода данные подвергаются хешированию.

В своей книге Кнут детально рассматривает эту особенность и приходит к выводу, что значение A = 0,61 является довольно удачным.

2.И самый простой алгоритм хеширования для строковых ключей(слайд 6). В котором хеш находится с помощью цикла из **n** итераций, где **n** – это количество символов в ключе. Этот цикл реализован следующим образом: хеш строки, равняется остатку от деления на **m**, суммы целочисленного представления i-ого символа сроки и произведения хеша (i-1)-ого символа на константу С. Эта константа должна быть больше любого int(str[i]).

Популярные хэш-алгоритмы сжатия

1. **CRC32(или Циклический избыточный код)** (слайд 7) — используется именно для создания контрольных сумм. Данная функция не является криптографической. Есть много вариаций этого алгоритма (число после CRC означает длину получаемого хеша в битах), в зависимости от нужной длины получаемого хеша. Функция очень простая и нересурсоемкая. В связи с этим используется для проверки целостности пакетов в различных протоколах передачи данных.
2. **MD5** (слайд 8)— старая, но до сих пор очень популярная версия уже криптографического алгоритма, разработанный профессором **Ро́нальдом Линном Риве́стом** из Массачусетского технологического института, которая создает хеш длиной в 128 бит. Хотя стойкость этой версии на сегодняшний день и не очень высока, она все равно часто используется как еще один вариант контрольной суммы, например, при скачивании файлов из сети.
3. **SHA-1** (слайд 9) — криптографическая функция формирующая хеш-суммы длиной в 160 байт. Сейчас идет активная миграция в сторону SHA-2, которая обладает более высокой устойчивостью, но SHA-1 по-прежнему активно используется хотя бы в качестве контрольных сумм. Но она так же по-прежнему используется и для хранения хешей паролей в базе данных сайта.