Задача 5. Лавинный эффект

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Для изучения качества хеш-функции принято рассматривать такое явление, как «лавинный эффект» (avalanche effect). Грубо говоря, он показывает, на какие биты хеша влияет каждый бит ключа. Считается, что если хеш-функция хорошая, то при переключении любого бита в ключе на противоположный в выходном хеше должна поменяться примерно половина битов. Например, хорошее лавинное поведение показывает хеш-функция Дженкинса. В этой задаче предлагается вычислить лавинный эффект для заданной хеш-функции.

Хеш-функция вычисляется следующим кодом на языке C:

```
uint64_t A, B, M, R, S;
uint32_t hashFunc(uint32_t x) {
    return (((A * x + B) % M) % R) / S;
}
```

Как видно, эта функция принимает 32-битные беззнаковые ключи, и выдаёт 32-битные беззнаковые хеши. **Осторожно:** знаковость и битность всех переменных в этом коде имеет значение и должна быть именно такой, как написано!

Таблица лавинного эффекта для этой функции имеет размер 32×32, то есть в ней 32 строки и 32 столбца. В i-ой строке в j-ом столбце записано число, обозначаемое p_{ij}. Число p_{ij} равно вероятности того, что изменение i-ого бита в случайном ключе приведёт к изменению j-ого бита в его хеше. При этом считается, что при выборе случайного ключа все 32-битные беззнаковые ключи равновероятны.

Формат входных данных

В единственной строке записано пять неотрицательных целых чисел: A, B, M, R, S — параметры хеш-функции. Обратите внимание, что все эти числа 64-битные и не превышают 10^{18}. Чтобы избежать деления на ноль, параметры M, R, S гарантированно ненулевые.

Формат выходных данных

Выведите таблицу лавинного эффекта как 32 строки по 32 значения в каждой. Для каждой ячейки таблицы выведите число p_{ij} в процентах: для этого нужно умножить p_{ij} на 100 и округлить до целого.

Все выведенные числа должны быть целыми. Поскольку за короткое время не получится посчитать p_{ij} с идеальной точностью, ваше решение может немного ошибиться: ваш ответ будет засчитан в том и только в том случае, если каждое число отличается от правильного не более чем на 1 (то есть ошибка в каждой вероятности должна быть не более процента).

Примеры

input.txt	output.txt
2654435769 0 4294967296 4294967296 4096	нормалёк =)
2654435769 0 1048576 1000000000 1	фу, неее =(
938572893 2139875776 1000000007 1048576 1	отлично!
15642 322777666 10000000 1000000000 1	как-то не очень...

Пояснение к примеру

В примерах входные данные не всегда умещаются в одну строку. Кроме того, выходные данные не указаны, так как они сильно большие и не входят в текст. Набор выходных данных для всех четырёх тестов можно скачать по ссылке.

По поводу хеш-функций из примеров можно сказать следующее:

1. В первом примере задана правильная хеш-функция Кнута, выдающая 20-битный хеш:
$$f(x) = \left\lfloor \frac{(A \cdot x) \mod 2^{32}}{2^{12}} \right\rfloor$$
Как видно, лавинный эффект очень хороший: каждый бит хеша зависит хотя бы от 10-12 битов ключа.
Единственная проблема — старшие биты ключа не влияют на младшие. Если кто-то решит хешировать числа, которые все делятся на 65536, то младшие 5 битов хеша не будут варьироваться вообще. С другой стороны, таких 32-битных чисел всего 65536, и может быть в хеш-таблице они разместятся без проблем.

2. Второй пример показывает неправильную реализацию хеш-функции Кнута, когда в хеш выбираются младшие биты вместо старших:
$$f(x) = (A \cdot x) \mod 2^{20}$$
Как видно, у этой функции очень плохой лавинный эффект: на каждый бит хеша влияет намного меньше битов ключа, чем у правильной хеш-функции Кнута. В целом, эта хеш-функция показывает, почему нужно быть осторожным с модулем, являющимся степенью двойки.

3. Третий пример показывает правильный вариант линейной хеш-функции с 20 выходными битами:
$$f(x) = [(A \cdot x + B) \mod P] \mod 2^{20}$$
Здесь число P равно $10^9 + 7$ и является простым, а коэффициенты A и B выбраны произвольно. Заметьте, что перед взятием остатка от деления на 2^{20} предварительно берётся остаток от деления на простое P — это важно.
У этой функции очень хорошее лавинное поведение: вся таблица заполнена. Хотя есть ячейки, в которых зависимость слабая (стоит почти ноль или почти 100), однако в общем каждый бит хеша зависит от подавляющего большинства битов ключа.

4. В последнем примере применяется такая же хеш-функция, но убрано взятие остатка от деления на простое число, отчего лавинный эффект резко ухудшается.