Задача 11. Большая сортирующая машина

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

Вам предлагается испытать себя в оптимизации сортировки массива большого размера.

В каждом тесте имеется массив из N элементов, у каждого элемента есть ключ и значение. Ключ является 64-битным беззнаковым целым числом, а значение — 32-битным.

Изначально каждому элементу массива присваивается значение, равное его номеру (считая с нуля). Далее нужно выполнить R раундов. На каждом раунде нужно:

1. Сгенерировать и записать N случайных чисел в ключи элементов массива.
2. Отсортировать элементы массива в порядке возрастания ключа.

В результате каждого раунда значения в массиве переставляются в некотором порядке, который зависит от генератора псевдослучайных чисел.

В данной задаче нужно использовать генератор псевдослучайных чисел xorshift+. Исходный код этого генератора:

```
uint64_t xorshift128plus(uint64_t state[2]) {
    uint64_t x = state[0];
    uint64_t const y = state[1];
    state[0] = y;
    x ^= x << 23; // a
    state[1] = x ^ y ^ (x >> 17) ^ (y >> 26); // b, c
    return state[1] + y;
}
```

Начальное состояние генератора (два числа в `state`) задаётся в каждом тесте. В начале каждого раунда ключи генерируются для элементов в их текущем порядке, причём старшие два бита отбрасываются:

```
for (int i = 0; i < n; i++)
    elements[i].key = xorshift128plus(state) & 0x3fffffffffffffffLL;
```

Обратите внимание, что процесс полностью детерминирован, значит порядок элементов после выполнения всех раундов определяется однозначно. В конце теста нужно вывести 64-битный хеш от финального порядка элементов:

```
uint64_t hash = 5381;
for (int i = 0; i < n; i++)
    hash = hash * 31 + elements[i].value;
```

Формат входных данных

В первой строке записано целое число Q — сколько тестов записано в файле ($1 \leqslant Q \leqslant 10$). Далее описано Q тестов.

Описание теста начинается со строки с двумя целыми числами: N — размер массива и R — сколько раундов сортировки нужно выполнить ($1 \leqslant N \leqslant 10^6$, $0 \leqslant R$). Во второй строке теста записано два шестнадцатеричных числа, по шестнадцать цифр в каждом — начальное содержимое массива `state` генератора `xorshift+`.

Гарантируется, что во всех раундах всех тестов все ключи будут различными. Суммарное количестов раундов R по всем тестам в файле не превышает 40. Кроме того, сумма всех N в файле не превышает 10^6.

Формат выходных данных

Для каждого из Q тестов нужно вывести одно целое число в отдельной строке — финальный хеш. Хеш следует выводить в виде шестнадцатеричного числа из ровно 16 цифр (если нужно, добавьте ведущие нули).

Пример

input.txt
5 15 0 b1c6114bf18c80b8 059cace124e9297b 15 1 b1c6114bf18c80b8 059cace124e9297b 15 2 b1c6114bf18c80b8 059cace124e9297b 15 3 b1c6114bf18c80b8 059cace124e9297b 10 7 4c373cdb0102026b a8b5ef27370796de
output.txt
25d1acb755e9cd42 f9237b1061947488 f1af6c757ccad3ba 7238a85181dfa6bc 3d356e76dc3e5d86

Комментарий

Финальный порядок элементов (`elements[i].value`) в пяти тестах примера такой:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
8 14 10 6 0 7 4 12 2 11 1 3 13 9 5
2 12 6 1 0 9 4 8 5 3 10 7 14 13 11
11 4 1 0 6 14 9 3 12 8 10 2 7 5 13
5 1 0 4 2 7 9 3 6 8
```

В данной задаче бессовестно жёсткое ограничение по времени.

Во входном файле под номером $5 + R$ записан один тест с R рандами и максимальным N (для каждого $R = 1 \dots 40$).

Внимание: не пытайтесь применять многопоточность! В nsuts замеряется суммарное процессорное время по всем потокам, поэтому многопоточность не поможет.