

Задача 1. Удаление дубликатов 2

Источник:	базовая*
Имя входного файла:	input.bin
Имя выходного файла:	output.bin
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Дан массив A , в котором содержится n целых чисел. Нужно удалить из него дубликаты (т.е. повторы чисел), так чтобы в массиве каждое имеющееся в нём значение встречалось ровно один раз.

Если значение встречается в массиве несколько раз, то нужно удалить все его вхождения, кроме самого первого. Порядок оставшихся элементов должен быть сохранён.

Внимание: задачу требуется решать с помощью **хеш-таблицы**.

Формат входных данных

В первых четырёх байтах записано число n — сколько чисел в массиве ($1 \leq N \leq 10^6$). Далее записано n чисел, по четыре байта каждое. Все числа целые, по модулю не превышают 10^9 .

Формат выходных данных

В первых четырёх байтах нужно вывести целое число k — сколько различных чисел в массиве A . Далее нужно вывести k этих чисел, по четыре байта каждое. Числа должны быть выведены в том порядке, в котором их первые вхождения идут в исходном массиве.

Пример

input.bin															
0A	00	00	00	01	00	00	00	01	00	00	00	FE	FF	FF	FF
04	00	00	00	03	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	FE	FF	FF	FF				
output.bin															
05	00	00	00	01	00	00	00	FE	FF	FF	FF	04	00	00	00
03	00	00	00	00	00	00	00								

Задача 2. Цикличность случайных чисел

Источник:	базовая*
Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	разумное

Как известно, у генератора псевдослучайных чисел есть внутреннее состояние, которое может принимать конечное количество различных значений. Из этого следует, что если достаточно долго генерировать псевдослучайные числа, то в какой-то момент они начнут повторяться. В данной задаче нужно найти, с какого момента начнётся повторение у заданного квадратичного конгруэнтного генератора.

Квадратичный конгруэнтный генератор определяется четырьмя целочисленными параметрами a , b , c и $M \geq 2$. Его состояние представляется целым числом `state`, которое всегда находится в диапазоне от 0 до $M - 1$ включительно. Функция перехода для этого генератора выглядит так:

```
uint64_t func(uint64_t s) {  
    return (s*s*a + s*b + c) % M;  
}
```

Изначально, состояние генератора `state` равно единице. Далее каждый раз, когда пользователь запрашивает новое случайное число:

1. Пользователю выдаётся текущее значение `state` в качестве случайного числа.
2. К состоянию применяется функция перехода: `state = func(state)`;

Обозначим последовательность случайных чисел, которую выдаёт генератор, через $x_0, x_1, x_2, x_3, \dots$. Нетрудно заметить, что всегда $x_0 = 1$. Будем говорить, что в этой последовательности циклически повторяется отрезок от l до r , если $x_{l+i} = x_{r+i}$ для любого $i \geq 0$.

Даны параметры генератора, нужно найти отрезок от l до r , который циклически повторяется. Поскольку вариантов выбора отрезка много, требуется найти такой, у которого число r минимально возможное.

Формат входных данных

В первой строке записано целое число M — модуль генератора ($2 \leq M \leq 10^{12}$). Во второй строке записано три целых числа a , b , c — параметры генератора ($0 \leq a, b, c \leq 10^9$).

Обратите внимание, что при указанных ограничениях в функции перехода `func` может происходить беззнаковое 64-битное переполнение. Это нормально, так и должно быть.

Формат выходных данных

Выведите два целых числа l и r через пробел — отрезок, которые циклически повторяются. Среди всех возможных вариантов нужно выбрать тот, в котором число r минимальное.

Гарантируется, что в ответе $r \leq 2 \cdot 10^6$.

Внимание: для обнаружения совпадений нужно использовать **хеш-таблицу**.

Примеры

input.txt	output.txt
11 1 4 5	1 4
999999999999 1 0 7	977966 1389969

Задача 3. Найти коллизию

Источник:	основная
Имя входного файла:	<code>stdin</code>
Имя выходного файла:	<code>stdout</code>
Ограничение по времени:	5 секунд
Ограничение по памяти:	разумное

В данной задаче вам предлагается найти коллизию для неизвестной вам хеш-функции, то есть указать два различных ключа, на которых значение хеш-функции совпадает.

Известно, что хеш-функция принимает 32-битное беззнаковое целое число на вход (ключ) и выдаёт 32-битное беззнаковое целое число на выход (хеш). Кроме того, известно, что хеш-функция очень хорошего качества.

Вы можете вычислять хеш-функцию на любых ключах, на каких хотите. Однако всего разрешается сделать не более $2 \cdot 10^5$ вычислений.

Протокол взаимодействия

В данной задаче ваша программа будет работать не с файлами, а совместно с программой-интерактором. Ваша программа и интерактор будут запускаться одновременно, и соединяться пайпами (теми самыми пайпами, о которых упоминалось на лекции при рассмотрении очереди и кольцевого буфера). Всё, что ваша программа выводит в `stdout`, читает интерактор из своего `stdin`, а всё, что пишет интерактор на `stdout`, читает ваша программа из своего `stdin`.

Ваша программа должна печатать команды, которые интерактор будет выполнять. Есть два типа команд:

- Команда `eval`, после которой через пробел должно быть записано 32-битное беззнаковое целое число X . Эта команда предписывает интерактору вычислить значение хеш-функции от числа X . Интерактор вычислит его и записывает искомый хеш: ваша программа должна прочитать его из `stdin` как беззнаковое 32-битное целое.
- Команда `answer`, после которой через пробел должно быть записано два 32-битных беззнаковых целых числа A и B . Этой командой ваша программа должна сообщить интерактору коллизию: числа A и B должны быть различными, но их хеш должен совпадать. После этого ваша программа должна сразу же завершить исполнение, ничего никому больше не записывая и ничего ниоткуда не читая.

Если вы сделаете больше вычислений хеш-функции, чем разрешено, или выдадите неверный ответ командой `answer`, то ваше решение получит `Wrong Answer`.

Поскольку задача интерактивная, требуется:

1. Не открывать никаких файлов, **не** использовать `freopen` и `fopen`.
2. Писать команды с помощью `printf` и читать ответы с помощью `scanf`.
3. После каждой команды выводить символ перевода строки и сразу после этого выполнить: `fflush(stdout);`

Если вы забудете сделать команду `fflush`, то записанные вами в `stdout` данные останутся в буфере, и никогда не попадут в пайп, а значит интерактор никогда их не получит и всё зависнет (вердикт `Timeout`).

Учтите, что в этой задаче все числа беззнаковые, так что писать и читать их надо с форматом `"%u"`.

Пример

stdin	stdout
2478003845	eval 1
894250524	eval 2
622810134	eval 3
894250524	eval 4
	answer 2 4

Пояснение к примеру

Обратите внимание, что в примере сначала программа печатает команды в `stdout`, а уже потом на них приходят ответы от интерактора. В данном случае у ключей 2 и 4 получается одинаковый хеш, равный 894250524.

Исполняемый файл интерактора вы можете скачать по ссылке (только для Windows). Чтобы запустить его просто поиграться, нужно использовать командную строку:

```
interactor.exe input.txt output.txt
```

Чтобы запустить его вместе с вашим решением `sol.exe`, можно использовать командную строку (предварительно надо поставить Python 3):

```
python run_interactive.py sol.exe
```

Выведенные вашим решением команды будут записаны в `output.txt`.

Задача 4. Сравнение подстрок

Источник:	повышенной сложности
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
Ограничение по времени:	3 секунды
Ограничение по памяти:	разумное

Дана строка S длиной в N символов, и размер блока B . В этой строке имеется ровно $(N - B + 1)$ подстрок длины B (подстрока — это часть строки, которая является непрерывным отрезком). Нужно раскрасить все эти подстроки в цвета, так чтобы одинаковые подстроки имели одинаковый цвет, а разные подстроки — разный цвет.

Формат входных данных

В первой строке входного файла записано два целых числа: N — длина строки и B — длина рассматриваемых подстрок ($1 \leq B \leq N \leq 10^6$).

Во второй строке дана сама строка S . Её длина равна N , и она состоит только из маленьких букв латинского алфавита.

Формат выходных данных

В выходной файл необходимо вывести строку $(N - B + 1)$ целых чисел через пробел: цвета всех подстрок длины B . Все цвета должны быть в диапазоне от 0 до $K - 1$ включительно, где K — количество различных цветов. Цвета нужно выводить в том порядке, в котором подстроки располагаются в строке S .

Пример

input.txt	
15 3	
abacabadabacaba	
output.txt	
0 3 1 5 0 4 2 6 0 3 1 5 0	