Задача 1. Количество единиц

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

Составить программу подсчета длины самой длинной последовательности рядом стоящих единиц в двоичном представлении заданного целого числа.

Формат входных данных

Во входном файле записано одно целое число N, по модулю не превосходящее 10^6.

Формат выходных данных

В выходной файл нужно выдать одно целое число – длину самой длинной последовательности рядом стоящих единиц в двоичном представлении числа N.

Примеры

input.txt	output.txt
9	1
115	3

Задача 2. Обращение числа

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

Обернуть заданное число N на битовом уровне. Выдать значение полученного числа.

Формат входных данных

В первой строке входного файла задано одно целое число N (входит в формат `int`).

Формат выходных данных

В первую строку выходного файла нужно вывести представление заданного числа на битовом уровне.

Во второй строке должно быть записано целое число – значение числа, полученного обращением заданного числа на битовом уровне.

В третьей строке – представление обращенного числа на битовом уровне.

Массивы и строки не использовать!

Пример

input.txt	output.txt
4	00000000000000000000000000000100 536870912 00100000000000000000000000000000

Задача 3. Численный эксперимент

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

Вернувшись от Модеста Матвеевича, я застал в машинном зале Витьку Корнеева, растерянно стоявшего у консоли. Оказалось, что он, воспользовавшись моим отсутствием, решил всё же провести какой-то свой численный эксперимент на системе «Янус-3». Естественно, что неоттестированная система дала сбой. По идее, на выходе программа должна была вывести несколько чисел, являющихся степенями двойки. Однако последняя команда отработала неправильно, и каждое из чисел было умножено на случайное четырёхзначное простое число. Похоже, что «сборка мусора» в системе функционировала некорректно.

Я заметил, что несанкционированный запуск неоттестированных программ приводит именно к таким последствиям, обозвал Корнеева бета-тестером, поблагодарил за найденный баг и занялся было отладкой. Но тут Витька сказал, что, если он правильно понимает, он сэкономил мне значительное время, и что мне всё же за это не мешало бы привести результаты в нужный вид. В чём-то он был прав, так что мне надо было быстро восстановить правильные результаты ...

Формат входных данных

В первой строке входного файла записано целое число N ($1 \leqslant N \leqslant 200000$). В следующих N строках записаны числа, по одному в строке, каждое из которых имеет вид $r_i \cdot 2^{k_i}$, где r_i – простое число, а k_i – натуральное число ($1000 < r_i < 10000, 0 < k_i \leqslant 16, 1 \leqslant i \leqslant N$).

Формат выходных данных

В выходной файл необходимо вывести N чисел вида 2^{k_i} , по одному в строку, в порядке, соответствующем входному.

Пример

input.txt	output.txt
5 8824 64576 2554 16024 10138624	8 64 2 8 1024

Задача 4. Пересечение множеств

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды*
Ограничение по памяти:	разумное

Дан набор из N различных битовых масок. Две битовые маски называются конфликтующими, если у них есть общий единичный бит (хотя бы один). Иными словами, если k-ый бит равен единице в обеих масках для некоторого k, тогда маски конфликтуют. Требуется определить, сколькими способами можно выбрать пару неконфликтующих масок из заданного набора.

Формат входных данных

В первой строке входного файла задано целое число N — количество масок ($1 \leqslant N \leqslant 3 \cdot 10^4$).

В каждой из оставшихся N строк записано по одной битовой маске. Битовая маска — это беззнаковое 64-битное число, заданное в шестнадцатеричном виде ровно шестнадцатью цифрами.

Для чтения битовых масок рекомендуется использовать формат: `%llx`

Формат выходных данных

В выходной файл необходимо вывести одно целое число (в десятичном виде) — количество пар неконфликтующих масок.

Пример

input.txt	output.txt
7 0000000000000000 FFFFFFFFFFFFFFFF 0000100000000000 0000000004000000 0000A00000000000 0000666666600000 0000000012345000	13

Комментарий

Чтобы решение работало быстро, нужно определять наличие конфликта очень быстро, не перебирая отдельные биты в масках.

Задача 5. Битовый массив 1

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

По заданной последовательности целых чисел сформировать битовый массив и применить к нему ряд команд, устанавливающих нужные значения в указанные места.

Формат входных данных

В первой строке входного файла находится целые числа N и M – количество целых чисел в массиве и количество команд ($1 \leqslant N, M \leqslant 100$).

В следующей строке через пробел записано N целых чисел, по модулю не превосходящих 1000 – элементы массива. Далее в M строках записано по одной команде. Каждая команда состоит из двух целых чисел. Первое число может принимать одно из двух значений: 0 или 1, а второе – номер элемента в битовом массиве, куда нужно установить указанное в первом числе значение. Считаем, что нумерация битового массива начинается с 0.

Формат выходных данных

В выходной файл нужно вывести преобразованный массив – N целых чисел через пробел.

Пример

input.txt	output.txt
2 3 5 10 0 31 1 27 1 63	20 11

Задача 6. Битовый массив 2

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда*
Ограничение по памяти:	**5 мегабайт**

Память современных компьютеров можно представить в виде очень длинного массива битов, в котором каждый бит может принимать значения 0 или 1. Однако возможность обращаться к конкретному биту по его номеру отсутствует: минимальная адресуемая единица памяти — это байт, обычно состоящий из 8 битов. Чтобы работать с отдельными битами памяти, необходимо обращаться к байтам (или словам), которые их содержат, и применять различные битовые операции.

В данной задаче нужно реализовать структуру данных «битовый массив» / “bitset”. Это массив, элементы которого принимают лишь значения 0 и 1. Эти элементы можно легко сохранить в массиве байтов, храня одно значение в каждом байте. Однако если сохранять по 8 элементов в байте, то снижается потребление памяти и появляется возможность ускорить некоторые операции — именно это и требуется сделать в данной задаче.

Нужно реализовать набор функцию со следующей сигнатурой:

```
//какой-нибудь целочисленный тип (желательно беззнаковый)
typedef ??? bitword;

//инициализирует массив битов длины num, устанавливая все его биты в ноль
void bitsetZero(bitword *arr, int num);
//возвращает значение idx-ого бита (0 или 1)
int bitsetGet(const bitword *arr, int idx);
//устанавливает значение idx-ого бита в newval (которое равно 0 или 1)
void bitsetSet(bitword *arr, int idx, int newval);
//возвращает 1, если среди битов с номерами k
//для left <= k < right есть единичный, и 0 иначе
int bitsetAny(const bitword *arr, int left, int right);
```

Массив слов `arr` управляется вызывающим, и может указывать, к примеру, на глобальный массив достаточного размера. Вы можете самостоятельно выбрать базовый тип для слова `bitword`. Однобайтовый тип подойдёт, однако для лучшей производительности рекомендуется использовать слова большего размера. Функция `bitsetAny` определяет, есть ли в заданном окне/отрезке битов хотя бы один бит, равный единице. При реализации этой функции **необходимо** быстро обрабатывать окна большого размера: для этого нужно целиком перебирать слова, полностью попадающие внутрь окна, не перебирая отдельные биты.

При помощи реализованной структуры нужно решить тестовую задачу.

Формат входных данных

В первой строке записано целое число N — количество операций, которые нужно обработать ($1 \leqslant N \leqslant 10^5$). В каждой из следующих N строк описывается одна операция над битовым массивом.

Описание операции начинается с целого числа t, обозначающего тип операции. Если $t = 0$, то это операция `bitsetZero`, и вторым целых числом в строке указан размер массива `num`. Если $t = 1$, то это операция `bitsetGet`, и в строке также записано целое число `idx` — номер бита. Значение этого бита нужно выдать в выходной файл. Если $t = 2$, то это операция

`bitsetSet`, и в строке также содержатся целые числа `idx` и `newval`. Здесь `idx` — номер бита, который нужно изменить, а `newval` — новое значение, которое надо записать (0 или 1). Если $t = 3$, то это операция `bitsetAny`, и указано ещё два параметра `left` и `right` — отрезок, на котором нужно искать единичные биты. Если единичный бит на отрезке есть, то надо вывести `some`, а если нет — то `none`.

Размер массива `num` больше нуля и не превышает 10^7. Гарантируется, что после операции `bitsetZero` с параметром `num` все последующие операции обращаются только к битам с номерами от 0 до `num`-1 включительно — как минимум до следующего вызова `bitsetZero`.

Сумма значений `num` по всем операциям `bitsetZero` не превышает $2 \cdot 10^9$. Сумма длин отрезков (`right-left`) по всем операциям `bitsetAny` не превышает $2 \cdot 10^9$.

Формат выходных данных

Для каждой операции `bitsetGet` или `bitsetAny` нужно вывести ответ в отдельной строке.

Пример

input.txt	output.txt
14 0 100 2 30 1 2 31 1 2 32 1 1 31 1 7 2 31 0 1 31 3 30 33 3 31 32 3 0 100 3 45 67 0 48 1 30	1 0 0 some none some none 0

Пояснение к примеру

Сначала инициализируется массив из 100 битов. Потом устанавливаются в 1 биты с номерами 30, 31, 32. Потом делается запрос на значения битов 31 и 7 — они равны 1 и 0 соответственно. Далее 31-ый бит зануляется и запрашивается его уже нулевое значение. Потом делаются запросы о том, есть ли единичные биты на отрезках $[30, 33)$, $[31, 32)$ и $[0, 100)$. Обратите внимание, что правый конец отрезка в данной задаче исключается. Наконец, выполняется пересоздание массива, теперь размера 48 битов, в результате 30-ый бит становится нулевым.

Задача 7. Битовый массив 3

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	**5 мегабайт**

Это расширенная версия задачи «Битовый массив 2».

В дополнение к функциям из оригинальной задачи, нужно также реализовать массовое изменение значения на отрезке и вычисление количества единиц на отрезке.

Сигнатура дополнительных функций, которые надо реализовать:

```
//установить в val значение всех k-ых битов для left <= k < right
void bitsetSetSeg(bitword *arr, int left, int right, int newval);
//посчитать, сколько битов равно единице на отрезке left <= k < right
int bitsetCount(const bitword *arr, int left, int right);
```

При помощи реализованной структуры нужно решить тестовую задачу.

Внимание: в этой задаче ограничение по времени выставлено очень сурово!

Формат входных данных

В первой строке записано целое число N — количество операций, которые нужно обработать ($1 \leqslant N \leqslant 10^5$). В каждой из следующих N строк описывается одна операция над битовым массивом.

Описание операции начинается с целого числа t, обозначающего тип операции. Если $t = 0$, то это операция `bitsetZero`, и вторым целым числом в строке указан размер массива `num`. Если $t = 1$, то это операция `bitsetGet`, и в строке также записано целое число `idx` — номер бита. Значение этого бита нужно выдать в выходной файл. Если $t = 2$, то это операция `bitsetSet`, и в строке также содержатся целые числа `idx` и `newval`. Здесь `idx` — номер бита, который нужно изменить, а `newval` — новое значение, которое надо записать (0 или 1). Если $t = 3$, то это операция `bitsetSetSeg`, и указано ещё три параметра `left`, `right` и `newval` — отрезок, на котором нужно выполнить присваивание, и новое значение для битов этого отрезка. Если $t = 4$, то это операция `bitsetCount`, и указано ещё два параметра `left` и `right` — отрезок, на котором нужно посчитать количество единичных битов. Полученный результат нужно выдать в выходной файл.

Размер массива `num` больше нуля и не превышает 10^7. Гарантируется, что после операции `bitsetZero` с параметром `num` все последующие операции обращаются только к битам с номерами от 0 до `num`-1 включительно — как минимум до следующего вызова `bitsetZero`.

Сумма значений `num` по всем операциям `bitsetZero` не превышает $2 \cdot 10^9$. Сумма длин отрезков (`right-left`) по всем операциям `bitsetSetSeg` не превышает $2 \cdot 10^9$, аналогично для операций `bitsetCount`.

Формат выходных данных

Для каждой операции `bitsetGet` или `bitsetCount` нужно вывести ответ в отдельной строке.

Пример

input.txt	output.txt
10 0 100 3 17 54 1 4 10 25 1 17 1 54 1 32 2 31 0 3 15 20 0 4 10 20 4 19 59	8 1 0 1 0 33

Комментарий

Вычисление количества единичных битов в целом числе — весьма интересная задача сама по себе. Для решения этой задачи вам потребуется какое-нибудь быстрое решение.

Много хороших вариантов можно найти в bit twiddling hacks (эта страница вообще очень познавательная) или в этом вопросе на SO.