

Задача 1. Биты и байты

Источник:	базовая
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Дана последовательность из N битов, каждый бит имеет значение 0 или 1. Нужно разделить эту последовательность на байты, по 8 битов в каждом (в последний байт может попасть меньше битов). После этого нужно распечатать значения всех полученных байтов в привычной людям десятичной системе исчисления.

Биты внутри байта записываются в привычном современным компьютерам порядке little-endian: сначала идут младшие биты, потом старшие.

Формат входных данных

В первой строке входного файла записано одно целое число N — количество битов в последовательности ($1 \leq N \leq 100\,000$). Во второй строке записано ровно N символов 0 или 1: значения битов последовательности.

Внимание: после второй строки файла символ перевода строки может быть, а может **не** быть.

Формат выходных данных

Выведите в одну строку через пробел десятичные значения полученных байтов.

Пример

<code>input.txt</code>	<code>output.txt</code>
34 0101000011111111000000011100101011	10 255 128 83 3

Пояснение к примеру

Разделим в примере биты на группы (байты). Обратите внимание, что в последнюю группу попадает только 2 бита. Далее преобразуем байты в десятичный вид:

- $01010000 = 2 + 8 = 10$
- $11111111 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$
- $00000001 = 128 = 128$
- $11001010 = 1 + 2 + 16 + 64 = 83$
- $11 = 1 + 2 = 3$

Задача 2. Количество единиц

Источник:	базовая
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
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

<code>input.txt</code>	<code>output.txt</code>
9	1
115	3

Задача 4. Численный эксперимент

Источник:	основная
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
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 ($1 \leq N \leq 200000$). 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 ($1000 < r_i < 10000, 0 < k_i \leq 16, 1 \leq i \leq N$).

Формат выходных данных

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

Пример

<code>input.txt</code>	<code>output.txt</code>
5	8
8824	64
64576	2
2554	8
16024	1024
10138624	

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:	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 функций 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 ($1 \leq N \leq 10^5$). 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 `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	1
0 100	0
2 30 1	0
2 31 1	some
2 32 1	none
1 31	some
1 7	none
2 31 0	0
1 31	
3 30 33	
3 31 32	
3 0 100	
3 45 67	
0 48	
1 30	

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 6. Битовый массив 2

Источник:	основная
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
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 ($1 \leq N, M \leq 100$).

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 7. Факториал

Источник:	основная
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Факториал от N (обозначается как $N!$) — это произведение всех целых чисел от 1 до N включительно.

Во входном файле задано одно целое число N в диапазоне $1 \leq N \leq 1\,000$. Нужно вычислить и распечатать факториал от N .

На этот раз вам понадобится обеспечить 3 000 цифр в структуре длинного числа, чтобы влез результат. Для вычисления ответа можно перебирать все k по порядку от 1 до N , и каждый раз домножать текущее длинное число на него.

Здесь нужно также реализовать умножение «в столбик», однако можно полагать множитель одной цифрой (даже, когда он больше десяти). В некотором смысле, эта операция называется умножением длинного числа (заданного массивом цифр) на короткое (заданное одним числом), и сама по себе она реализуется без вложенных циклов.

Пример

<code>input.txt</code>	<code>output.txt</code>
10	3628800

Задача 8. Битовый массив 3

Источник:	повышенной сложности
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	5 мегабайт

Это расширенная версия задачи «Битовый массив 1».

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 ($1 \leq N \leq 10^5$). 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	8
0 100	1
3 17 54 1	0
4 10 25	1
1 17	0
1 54	33
1 32	
2 31 0	
3 15 20 0	
4 10 20	
4 19 59	

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 по ссылке **bit twiddling hacks** :

<http://graphics.stanford.edu/~seander/bithacks.html>CountBitsSetTable

или в этом вопросе на SO:

<https://stackoverflow.com/questions/109023/how-to-count-the-number-of-set-bits-in-a-32-bit-integer>.

Дана последовательность из N целых чисел. Некоторые из них заданы явно (как число в виде десятичной записи), а некоторые определяются как сумма, разность или произведение двух более ранних чисел последовательности. Нужно найти и вывести все N чисел, выполнив заданные арифметические операции.

Формат входных данных

Если $t = 1$, то данное число определяется как сумма, если $t = 2$ — то как разность, а если $t = 3$ — то как произведение. В любом из этих трёх случаев далее через пробел указано два целых числа a и b — номера членов последовательности, над которыми нужно выполнить арифметическую операцию, чтобы получить k -ое число ($0 \leq a, b < k$).

Формат выходных данных

Пример

[illegible]

Комментарий

Страница 11 из 12

Задача 10. Длинное деление +

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

Во входном файле задано два целых положительных числа, одно в первой строке, второе — во второй строке. Нужно вычислить частное этих двух чисел (округлённое вниз) и вывести его в выходной файл.

Количество десятичных цифр в каждом входном числе может достигать 25 000.

Для решения этой задачи необходимо хранить не по одной десятичной цифре в каждом элементе массива, а хотя бы по четыре цифры, в некотором смысле, храня длинное число в 10 000-ичной системе исчисления. В процессе деления столбиком нужно на каждом шаге подбирать цифру ответа, и делать это перебором 10 000 возможных вариантов уже неприемлемо. Определение цифры следует осуществлять бинарным поиском.

Примеры

<code>input.txt</code>	<code>output.txt</code>
1023405633 84037	12178
10000000000000000000000000000000 7	14285714285714285714285714