Задача 1. Разница множеств

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

Дано два массива целых чисел A и B.

Требуется найти все такие значения элементов массива A, которых нет среди элементов массива B.

Замечание: В задаче необходимо использовать функцию `qsort` из стандартной библиотеки языка C.

Формат входных данных

В первой строке записано целое число N ($1 \leqslant N \leqslant 10^5$) — количество элементов массива A.

Во второй строке через пробел записано N целых чисел, каждое из которых не превосходит 10^9 по абсолютной величине — элементы массива A.

В следующих двух строках в аналогичном формате записаны элементы массива B.

Формат выходных данных

В первой строке выведите одно целое число — количество значений, удовлетворяющих описанному условию.

Во второй строке выведите все такие значения в порядке возрастания.

Примеры

input.txt	output.txt
7 1 2 3 3 6 8 8 4 1 3 7 9	3 2 6 8
3 1 2 3 3 3 2 1	0

Задача 2. Растущий массив

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

В данной задаче нужно реализовать массив переменного размера, в который можно дописывать элементы, не зная заранее его окончательный размер. Используя эту структуру данных, нужно решить приведённую ниже задачу.

В первой строке записано целое число N — количество записей ($1 \leqslant N \leqslant 2 \cdot 10^5$). В остальных N строках содержатся записи, по одной в строке.

Для каждой записи указаны ключ и значение через пробел. Ключ — это целое число в диапазоне от 0 до 10^6 включительно, а значение — это строка от одного до семи символов включительно, состоящая только из маленьких букв латинского алфавита.

Требуется вывести ровно те же самые N записей, но в другом порядке. Записи должны быть упорядочены по возрастанию ключа. Если у нескольких записей ключ равный, то нужно упорядочить их в том порядке, в котором они встречаются по входном файле.

Важно: Решать задачу **нужно** следующим образом (другие решения засчитываться **не** будут). Нужно завести 10^6 **массивов** переменного размера, и в каждый k-ый массив складывать все записи с ключом, равным k. После раскидывания записей по массивам достаточно будет пробежаться по массивам в порядке увеличения k и распечатать их.

Пример

input.txt	output.txt
7 3 qwerty 3 string 6 good 1 a 3 ab 2 hello 5 world	1 a 2 hello 3 qwerty 3 string 3 ab 5 world 6 good

Пояснение к примеру

В примере 7 записей с ключами 1, 2, 3, 5 и 6 — именно в таком порядке записи и выведены в выходном файле. Обратите внимание, что есть три записи с ключом 3: `qwerty`, `string`, `ab`. Они выведены ровно в том порядке, в котором они идут во входном файле.

Задача 3. Тасовка перфокарт

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	**специальное**

Имеется две колоды перфокарт: левая колода и правая. Изначально в каждой колоде ровно N перфокарт. В левой колоде перфокарты пронумерованы числами от 1 до N по порядку, если просматривать их сверху вниз. В правой колоде перфокарты пронумерованы числами от -1 до $-N$ по порядку, если просматривать их сверху вниз.

Для перемешивания колод нужно выполнить M заданных операций, каждая операция заключается в перекладывании одной карты. Каждая операция обозначается одной шестнадцатеричной цифрой в диапазоне от `0` до `F` (15) включительно. Операция определяется значениями четырёх битов в двоичной записи этой цифры:

- Если старший бит (8) единичный, то нужно взять карту с правой колоды, а иначе — с левой колоды.
- Если предпоследний бит (4) единичный, то нужно взять карту снизу колоды, а иначе — сверху колоды.
- Если второй бит (2) единичный, то нужно положить карту в правую колоды, а иначе — в левую.
- Если младший бит (1) единичный, то нужно положить карту в колоду снизу, а иначе — сверху.

Если в какой-то момент нужно выполнить операцию, которая предписывает взять карту из пустой колоды, то такую операцию нужно пропустить (ничего не делая).

В первой строке записано два целых числа: N — начальное количество карт в каждой колоде и M — сколько операций нужно выполнить ($1 \leqslant N \leqslant 5 \cdot 10^5$, $0 \leqslant M \leqslant 5 \cdot 10^6$).

Во второй строке записано подряд ровно M символов — описание операций в порядке их выполнения. Каждый символ является шестнадцатеричной цифрой и изменяется в диапазоне от `0` до `9` или от `A` до `F` включительно.

После выполнения всех операций требуется вывести содержимое левой колоды в первой строке выходного файла, и содержимое правой колоды — во второй строке. В каждой строке нужно сначала вывести целое число K — количество карт в колоде после выполнения всех операций, а затем через пробел K целых чисел — номера перфокарт в колоде, перечисленные в порядке сверху вниз.

Важно: Требуется хранить каждую колоду в **кольцевом буфере** размером ровно на $(2N + 1)$ элементов. Память под кольцевые буферы выделяйте динамически.

Пример

input.txt	output.txt
5 0	5 1 2 3 4 5 5 -1 -2 -3 -4 -5
5 10 180FA45DB2	6 -1 2 3 4 5 -5 4 1 -3 -4 -2
3 20 CCCCCCCCCC9999999999	6 -1 -2 -3 1 2 3 0

P.S. По второму примеру можно заметить, что команды `0`, `F`, `A` и `5` никогда ничего не изменяют.

Задача 4. Транзитивное замыкание

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

Дано бинарное отношение R над множеством чисел $X = \{1, 2, 3, \ldots, N\}$. Требуется найти его рефлексивно-транзитивное замыкание.

Указание: Используйте алгоритм Уоршалла (Флойда-Уоршалла), рассказанный на дискретной математике.

Формат входных данных

В первой строке записано одно целое число N — размер множества ($1 \leqslant N \leqslant 500$). Далее идёт N строк по N символов в каждой, задающие отношение R. j-ый символ i-ой строки равен 1, если пара $(i, j) \in R$ (т.е. лежит в отношении R), и равен 0 в противном случае.

Формат выходных данных

Выведите N строк по N символов в каждой — рефлексивное и транзитивное замыкание отношения R, описанное в том же формате, что и исходное отношение R во входных данных.

Пример

input.txt	output.txt
5 00001 10010 01101 10000 00011	10011 11011 11111 10011 10011

Задача 5. Вычисление синуса

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

В данной задаче нужно научиться вычислять синус. Использовать функцию `sin` из стандартной библиотеки или откуда-то ещё **запрещено**.

Подсказка: используйте ряд Тейлора.

Формат входных данных

В первой строке записано одно целое число N — количество аргументов, для которых нужно вычислить синус ($1 \leqslant N \leqslant 10^5$). Далее идёт N строк, по одному вещественному числу X в каждой. Каждое число — это число, синус которого надо вычислить.

Все числа X по абсолютной величине не превышают единицу (заданы в радианах).

Формат выходных данных

Выведите N строк, в каждой строке одно вещественное число, которое равно $\sin X$ для соответствующего аргумента X из входного файла.

Рекомендуется выводить числа с помощью формата `"%0.15lf"`, чтобы выводилось 15 знаков после десятичной точки. Ошибка в каждом вашем ответе не должна превышать 10^{-12}.

Пример

input.txt	output.txt
5 0.0 0.523598775598298 0.785398163397448 1.047197551196597 1.570796326794896	0.000000000000000 0.500000000000000 0.707106781186548 0.866025403784439 1.000000000000000

Задача 6. Топологическая сортировка

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

Имеется N лекционных тем, пронумерованных числами от 1 до N. За одну лекцию можно целиком рассказать одну тему, смешивать темы на лекции нельзя. Некоторые темы зависят от других тем. Если тема A зависит от темы B, то это означает, что нельзя рассказывать тему A, не рассказав предварительно тему B. Требуется определить, в каком порядке нужно рассказывать темы на лекции, чтобы соблюсти все зависимости.

Если есть несколько подходящих порядков, нужно выбрать лексикографически наименьший из них. То есть номер первой рассказываемой темы должен быть минимально возможный, среди таких порядков нужно выбрать такой, у которого номер темы на второй лекции минимально возможный, далее нужно минимизировать номер темы на третьей лекции, и так далее.

Указание: Используйте алгоритм Кана, рассказанный на дискретной математике.

Формат входных данных

В первой строке записано два целых числа: N — количество тем и M — количество зависимостей ($1 \leqslant N \leqslant 400$, $1 \leqslant M \leqslant \frac{1}{2}N(N-1)$).

В оставшихся M строках описаны зависимости. Каждая зависимость описывается двумя целыми числами B и A, что означает, что тема A зависит от темы B ($1 \leqslant A \neq B \leqslant N$).

Формат выходных данных

Выведите искомый порядок в единственную строку выходного файла: N целых чисел, обозначающих номера тем. Если в темах имеется циклическая зависимость и искомого порядка не существует, выведите слова “`bad course`” вместо порядка.

Пример

input.txt	output.txt
5 7 1 5 1 3 5 2 5 4 3 4 1 2 4 2	1 3 5 4 2
3 3 1 2 2 3 3 2	bad course

Задача 7. Топологическая сортировка+

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

Имеется N лекционных тем, пронумерованных числами от 1 до N. За одну лекцию можно целиком рассказать одну тему, смешивать темы на лекции нельзя. Некоторые темы зависят от других тем. Если тема A зависит от темы B, то это означает, что нельзя рассказывать тему A, не рассказав предварительно тему B. Требуется определить, в каком порядке нужно рассказывать темы на лекции, чтобы соблюсти все зависимости.

Если есть несколько подходящих порядков, нужно выбрать лексикографически наименьший из них. То есть номер первой рассказываемой темы должен быть минимально возможный, среди таких порядков нужно выбрать такой, у которого номер темы на второй лекции минимально возможный, далее нужно минимизировать номер темы на третьей лекции, и так далее.

Указание: Чтобы ускорить алгоритм Кана, для каждой темы храните и поддерживайте количество ещё не рассказанных тем, от которых она зависит.

Формат входных данных

В первой строке записано два целых числа: N — количество тем и M — количество зависимостей ($1 \leqslant N \leqslant 5\,000$, $1 \leqslant M \leqslant 100\,000$).

В оставшихся M строках описаны зависимости. Каждая зависимость описывается двумя целыми числами B и A, что означает, что тема A зависит от темы B ($1 \leqslant A \neq B \leqslant N$).

Формат выходных данных

Выведите искомый порядок в единственную строку выходного файла: N целых чисел, обозначающих номера тем. Если в темах имеется циклическая зависимость и искомого порядка не существует, выведите слова “`bad course`” вместо порядка.

Пример

input.txt	output.txt
5 7 1 5 1 3 5 2 5 4 3 4 1 2 4 2	1 3 5 4 2
3 3 1 2 2 3 3 2	bad course

Задача 8. Транзитивное замыкание+

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

Дано бинарное отношение R над множеством чисел $X = \{1, 2, 3, \ldots, N\}$. Требуется найти его рефлексивно-транзитивное замыкание.

Формат входных данных

В первой строке записано одно целое число N — размер множества ($1 \leqslant N \leqslant 2\,000$). Далее идёт N строк по N символов в каждой, задающие отношение R. j-ый символ i-ой строки равен 1, если пара $(i, j) \in R$ (т.е. лежит в отношении R), и равен 0 в противном случае.

Формат выходных данных

Выведите N строк по N символов в каждой — рефлексивное и транзитивное замыкание отношения R, описанное в том же формате, что и исходное отношение R во входных данных.

Пример

input.txt	output.txt
5 00001 10010 01101 10000 00011	10011 11011 11111 10011 10011

Задача 9. Скобки

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	разумное

Как многим известно, основу синтаксиса LISP-подобных языков составляют скобочки. Круглые, фигурные, квадратные — на любой вкус. Конечно, там применяют и другие символы, но главное — скобочки.

В лабораториях Ну Гляди-какого-умного Университета (НГУ) придумали новый язык программирования — Uncommon LISP. Он впитал в себя самую суть всех функциональных языков. В нём кроме скобочек нет ничего...

В первой версии языка корректной программой на Uncommon LISP считалась правильная скобочная последовательность из четырех видов скобок: (), {}, `[]` и <>. Правильная скобочная последовательность определяется следующим образом:

- пустая строка — правильная скобочная последовательность;
- правильная скобочная последовательность, взятая в скобки одного типа — правильная скобочная последовательность;
- правильная скобочная последовательность, к которой приписана слева или справа правильная скобочная последовательность — тоже правильная скобочная последовательность.

К сожалению, эти правила были слишком сложными, так что во второй версии языка (Uncommon LISP v2) корректными программами стали считаться также те программы, из которых можно получить правильную скобочную последовательность, переставляя скобки определённым образом. В последовательности можно переставлять местами скобки, стоящие рядом, если они обе открывающие или обе закрывающие. Такую операцию можно выполнять сколько угодно раз, в том числе по несколько раз переставляя одни и те же скобки.

Необходимо определить для данной скобочной последовательности, является ли она корректной программой для Uncommon LISP v2.

Формат входных данных

В первой строке входного файла записано целое число N — количество тестов, следующих ниже. Каждый тест — это непустая строка, состоящая из скобок. Суммарная длина всех строк не превышает 1 000 000.

Формат выходных данных

В выходной файл необходимо для каждого теста в отдельную строку вывести ответ `T`, если заданную скобочную последовательность можно привести к правильному виду, и `NIL`, если нельзя.

Пример

input.txt	output.txt
2 ([)])([]	T NIL