Задача 1. Сравнение асимптотик

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

В курсе всё больше и больше делается акцент на улучшение асимптотического времени работы различных операций. При этом важно понимать, при каких изменениях асимптотическое время работы становится лучше (т.е. быстрее), а при каких хуже. В данной задаче предлагается реализовать сравнение для наиболее часто встречающихся асимптотик.

В данной задаче асимптотическое время работы задаётся как функция вида:

$$T(N) = O(p^N \cdot N^s \cdot \log^l N)$$

Здесь $p \geqslant 1$, $s \geqslant 0$ и $l \geqslant 0$ — произвольные вещественные числа. Легко видеть, что в этот класс попадают, например, асимптотики сортировки слиянием $O(N \log N)$, бинарного поиска $O(\log N)$, перебора всех N-битных чисел $O(2^N N)$.

Формат входных данных

В первой строке входного файла записано число Q — сколько тестовых случаев нужно обработать ($1 \leqslant Q \leqslant 10^5$). Далее идёт $2Q$ строк, каждая пара строк описывает один тестовый случай, то есть две асимптотики, которые надо сравнить.

Асимптотика в полном виде записывается как: `"O( p^N N^s logN^l )"` (без кавычек). В полном виде в ней три части, обязательно отделённые друг от друга и от окружающих скобок пробелом. Других пробелов нет. Части могут быть записаны в произвольном порядке.

Вместо букв `p`, `s` и `l` в описании асимтотики записаны вещественные числа, задающие соответствующие коэффициенты. Все вещественные числа записаны с не более чем тремя знаками после десятичной точки, и лежат в пределах от 0 до 10 включительно. Кроме того, для коэффициента `p` верно: $p \geqslant 1$.

Кроме того, некоторые части могут быть опущены: в таком случае в произведении этой части нет. Если опущены все три части, то асимптотика будет записана в виде `"O( 1 )"` (без кавычек). Наконец, в компонентах `N^s` и `logN^l` может быть опущена степень: в таком случае она равна единице. Если степень опущена, то в описании отсутствует как вещественное число `s` или `l`, так и символ крышки непосредственно до него.

Замечание: рекомендуется использовать `gets`, `strtok`, `sscanf` и прочие стандартные функции для чтения асимптотики.

Формат выходных данных

В выходных данных должно быть ровно Q целых чисел, по одному числу в строке. Если в запросе первая асимптотика меньше второй, число должно быть равно -1. Если первая асимптотика больше второй, то нужно вывести 1. Наконец, если они совпадают, то нужно вывести 0.

Пример

input.txt	output.txt
6 O(2^N N^3.5 logN^7.3) O(2^N N^4 logN^7.267) O(N^3.5 logN^7) O(logN^7.000 N^3.5) O(1) O(N^2) O(N^0.5) O(logN^7) O(2^N N) O(2^N) O(N logN) O(N^1.5)	-1 0 -1 1 1 -1

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. 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 4. Вычисление экспоненты

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 экспоненту от заданного числа X, то есть e^X для e — натурального логарифма. Использовать функции `exp`, `pow` и подобные из стандартной библиотеки или откуда-то ещё **запрещено**. Лучше вообще `math.h` не подключать.

Подсказка: используйте ряд Тейлора.

Формат входных данных

В первой строке записано одно целое число N — количество аргументов, для которых нужно вычислить экспоненту ($1 \leqslant N \leqslant 10^4$).

Далее идёт N строк, по одному вещественному числу X в каждой. Каждое число — это число, экспоненту от которого надо вычислить.

Все числа X лежат в диапазоне $(-100, 100)$.

Формат выходных данных

Выведите N строк, в каждой строке одно вещественное число, которое равно $\exp(X) = e^X$ для соответствующего аргумента X из входного файла.

Следует выводить числа с помощью формата `"%0.15g"`, чтобы число выводилось в экспоненциальном виде с 15 знаками после десятичной точки.

Ответ считается верным, если его относительная ошибка не превышает 10^{-12}.

Пример

input.txt	output.txt
8 0.0 1.0 -1.0 2.0 -2.0 100.0 -100.0 0.189376476361643	1 2.71828182845905 0.367879441171442 7.38905609893065 0.135335283236613 2.68811714181614e+43 3.72007597602084e-44 1.20849583696666

Задача 5. Угол в треугольнике

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

Требуется найти угол $\angle BAC$ в заданном треугольнике ABC.

Формат входных данных

В первой строке задано целое число Q — количество тестовых случаев ($1 \leqslant Q \leqslant 3 \cdot 10^5$). В каждой из следующих Q строк описан один случай.

Каждый случай описывается шестью вещественными числами:

A_x, A_y — координаты вершины угла,

B_x, B_y — координаты другой вершины треугольника,

C_x, C_y — координаты третьей вершины.

Все вещественные числа заданы с максимально возможной точностью и по абсолютной величине не превышают 10^3.

Пусть M — максимум из абсолютных величин всех шести координат, заданных в тестовом случае.

Гарантируется, что $|AB|, |AC| > \frac{1}{20}M$. То есть стороны треугольника, инцидентные искомому углу, имеют довольно большую длину.

Формат выходных данных

Нужно вывести Q строк, в каждой из которых должен быть записан угол при вершине A в треугольнике ABC. Все углы нужно выводить с максимально возможной точностью, рекомендуется использовать формат `"%0.20g"`. Углы нужно выводить в градусах, в пределах от 0 до 180 градусов включительно.

Ответ засчитывается, если он отличается от истинного менее чем на 10^{-11}.

Пример

input.txt	output.txt
8 2 1 2 3 5 5 2 1 4 3 2 3 3 1 3 5 2 3 0 0 1 0 10 0 0 0 1 0 -10 1 7 4 3 3 3 3 0 0 1 0 1 1e-5 0 0 1 0 1e-5 1	36.869897645844019962 45 26.565051177077990019 0 174.28940686250035697 0 0.00057295779511172474814 89.999427042204885652