Задача A. Союзы

Имя входного файла:	*alliance.in*
Имя выходного файла:	*alliance.out*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В Триаметистовом королевстве было n городов и m дорог, соединявших некоторые из них друг с другом. Однажды в результате экспериментов придворного мага Д произошло катастрофическое расщепление: во вселенной вместо одного Триаметистового королевства появлось множество его копий, или *отражений*. Более того, в каждом из них каждая дорога стала заколдованной либо способом α, либо способом ω (стоит отметить, что каждое из возможных сочетаний заколдованностей дорог появилось ровно в одном из отражений).

Свойства заколодованностей α и ω таковы, что города a, b и c образуют торговый союз тогда и только тогда, когда есть дороги между a и b, между b и c и между c и a, заколдованные одним и тем же способом.

Формат входного файла

В первой строке входных данных записаны два целых числа n и m — количество городов и дорог Триаметистового королевства. В следующих m строках записаны пары чисел, задающие города, соединённые соответствующими дорогами. $3 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 500\,000$. Никакие два города не соединены более чем одной дорогой, никакая дорога не соединяет город сам с собой.

Формат выходного файла

Выведите как можно точнее единственное вещественное число — среднее количество союзов, которое образовалось во всех отражениях Триаметистового королевства.

Пример

alliance.in	alliance.out
3 3 1 2 2 3 3 1	0.25

Задача B. Коробки

Имя входного файла:	boxes.in
Имя выходного файла:	boxes.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дано N коробок в пространстве. Про каждую коробку известны её три измерения. Коробки можно вращать — то есть выбирать, какое измерение будет длиной, какое — шириной, а какое — высотой. Из этих коробок необходимо построить башню максимальной высоты. Коробки разрешается ставить друг на друга, только если длина коробки сверху не превосходит длины коробки снизу, а ширина коробки сверху — ширины коробки снизу.

Формат входного файла

В первой строке задано целое число N ($1 \leqslant N \leqslant 30$). Следующие N строк содержат по три целых числа a_i, b_i и c_i ($1 \leqslant a_i, b_i, c_i \leqslant 10^6$) — три измерения i-ой коробки.

Формат выходного файла

В первой строке выведите максимальную высоту башни H. Во второй строке выведите целое число k — количество коробок в искомой башне. В последующих k строках выведите по четыре числа для каждой коробки — её номер m_i, длину l_i, ширину d_i и высоту h_i (коробки нужно выводить в порядке снизу вверх). Коробки нумеруются с единицы в порядке, заданном во входном файле.

Пример

boxes.in	boxes.out
3 3 1 3 2 2 2 1 2 1	5 2 1 1 3 3 3 1 1 2

Задача C. ВнутренНЯя точка

Имя входного файла:	inside.in
Имя выходного файла:	inside.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дан строго выпуклый N-угольник и K точек. Для каждой точки нужно определеить, где она находится — внутри, на границе, или снаружи.

Формат входного файла

N ($3 \leqslant N \leqslant 10^5$). Далее N точек — вершины многоугольника.
K ($0 \leqslant K \leqslant 10^5$). Далее K точек — запросы.
Все координаты — целые числа по модулю не превосходящие 10^7.

Формат выходного файла

Для каждого запроса одна строка — `INSIDE`, `BORDER` или `OUTSIDE`.

Примеры

inside.in	inside.out
4 0 0 2 0 2 2 0 2 4 1 1 0 0 0 1 0 3	INSIDE BORDER BORDER OUTSIDE

Задача D. Обратные

Имя входного файла:	`inv.in`
Имя выходного файла:	`inv.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	64 мегабайта

Дано простое число n. Обратным к числу $1 \leqslant i < n$ называется такое j, что $i \cdot j = 1 \pmod n$. Можно доказать, что существует единственное обратное.

Для всех допустимых i найдите обратные к ним.

Формат входного файла

Во входном файле содержится простое число n ($2 \leqslant n \leqslant 10^6$).

Формат выходного файла

В выходной файл выведите $n - 1$ чисел, разделенных пробелами. i-е число означает обратное к i.

Примеры

inv.in	inv.out
5	1 3 2 4

Задача E. LCA offline

Имя входного файла:	`lca.in`
Имя выходного файла:	`lca.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Изначально имеется дерево состоящее только из корня (вершина с номером 1). Требуется научиться отвечать на следующие запросы:

- ADD a b — подвесить вершину b за вершину a (гарантируется, что вершина a уже существует).
- GET a b — вернуть LCA вершин a и b.

Все номера вершин от 1 до N.
В каждый момент времени у нас есть одно дерево.

Формат входного файла

В первой строке входного файла содержится число k — количество запросов. Следующие k строк содержат сами запросы. Гарантируется, что число запросов каждого из типов не превосходит 500 000.

Формат выходного файла

Для каждого запроса типа GET выведите в отдельную строку одно целое число — ответ на соответствующий запрос.

Примеры

lca.in	lca.out
9 ADD 1 2 ADD 1 3 ADD 2 4 GET 1 3 GET 2 3 GET 3 4 ADD 2 5 GET 4 5 GET 5 5	1 1 1 2 5

Задача F. Перестановки

Имя входного файла:	`permutation.in`
Имя выходного файла:	`permutation.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вася выписал на доске в каком-то порядке все числа от 1 по N, каждое число ровно по одному разу. Количество чисел оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая отвечает на вопрос — сколько среди чисел, стоящих на позициях с x по y, по величине лежат в интервале от k до l. Сделайте то же самое.

Формат входного файла

В первой строке лежит два натуральных числа — $1 \leqslant N \leqslant 100\,000$ — количество чисел, которые выписал Вася и $1 \leqslant M \leqslant 100\,000$ — количество вопросов, которые Вася хочет задать программе. Во второй строке дано N чисел — последовательность чисел, выписанных Васей. Далее в M строках находятся описания вопросов. Каждая строка содержит четыре целых числа $1 \leqslant x \leqslant y \leqslant N$ и $1 \leqslant k \leqslant l \leqslant N$.

Формат выходного файла

Выведите M строк, каждая должна содержать единственное число — ответ на Васин вопрос.

Пример

permutation.in	permutation.out
4 2 1 2 3 4 1 2 2 3 1 3 1 3	1 3

Задача G. Подстроки

Имя входного файла:	`substr.in`
Имя выходного файла:	`substr.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дана строка s. Вам требуется подсчитать количество её различных подстрок. Пустую строку учитывать не следует.

Формат входного файла

В единственной строке входного файла содержится данная строка s, состоящая из строчных латинских букв. Длина строки не превосходит 20 000 символов.

Формат выходного файла

В единственной строке выходного файла выведите единственное число — количество различных подстрок s.

substr.in	substr.out
aaaa	4
abacaba	21

Задача H. Такси

Имя входного файла:	`taxi.in`
Имя выходного файла:	`taxi.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Управлять службой такси — сосвем не простое дело. Помимо естественной необходимости централизованного управления машинами для того, чтобы обслуживать заказы по мере их поступления и как можно быстрее, нужно также планировать поездки для обслуживания тех клиентов, которые сделали заказы заранее.

В вашем распоряжении находится список заказов такси на следующий день. Вам необходимо минимимизировать число машин такси, необходимых чтобы выполнить все заказы.

Для простоты будем считать, что план города представляет собой квадратную решетку. Адрес в городе будем обозначать парой целых чисел: x-координатой и y-координатой. Время, необходимое для того, чтобы добраться из точки с адресом (a, b) в точку (c, d), равно $|a - c| + |b - d|$ минут. Машина такси может выполнить очередной заказ, либо если это первый ее заказ за день, либо она успевает приехать в начальную точку из предыдущей конечной хотя бы за минуту до указанного срока. Обратите внимание, что выполнение некоторых заказов может окончиться после полуночи.

Формат входного файла

В первой строке входного файла записано число заказов M $(0 < M < 500)$. Последующие M строк описывают сами заказы, по одному в строке. Про каждый заказ указано время отправления в формате `hh:mm` (в интервале с `00:00` по `23:59`), координаты (a, b) точки отправления и координаты (c, d) точки назначения. Все

координаты во входном файле неотрицательные и не превосходят 200. Заказы записаны упорядоченными по времени отправления.

Формат выходного файла

В выходной файл выведите единственное целое число — минимальное количество машин такси, необходимых для обслуживания всех заказов.

Пример

taxi.in	taxi.out
2 08:00 10 11 9 16 08:07 9 16 10 11	1
2 08:00 10 11 9 16 08:06 9 16 10 11	2

Задача I. Терминатор

Имя входного файла:	`terminator.in`
Имя выходного файла:	`terminator.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Два игрока играют в настольную игру. Игровое поле представляет собой квадратный лабиринт, 8×8 клеток. В некоторых клетках располагаются стенки. Один игрок управляет фишкой-терминатором, а второй — фишкой-беглецом. Игроки ходят по очереди, ходы пропускать нельзя (гарантируется, что ход всегда возможен). За один ход игрок может переместить свою фишку в любую из свободных клеток, расположенных рядом с исходной по горизонтали, вертикали или по диагонали (то есть ходом короля). Терминатор, кроме того, может стрелять в беглеца ракетами. Выстрел идет по прямой в любом направлении по горизонтали, вертикали или диагонали. Если беглец оказывается на линии выстрела терминатора и не прикрыт стенками, то терминатор незамедлительно делает выстрел (вне зависимости от того, чей ход), и беглец проигрывает. Начальное положение фишек задано. Первый ход делает беглец. Он выигрывает, если сделает ход с восьмой строки за пределы игрового поля, так как остальные границы поля окружены стенками.

Вопрос задачи: может ли беглец выиграть при оптимальной игре обеих сторон?

Формат входного файла

Во входном файле задано игровое поле. Свободная клетка обозначена цифрой 0, а клетка со стенкой — цифрой 1. Клетка, в которой находится беглец, обозначена цифрой 2, а клетка с терминатором — цифрой 3.

Формат выходного файла

В выходной файл выведите число 1, если беглец выигрывает, и -1 — в противном случае.

Примеры

terminator.in	terminator.out
01000000 10100000 31100000 00020000 00000000 00000000 00000000 00000000	-1