Задача A. Двоичное дерево

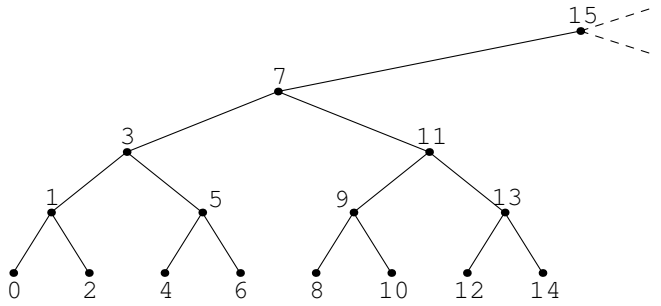
Имя входного файла:	`binary.in`
Имя выходного файла:	`binary.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Never underestimate the power of two!

цитата **7ania7** на www.topcoder.com

На Кафедре Оптимизации Деревьев Отрезков и (двоичных) Куч (сокр. КОДОК) знают, что номера вершинам в двоичных деревьях удобно присваивать сверху вниз: корень имеет номер 1, его дети — 2 и 3, и так далее. Действительно, удобно — чтобы найти «отца» какой-либо вершины, нужно всего лишь её номер поделить пополам.

Но в математике, как обычно, всё задом наперёд, и в этой задаче вершины нумеруют слева направо (см. рисунок). Ваша же задача проста — найти сумму номеров вершин на пути от вершины a до вершины b. Считайте, что корень дерева имеет номер 55 213 970 774 324 510 299 478 046 898 216 203 619 608 871 777 363 092 441 300 193 790 394 367.



Формат входного файла

В первой и единственной строке входного файла даны два числа: a и b — это номера вершин ($0 \leqslant a, b \leqslant 10^{15}$).

Формат выходного файла

В единственной строке выходного файла должно быть одно число — сумма номеров вершин на пути от вершины с номером a до вершины с номером b.

Примеры

binary.in	binary.out
1 5	9
3 4	12

Задача B. Машинки

Имя входного файла:	`cars.in`
Имя выходного файла:	`cars.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася проводит ежегодные соревнования в ИРМа (Институте Радиоуправляемых Машинок). Соревнования проходят следующим образом. На игровой площадке заранее выбираются стартовые позиции, а также фиксируются зоны — прямоугольники со сторонами, параллельными сторонам площадки. Машинки игроков начинают из исходных позиций и должны побывать внутри каждой зоны.

Но сами соревнования только завтра, а сейчас Вася проводит тестирование. Для этих целей у него есть одна эталонная машинка и уже размеченная игровая площадка. Вася собирается последовательно провести эту машинку из каждой стартовой позиции до каждой зоны (если стартовая позиция находится внутри зоны, перемещение не потребуется). Естественно, каждый раз Вася будет выбирать кратчайший путь. Известно, что время, требуемое для прохождения x метров, равно x^2 секунд (время на разгон, торможение... — но не будем утомлять вас физическими подробностями).

Подсчитайте, через сколько секунд Вася наконец сможет закончить проверку трассы. Временем, необходимым на перемещение машинки вручную от одной стартовой позиции до другой, а также от зоны обратно к стартовой позиции, можно пренебречь.

Формат входного файла

Первая строка ввода содержит целые числа n и k — количество стартовых позиций и зон, соответственно ($1 \leqslant n, k \leqslant 100\,000$). Следующие n строк содержат по два целых числа x_i и y_i — координаты стартовых позиций. Следующие k строк содержат по четыре целых числа каждая — координаты левого нижнего и правого верхнего углов зоны, соответственно.

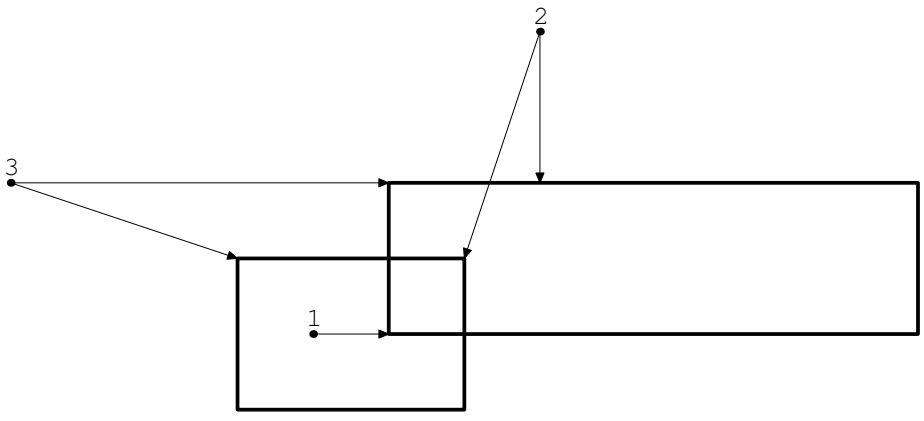
Все координаты не превосходят 10^6 по абсолютной величине и заданы в метрах.

Формат выходного файла

Выведите единственное целое число — количество секунд, требуемых на тестирование.

Пример

cars.in	cars.out
3 2 2 2 5 6 -2 4 1 1 4 3 3 2 10 4	50



Задача C. Знак определителя

Имя входного файла:	`detsign.in`
Имя выходного файла:	`detsign.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Первый Институт Линейной Алгебры (ПИЛА) поручил Васе найти знак определителя сильно разреженной квадратной матрицы. Почти сразу Вася обнаружил, что в каждом столбце матрицы содержится не более одного ненулевого числа. К его глубокому разочарованию, размеры матрицы оказались просто огромными! Помогите ему написать программу, решающую поставленную перед ним задачу.

Напомним два эквивалентных определения $\det A$ — *определителя* квадратной матрицы A размера $n \times n$: через разложение по строке и через перестановки.

$$\textbf{1.}\ \det A = \sum_{i=1}^{n} (-1)^{1+i} a_{1,i} \det A_i^1,$$

где A_q^p — это матрица $(n-1) \times (n-1)$, полученная из матрицы A вычёркиванием p-й строки и q-го столбца (здесь строки и столбцы нумеруются с единицы).

$$\textbf{2.}\ \det A = \sum_{p \in S_n} (-1)^{N(p)} a_{1,p_1} a_{2,p_2} a_{3,p_3} \cdots a_{n,p_n},$$

где S_n — группа всех перестановок порядка n, а $N(p)$ — количество инверсий в перестановке p, то есть таких пар индексов i и j $(1 \leqslant i < j \leqslant n)$, что $p_i > p_j$.

Формат входного файла

Первая строка входного файла содержит два целых положительных числа m и n, где n — это размер матрицы $(1 \leqslant n \leqslant 5\,000\,000)$. Следующие m строк входного файла содержат сокращённое описание матрицы, для которой надо вычислить знак определителя — i-я из них содержит четыре целых числа k_i, r_i, d_i и a_i $(1 \leqslant r_i \leqslant n,\ 0 \leqslant d_i < n,\ |a_i| \leqslant 10^9$; гарантируется, что сумма всех k_i равна $n)$. Первая из этих строк задаёт числа в первых k_1 столбцах — нужно поставить число a_1 в каждую из ячеек $(r_1,\ 1)$, $((r_1 + d_1) \overline{\mathrm{mod}}\ n,\ 2)$, $((r_1 + 2d_1) \overline{\mathrm{mod}}\ n,\ 3)$, …, $((r_1 + (k_1 - 1)d_1) \overline{\mathrm{mod}}\ n,\ k_1)$; вторая задаёт числа в следующих k_2 столбцах — нужно поставить число a_2 в каждую из ячеек $(r_2,\ k_1 + 1)$, $((r_2 + d_2) \overline{\mathrm{mod}}\ n,\ k_1 + 2)$, $((r_2 + 2d_2) \overline{\mathrm{mod}}\ n,\ k_1 + 3)$, …, $((r_2 + (k_2 - 1)d_2) \overline{\mathrm{mod}}\ n,\ k_1 + k_2)$; и так далее. Последняя из этих строк задаёт числа в последних k_m столбцах — нужно поставить число a_m в каждую из ячеек $(r_m,\ n - k_m + 1)$, $((r_m + d_m) \overline{\mathrm{mod}}\ n,\ n - k_m + 2)$, $((r_m + 2d_m) \overline{\mathrm{mod}}\ n,\ n - k_m + 3)$, …, $((r_m + (k_m - 1)d_m) \overline{\mathrm{mod}}\ n,\ n)$. Здесь первое число в скобках — это номер строки, а второе — номер столбца; выражение $a \overline{\mathrm{mod}}\, b$ означает $((a-1) \bmod b) + 1$. Во всех остальных ячейках матрицы стоят нули. Гарантируется, что размер входного файла не превышает одного мебибайта.

Формат выходного файла

Выходной файл должен содержать одну строку, состоящую из одного символа. Если определитель равен нулю, выведите '0', иначе выведите знак определителя ('+' или '-').

Примеры

detsign.in	detsign.out
1 30 30 1 1 1	+
1 239 239 1 1 -1	-
1 10 10 1 0 1	0
2 2 1 1 0 0 1 2 0 1	0

Задача D. Римская дробь

Имя входного файла:	`fraction.in`
Имя выходного файла:	`fraction.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Сотруднику НИИ Данных Строк Васе поручили разработать эффективный метод представления вещественных чисел строками, не содержащими арабских цифр. После длительных размышлений Вася решил аппроксимировать вещественные числа рациональными дробями, у которых числитель и знаменатель записаны римскими цифрами.

На заданном отрезке $[\alpha, \beta]$ Вася хочет найти рациональное число $\frac{A}{B}$, удовлетворяющее следующим условиям:

- числа A и B целые и лежат в промежутке $[1, 999\,999]$,
- сумма длин представлений чисел A и B римскими цифрами минимальна.

Запись числа от 1 до 999 римскими цифрами **в этой задаче** устроена следующим образом. Сначала записывается количество сотен, затем количество десятков, и, наконец, количество единиц. Числа от 1 до 9 записываются так: `I`, `II`, `III`, `IV`, `V`, `VI`, `VII`, `VIII`, `IX`. Десятки (от 10 до 90) записываются так: `X`, `XX`, `XXX`, `XL`, `L`, `LX`, `LXX`, `LXXX`, `XC`. Сотни (от 100 до 900) записываются так: `C`, `CC`, `CCC`, `CD`, `D`, `DC`, `DCC`, `DCCC`, `CM`. Если какой-то из десятичных разрядов равен нулю, он никак не записывается.

Запись числа от 1 до 999 999 римскими цифрами выглядит как $\overline{S}T$, где строки S и T — это записи чисел от 1 до 999. Само число при этом получается как $1000 \cdot num(S) + num(T)$, где num — это значение соответствующего числа.

Примеры записи чисел римскими цифрами:

- $\overline{XXX}CCXXXIX = 30\,239$
- $\overline{DCCCLXXXVIII} = 888\,000$
- $\overline{CMXCIX}CMXCIX = 999\,999$

Помогите Васе написать программу, решающую поставленную перед ним задачу.

Формат входного файла

Первая строка входного файла содержит два вещественных числа α и β ($0 < \alpha \leqslant \beta < 1$). Вещественные числа во входном файле даны с точностью не более девяти цифр после точки.

Формат выходного файла

Выходной файл должен содержать одну строку, состоящую из одного положительного целого — минимальной суммы длин представлений чисел A и B римскими цифрами. Выведите "`IMPOSSIBLE`", если невозможно найти рациональное число, удовлетворяющее условиям.

Примеры

fraction.in	fraction.out
0.2 0.2	2
0.123456789 0.123456789	IMPOSSIBLE
0.1 0.9	2

Задача E. Честное деление

Имя входного файла:	`honest.in`
Имя выходного файла:	`honest.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася работает в НИИЧДТ (НИИ честности деления торта). И однажды начальник пригласил его на чай. У них на двоих был один прямоугольный торт (размера $W \times H$). Кроме того, на этом торте было две вишенки. Если смотреть от левого нижнего угла, то одна из них имела координаты (x_1, y_1), а вторая (x_2, y_2). Чтобы проверить, насколько Вася хорошо работает, начальник предложил Васе по-честному разделить торт одним прямым разрезом. Деление считается честным, если, во-первых, две части окажутся равными по площади, а во-вторых, на каждом кусочке окажется по одной вишенке. Проводить разрез через вишенку нельзя. Помогите Васе!

Формат входного файла

В первой строке входного файла заданы два целых числа $2 \leqslant W, H \leqslant 10\,000$. Во второй строке содержатся координаты первой вишенки — целые числа x_1 и y_1 ($0 < x_1 < W$, $0 < y_1 < H$). В третьей строке аналогично заданы координаты второй вишенки — целые числа x_2 и y_2 ($0 < x_2 < W$, $0 < y_2 < H$). Гарантируется, что вишенки находятся в разных точках.

Формат выходного файла

Если ответ существует, то в первой строке выведите слово "`YES`", а во второй — четыре целых числа в промежутке от -10^9 до 10^9 — координаты двух различных точек, через которые надо провести разрез. Если ответа не существует, то выведите единственное слово "`NO`". Гарантируется, что если ответ на пример существует, то существует и такой ответ, который подходит под ограничения вывода.

Пример

honest.in	honest.out
6 6 1 1 5 5	YES 3 0 3 1

Задача F. Сок

Имя входного файла:	`juice.in`
Имя выходного файла:	`juice.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася — завхоз в ИСТерИКА (Институте Соковой Терапии Индифферентных Конечных Автоматов). Неудивительно, что именно в его обязанности входит обеспечение структурных подразделений института соком.

Как известно, сок выпускается в коробках по $a_1, a_2, \ldots, a_n$ литров. Поставщиков же Васе известно всего k, при этом у i-го поставщика коробка в a_j литров стоит $x_i a_j + y_i$ рублей. Как показало более тщательное исследование предложений, у i-го поставщика также отсутствуют в продаже коробки по a_{q_i} литров (а все остальные ёмкости доступны для заказа в любом количестве).

Помогите Васе выбрать одного поставщика для заключения договора на приобретение суммарно w литров сока (не больше и не меньше), минимизируя затраты.

Формат входного файла

Первая строка ввода содержит целые числа n, k и w — количество видов коробок сока, количество поставщиков и объём заказа, соответственно ($1 \leqslant n, k, w \leqslant 5\,000$). Следующая строка содержит n различных целых чисел a_i — объёмы коробок ($1 \leqslant a_i \leqslant 5\,000$). Следующие k строк содержат по 3 целых числа x_i, y_i и q_i каждая — составляющие цены, а также номер отсутствующего в продаже объёма сока ($0 \leqslant x_i, y_i \leqslant 10^4$, $1 \leqslant q_i \leqslant n$).

Формат выходного файла

Если ровно w литров сока приобрести не удастся, выведите два нуля. Иначе выведите две строки: первую, содержащую итоговую стоимость закупки и номер поставщика и вторую, содержащую n целых чисел — количества коробок соответствующего объёма, которые нужно заказать. Поставщики нумеруются целыми числами от 1 до k в порядке, в котором они перечислены во входных данных.

Если оптимальных решений несколько, выведите любое.

Примеры

juice.in	juice.out
3 4 29 1 5 10 1 0 1 1 11 2 1 2 3 1 4 2	47 3 4 5 0
2 1 5 2 3 0 0 1	0 0

Задача G. Песцы

Имя входного файла:	`polarfox.in`
Имя выходного файла:	`polarfox.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Песец подкрадывается!

Николай Дроздов

Вокруг северного полюса (по окружности, соответствующей постоянной широте) бегают n песцов. В данный момент, i-й песец находится в точке с долготой a_i и бежит по часовой стрелке с постоянной скоростью v_i градусов в секунду.

Время от времени случается, что один песец догоняет другого. Тогда догнавший поедает своего собрата и продолжает бежать со своей скоростью. Поскольку песцы очень голодные, временем поедания можно пренебречь. Если предоставить песцов самих себе, то оставшиеся после поеданий песцы будут бегать по кругу до конца долгого полярного дня.

Пока этого не произошло, вам, как сотруднику НИИСПАТ (Научно-Исследовательского Института Северного Полюса и Арктических Течений), требуется поймать как можно больше песцов. Для этого вам надо встать в произвольную точку на окружности (вы не можете в точности встать в точку, где находится песец, но можете встать в сколь угодно приближенную к ней точку), после чего начать бежать по часовой стрелке со скоростью v_{you}. Догнав песца, вы его ловите и кладёте в корзинку.

Покинуть круг придётся, если не останется больше песцов, которых вы можете поймать таким образом, или если какой-то песец сам вас догонит и *очень* больно укусит.

Формат входного файла

Первая строка входного файла содержит натуральное число n (количество песцов в кругу в тот момент, когда вы на него встанете, $n \leqslant 100\,000$) и вашу скорость v_{you} с тремя знаками после запятой ($0.001 \leqslant v_{you} \leqslant 360.000$), в градусах в секунду. Каждая из следующих n строк содержит описание одного песца, которое состоит из двух чисел: координаты и скорости песца. Гарантируется, что координаты двух различных песцов не совпадают.

Формат выходного файла

Единственная строка выходного файла должна содержать максимальное количество песцов, которое вы можете поймать.

Примеры

polarfox.in	polarfox.out
2 0.500 90.000 0.300 270.000 0.700	1
4 12.000 60.000 11.000 61.000 10.000 242.000 11.000 243.000 10.000	3

Примечание

Долгота в этой задаче измеряется с востока на запад и принимает значения от 0 до 360 градусов. Таким образом, песец, двигающийся со скоростью 1.0 градус/секунда, за одну секунду может перебежать из точки с долготой 30.0 градусов в точку с долготой 31.0 градус.

Задача H. Ребус

Имя входного файла:	`rebus.in`
Имя выходного файла:	`rebus.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вы любите ребусы? Вот и Вася, работающий в НИИ ГРИПИП (головоломок, ребусов и прочего и прочего) их любит. Однажды Вася и несколько других сотрудников этого НИИ исследовали царь-ребус — левая часть была одним словом из N ($N \leqslant 10^6$) букв. Но вдруг их всеобщий любимец таракан Петя съел всю правую часть ребуса. Как он это смог сделать — неважно, важно то, что Вася остался только с Петей и левой частью ребуса. Всё, что он помнил, так это то, что левая часть делится на k. Помогите Васе понять, какое минимальное значение может принимать левая часть.

Напомним, что ребус — это равенство, составленное из целых неотрицательных чисел в десятичной записи, причём каждая цифра заменена на какую-то букву (одинаковые цифры на одинаковые буквы, а разные на разные). Решить ребус — значит заменить буквы на цифры так, чтобы равенство оказалось верным. Числа, зашифрованные в ребусе, не могут иметь лидирующих нулей; число 0 записывается ровно одной цифрой.

Формат входного файла

В первой строке содержится строка, состоящая из нескольких заглавных латинских букв. Длина строки лежит в пределах от 1 до 10^6. Во второй строке содержится целое число k ($1 \leqslant k \leqslant 10^6$).

Формат выходного файла

Выведите единственное целое число — минимальную левую часть, или -1, если ответа не существует.

Примеры

rebus.in	rebus.out
MAMA 3	1212
SPBSUCHAMPIONSHIPXXVI 1	-1
I 1	0

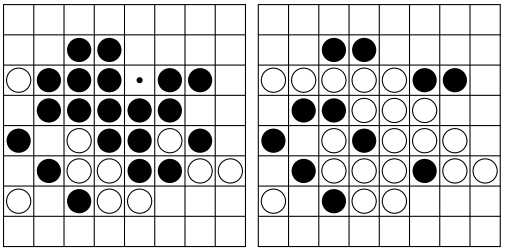
Задача I. Реверси

Имя входного файла:	`reversi.in`
Имя выходного файла:	`reversi.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

При игре в Реверси используются доска 8×8 и 64 фишки. Одна сторона каждой из фишек белая, а другая — чёрная. Один игрок выставляет фишки чёрной стороной вверх, а его противник — белой.

Перед началом партии игроки выставляют по две фишки в центре доски. Начинает игрок, играющий чёрными фишками. Игроки ходят по очереди. Фишка должна выставляться на свободную клетку рядом с фишкой противника таким образом, чтобы хотя бы одна из фишек противника оказалась зажатой по вертикали, горизонтали или диагонали между только что поставленной и уже стоящей на доске фишками того цвета, которым владеет игрок, делающий ход. Одним ходом можно зажать и более одной фишки противника, если зажимаемая цепь фишек противника расположена на одной прямой и не имеет разрывов. Все зажатые фишки переворачиваются **одновременно** и становятся фишками того цвета, что и фишка игрока, сделавшего ход.

Если один из игроков не может сделать ни одного хода, его противник должен совершать ходы, пока они не появятся у его партнёра. Игрок не имеет права отказываться от хода из тактических соображений. Игра заканчивается, если ни один из игроков не может сделать ход. Обычно это происходит, когда все поля заполнены фишками. Частный случай этого правила возникает, когда у одного из игроков нет фишек своего цвета. Побеждает игрок, у которого на доске окажется больше фишек. Если фишек поровну, игра завершается вничью.



Помогите Васе, сотруднику НИИ Фишечных Игр, Глобально Анализируемых (НИИФИГА), написать программу, определяющую, как завершится игра, при условии, что обе стороны играют наилучшим образом.

Формат входного файла

Первые восемь строк входного файла описывают расположение фишек на доске. Символ '`W`' обозначает белую фишку, '`B`' — чёрную, '`.`' — пустую клетку. Девятая строка состоит из единственного символа, определяющего цвет фишек игрока, который ходит в заданной позиции. Гарантируется, что количество пустых клеток не превышает 12.

Формат выходного файла

Выведите "`DRAW`", если встреча завершится вничью. Выведите "`WHITE`", если выиграет игрок, играющий белыми фишками. Если выиграет игрок, играющий чёрными фишками, выведите "`BLACK`".

Пример

reversi.in	reversi.out
..BBBB. W.BWBW.W WWWWWWWW WBBBBBWW WBBWBBWW WWWWWBWW W.WBWW.W ..WBBW.. B	BLACK

Задача J. Физкультура

Имя входного файла:	`sport.in`
Имя выходного файла:	`sport.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

На паре по физкультуре n студентов выстроились в шеренгу. Когда с Кафедры Различных Упражнений на Технику Отжимания (сокр. КРУТО) пришёл преподаватель (не будем называть его имени, его и так все знают), он не очень обрадовался.

Дело было в том, что студенты должны стоять по росту, а они стояли как попало — у первого в шеренге был рост a_1, у второго a_2, и так далее. «Непорядок, — заметил преподаватель — ну-ка, встаньте как надо». Однако ленивые студенты и не подумали меняться местами или куда-то ходить. Но, чтобы преподаватель не грустил, они сказали, что стоят не в одной шеренге, а в нескольких.

Так, если четыре студента ростом 176, 174, 178, 168 сантиметров стоят в таком порядке, можно сказать, что на самом деле это две шеренги по два студента в каждой. И тогда в обеих шеренгах студенты будут стоять как надо — по невозрастанию роста.

Помогите студентам подсчитать количество способов разбиться на шеренги. Каждая шеренга должна состоять из нескольких (хотя бы одного) стоящих подряд студентов, каждый следующий должен быть не выше предыдущего. Каждый студент должен принадлежать ровно одной шеренге.

Формат входного файла

В первой строке входного файла содержится целое число n — количество студентов, пришедших на пару ($0 \leqslant n \leqslant 100\,000$). Во второй строке содержится n целых чисел $a_1, a_2, \ldots, a_n$ — рост студентов ($1 \leqslant a_i \leqslant 10^9$).

Формат выходного файла

В единственной строке выходного файла должно содержаться одно число — количество способов разбить студентов на шеренги.

Пример

sport.in	sport.out
4 176 174 178 168	4

Задача K. Треугольник

Имя входного файла:	`triang.in`
Имя выходного файла:	`triang.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ Треугольников и Плоскости (НИИ ТреПло). Ему срочно требуется решить следующую задачу. Даны n точек на плоскости. Нужно построить треугольник с вершинами в каких-то трёх из этих точек такой, что его площадь минимальна, но больше нуля.

Формат входного файла

В первой строке входного файла содержится целое число n ($3 \leqslant n \leqslant 2\,000$). В следующих n строках содержатся координаты точек в формате x_i y_i. Все координаты целые и по модулю не превосходят 10^9.

Формат выходного файла

В единственной строке выходного файла должно содержаться одно число — минимальная возможная площадь. Максимальная допустимая ошибка — 0.1. Гарантируется, что хотя бы один треугольник с положительной площадью построить можно.

Пример

triang.in	triang.out
4 0 0 10 0 5 10 5 6	10.0