Задача 1. Перекраска крыш

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Король Берляндии очень любит порядок. Например, столица Берляндии на карте выглядит как прямоугольное клеточное поле, на котором каждая клетка является кварталом.

Недавно он издал указ, чтобы в каждом квартале все крыши домов были покрашены в один из двух цветов — красный или синий, причём внутри каждого квартала цвет всех крыш должен быть одинаковый. Теперь в столице Берляндии ожидают появления ревизоров, которые будут проверять исполнение указа. Ревизоры будут ходить по городу, периодически переходя из квартала в соседний по стороне квартал. Для успешности проверки необходимо, чтобы ревизоры могли добраться из любого квартала в любой.

Министр финансов через знакомых выяснил, что у ревизоров есть занятная фобия. Оказалось, что они не могут перейти в соседний квартал, если в нём такой же цвет крыш, как в текущем квартале. Таким образом, в некоторых кварталах придётся перекрасить крыши всех домов, чтобы пройти проверку. Министр финансов как обычно хочет сэкономить деньги, поэтому просит Вас определить минимальное количество кварталов, в которых придётся перекрасить все крыши.

Формат входных данных

В первой строке входного файла записаны два целых числа m и n — количество строк и количество столбцов на карте столицы ($1 \leqslant m, n \leqslant 1000$).

Далее следуют m строк, состоящих из n символов, каждый из которой либо '1', либо '2'. Символ '1' означает, что в соответствующем квартале все крыши синие, а символ '2' — что все крыши имеют красный цвет.

Формат выходных данных

В выходной файл необходимо вывести одно целое число — минимальное количество кварталов, в которых потребуется перекрасить все крыши.

Пример

input.txt	output.txt
1 4 2211	2

Задача 2. Симметричная матрица

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана квадратная матрица, имеющая n строк, все элементы которой — целые числа. Разрешается за одно действие переставить местами два элемента в матрице. Требуется определить, за какое минимальное количество действий можно получить из данной матрицы симметричную. Напомним, что симметричной называется матрица, у которой на пересечении i-ой строки и j-го столбца стоит такой же элемент, что и на пересечении j-ой строки и i-го столбца для любых i, j.

Гарантируется, что в заданной матрице n элементов встречаются ровно один раз, все остальные встречаются по два раза.

Формат входных данных

В первой строке входного файла записано целое число n — количество строк в матрице ($1 \leqslant n \leqslant 500$).

Далее следуют n строк заданной матрицы. Каждая из них содержит n целых чисел, разделённых пробелом, по модулю не превосходящих 10^9.

Формат выходных данных

В первую строку выходного файла необходимо вывести одно целое число m — минимальное количество действий, за которое можно получить симметричную матрицу. Далее нужно вывести пример таких m действий. Для i-го действия выведите четыре целых числа a_i, b_i, c_i, d_i в отдельную строку, которые означают, что требуется переставить элемент, стоящий в a_i-ой строке, в b_i-ом столбце с элементом, стоящим в c_i-ой строке, в d_i-ом столбце.

Примеры

input.txt	output.txt
2 1 2 3 1	2 1 1 1 2 2 1 2 2
3 1 4 -3 4 2 5 6 6 5	2 3 3 3 2 3 3 1 3

Задача 3. Создание карты

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Вася по-прежнему любит играть в компьютерные игры, и всё ещё работает тестировщиком игр. Но он уже потерял надежду, что будет заниматься чем-то мало-мальски интересным на работе. Чтобы дать выход своему творческому потенциалу, он скооперировался с коллегами Петей и Колей и вместе с ними работает над хобби-проектом. Петя рассказал Васе, что сегодня есть много успешных инди-игр, и даже маленькой командой можно сделать что-нибудь крутое. Правда он умолчал о том, какая доля таких игр становится известной, и какая доля таких проектов вообще доходит до конца.

Новая игра планируется в популярном нынче жанре roguelike. Хотя бородатый Коля каждый раз поправляет, что правильнее говорить roguelite или roguelikelike. Одна из важнейших составляющих любой roguelike игры — это генерация игрового мира случайным образом. Новое приключение при каждом перезапуске игры необходимо, чтобы формула “Losing is fun” работала. Сейчас друзья работают над генерацией карты мира для своей игры.

Действие игры происходит в пещерах. Карта мира представляется прямоугольным клеточным полем. Каждая клетка поля либо проходима и содержит уровень, который надо пройти игроку, либо непроходима, и в ней ничего нет. Игрок начинает игру в самой верхней левой клетке поля, которая должна быть проходимой. Пройдя очередной уровень, игрок может переместиться по карте мира вниз или вправо в соседнюю по стороне клетку, если эта клетка проходима. Находясь в последней строке или в последнем столбце поля, игрок также может выйти за пределы поля. В этом случае игра заканчивается, и он побеждает.

Исходя из соображений баланса, Петя наложил дополнительные ограничения на карту:

1. Игрок может попасть в каждую проходимую клетку из начальной клетки по единственному пути.
2. От каждой проходимой клетки игрок может дойти до победы, хотя способов сделать это может быть несколько.

Коля — главный программист в команде, и ему писать код генерации. Он заметил, что если генерировать поле случайным образом, то оно практически никогда не удовлетворяет требованиям. Поэтому он генерирует случайным образом только некоторые клетки поля, оставляя остальные в «неопределённом» состоянии. Дальше нужно запустить алгоритм, который определит содержимое оставшихся клеток так, чтобы карта удовлетворяла всем условиям. Для того чтобы в игре было побольше контента, алгоритм должен делать количество проходимых клеток максимально возможным.

Беда в том, что Коля не силён в алгоритмическом программировании и не может сам реализовать последний шаг. Помогите друзьям осуществить их мечту.

Формат входных данных

В первой строке входного файла содержится два целых числа, задающие размеры игрового поля: H — количество строк и W — количество столбцов ($1 \leqslant H, W \leqslant 18$).

Далее задаётся содержимое поля в виде H строк по W символов в каждой. Проходимые клетки задаются символом точки `'.'` (ASCII 46), непроходимые — буквой `'X'` (ASCII 88), а неопределённые — символом вопроса `'?'` (ASCII 63).

Формат выходных данных

Если доопределить игровое поле до корректного нельзя, в выходной файл требуется вывести число -1.

В противном случае, в первую строку выведите целое число K — количество проходимых клеток в доопределённом игровом поле. Это число должно быть максимально возможным. Далее нужно вывести доопределённое игровое поле в том же формате, в котором оно задаётся во входных данных, разумеется, уже без символов вопроса.

Если оптимальных решений несколько, выведите любое из них.

Примеры

input.txt	output.txt
4 6 ??X??? .????? ????.. ?X?X?.	13 .XXXXX .....X .X.X.. .X.XX.
3 3 ??X ?X? X??	-1

Задача 4. Облачные вычисления

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	2 секунды
	5 секунд (для Java)
Ограничение по памяти:	256 мегабайт

Набирающие в последнее время популярность облачные вычисления предоставляют их пользователям широкие возможности. В то же время они обладают ощутимым недостатком: обработка ваших данных на чужом компьютере существенно снижает вашу информационную безопасность.

Ваня работает в Организации, которая внедрила использование облачных вычислений для нахождения порядковых статистик массива. Порядковой статистикой массива для заданного числа k называется элемент, который окажется на k-ом месте в массиве, если данный массив будет отсортирован.

Однако массив, в котором нужно находить порядковые статистики, является секретным. Единственное, что известно про этот массив, это то, что все его элементы различны. По этой причине решено было реализовать следующую схему: массив хранится на сервере Организации, а облачный сервер, выполняющий нахождение порядковых статистик, может обращаться к серверу Организации, чтобы узнать результат сравнения двух элементов массива. Таким образом, облачный сервер может определить позицию, на которой находится k-ая порядковая статистика, при этом сам секретный массив ему не раскрывается. Правда возникает другая проблема: нельзя делать слишком много запросов от облачного сервера к серверу Организации.

В частности, для нахождения второй порядковой статистики решено было ограничиться не более чем $N + 20$ запросами, где N — это размер массива. Помогите Ване реализовать алгоритм нахождения второго по величине элемента секретного массива, соблюдая данное ограничение.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

При старте вашей программы в стандартный поток ввода подаётся целое число N — размер массива ($2 \leqslant N \leqslant 10^5$). Далее ваша программа должна отправлять запросы в стандартный поток вывода.

Каждый запрос должен состоять из одной строки, в которой после знака «?» через пробел записаны два различных числа i и j — индексы тех элементов массива, которые нужно сравнить ($0 \leqslant i, j < N$).

В ответ вам будет дана строка, содержащая символ «меньше» («<»), если i-ый элемент секретного массива меньше, чем j-ый, и символ «больше» («>»), если j-ый элемент секретного массива меньше, чем i-ый.

Когда ваша программа определит, на какой позиции секретного массива находится его вторая порядковая статистика, она должна вывести строку, в которой после знака «!» через пробел записана её позиция.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной строки. Иначе решение может получить вердикт Timeout.

Внимание: интерактор вправе подменять содержимое секретного массива в ходе работы вашей программы, если при этом все уже выданные ответы остаются верными.

Пример

Для удобства чтения команды в примере разделены пустыми строками.

стандартный поток ввода	стандартный поток вывода
4 < < < > > 	 ? 0 1 ? 0 2 ? 0 3 ? 3 2 ? 2 1 ! 1

Задача 5. Конусное свечение

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

При неудавшейся попытке перехода границы Остап был ограблен. Но отнюдь не все бранзулетки достались проклятой сигуранце — архиерейский крест и портсигары командор придавил каблуком, припрятав в сугробе. С помощью утаенного ордена Золотого руна великий комбинатор надеется подкупить пограничников и найти сокровища. Он наметил маршрут для поисков в снегах — некоторую ломаную на карте, с которой связана координатная плоскость XY. В одной из точек этой ломаной и находятся сокровища. Проблема в том, что поиски происходят в полной темноте, и найти драгоценности не так-то просто.

Все, что остаётся — договориться с начальником охраны, чтобы тот осветил маршрут поиска прожекторами. Каждая точка этой ломаной должна быть освещена не менее, чем K прожекторами. Все прожекторы включаются одновременно и имеют одинаковый угол освещения, который мы можем менять.

Для каждого прожектора заданы его координаты и вектор, вдоль которого направлен прожектор. Осталось немного: найти минимально возможный угол освещения, при котором наши герои смогут найти сокровища. Чем меньше угол, тем меньше придется отстёгивать начальнику. Причём угол этот единый для всех прожекторов.

Более формально — если прожектор находится в точке P с координатами P_x, P_y, P_z, направлен вдоль направления L с координатами L_x, L_y, L_z и имеет угол освещения φ, то точка Q будет освещена тогда и только тогда, когда угол между векторами $\overrightarrow{PQ}$ и $\overrightarrow{L}$ не превосходит φ.

Формат входных данных

В первой строке входного файла через пробел записаны три целых числа N, M и K — количество звеньев ломаной, количество имеющихся в наличии прожекторов и необходимое число прожекторов, соответственно ($1 \leqslant N \leqslant 100, 1 \leqslant M \leqslant 100, 1 \leqslant K \leqslant M$).

В следующих $N + 1$ строках описываются координаты вершин ломаной. В каждой i-й строке записано по два целых числа X_i и Y_i — координаты i-й вершины ломаной. У всех вершин ломаной z-координата равна 0. Ломаная может иметь самопересечения и самокасания.

Далее идут M строк, описывающих прожекторы, по одному в каждой строке. Описание прожектора состоит из шести целых чисел P_x, P_y, P_z, L_x, L_y, L_z — соответственно, координаты прожектора и координаты вектора направления, вдоль которого светит прожектор ($P_z > 0$). Гарантируется, что хотя бы одна из координат L_x, L_y, L_z не равна нулю.

Все вершины ломаной находятся в разных точках. Положения прожекторов могут совпадать. Все координаты по модулю не превосходят 10^3.

Формат выходных данных

В выходной файл требуется вывести одно вещественное число — минимально возможный угол освещения в градусах, при котором каждая точка маршрута будет освещена не менее чем K прожекторами.

Относительная или абсолютная погрешность ответа должна быть не хуже 10^{-6}.

Примеры

input.txt	output.txt
2 5 2 2 3 8 3 6 6 2 3 3 0 0 -1 5 3 3 0 0 -1 8 3 3 0 0 -1 4 7 1 1 0 -1 6 7 2 0 0 -1	45.00000000
1 1 1 2 0 2 2 0 0 1 0 0 1	116.565051177077989

Пояснение к примеру

В первом примере почти все прожекторы направлены вниз и при угле 45 градусов освещают круг, равный их высоте. Для двукратного освещения первого звена ломаной достаточно первых трёх прожекторов. Оставшуюся часть второго звена освещают последние два.

Во втором примере прожектор смотрит вверх, так что нужен угол освещения больше 90 градусов. Главное — покрыть точку (2, 0), для чего нужен угол $90^\circ + \arctan \frac{1}{2}$.

Задача 6. Тотализатор

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Первое, что сделал Остап, попав в Рио-де-Жанейро, — купил белые штаны. Ну а второе, естественно, спустил все вывезенные деньги в местном футбольном тотализаторе.

В первенстве штата проводится n матчей, в каждом из которых возможен один из трёх исходов: выигрыш первой команды, ничья или выигрыш второй команды.

В билете тотализатора приведён список всех матчей, и покупатель должен сразу при покупке для каждого матча отметить, какие исходы он предполагает. Напротив каждого матча можно отметить либо один исход из трёх возможных (так называемый сингл), либо два (дабл), либо даже все три (трипл).

Билеты продаются разного типа. Тип билета определяет его цену, а также два числа i и j. Покупатель должен поставить в билете ровно i даблов, ровно j триплов и ровно $n - i - j$ синглов. На какие именно матчи ставить синглы, даблы или триплы, и на какие именно исходы — выбирает покупатель.

Если покупатель билета угадал результаты всех n матчей, то он получает за билет P тугриков. А если не угадал — остаётся с носом. Считается, что покупатель угадал результат матча, если в его билете напротив этого матча есть отметка на случившемся исходе.

Благодаря знакомству с Велиалом, демоном азартных игр, Остап узнал, что исход каждого матча определяется случайным образом. Остап даже узнал вероятности всех исходов! Кроме того, он узнал, что исходы всех n матчей определяются независимо друг от друга.

У Остапа осталось S тугриков, которые он может потратить на билеты. Будучи постоянным клиентом, он может купить сколько угодно билетов каждого типа, имеющегося в продаже — лишь бы на всё хватило тугриков. Дело за малым — грамотно распределить деньги так, чтобы математическое ожидание выигрыша было максимальным. Учтите, что при подсчёте выигрыша Остап не обращает внимания на затраты при покупке билетов.

Формат входных данных

В первой строке входного файла записано четыре целых числа n, k, S и P, где n — количество матчей, k — количество типов билетов в продаже, S — сколько тугриков у Остапа и P — сколько тугриков дают за выигравший билет ($1 \leqslant n, k \leqslant 100$, $1 \leqslant S \leqslant 10^6$, $1 \leqslant P \leqslant 10^{18}$).

В следующих n строках записаны вероятности исходов матчей. Каждая i-ая из этих строк содержит три вещественных числа x_i, y_i и z_i с не более чем восемью знаками после десятичной точки — вероятности выигрыша первой команды, ничьей и выигрыша второй команды соответственно в i-ом матче ($0 \leqslant x_i, y_i, z_i \leqslant 1$, $x_i + y_i + z_i = 1$).

Далее идут k строк, в которых задаются типы билетов, имеющиеся в продаже. В каждой строке записано три целых числа: i и j — сколько даблов и триплов соответственно должно быть отмечено в билете этого типа, и c_{ij} — стоимость одного билета этого типа ($0 \leqslant i, j \leqslant n$, $i + j \leqslant n$, $1 \leqslant c_{ij} \leqslant 10^6$).

Формат выходных данных

В выходной файл необходимо вывести одно вещественное число — максимальное значение математического ожидания выигрыша.

Абсолютная или относительная погрешность ответа не должна превышать 10^{-9}.

Примеры

input.txt	output.txt
2 2 10 10 0.5 0.3 0.2 0.8 0.2 0 0 0 4 1 0 6	10.4
3 4 1000 10000 1.0 0 0 0.5 0.5 0 0.3 0.4 0.3 0 0 100 1 0 200 1 1 300 1 2 400	32000

Пояснение к примеру

В первом примере в билете первого типа надо ставить только синглы. Если отметить выигрыш первой команды в обоих матчах, то получится вероятность выигрыша билета 40%. Значит, математическое ожидание выигрыша при покупке такого билета равно 4.

В билетах второго типа нужно указать один дабл. Очевидно, выгоднее всего отметить дабл в первом матче на выигрыш первой команды и ничью, а во втором матче поставить сингл на выигрыш первой. Тогда билет выиграет с вероятностью 64%, что с учётом размера выигрыша 10 даёт в среднем 6.4 тугрика выигрыша.

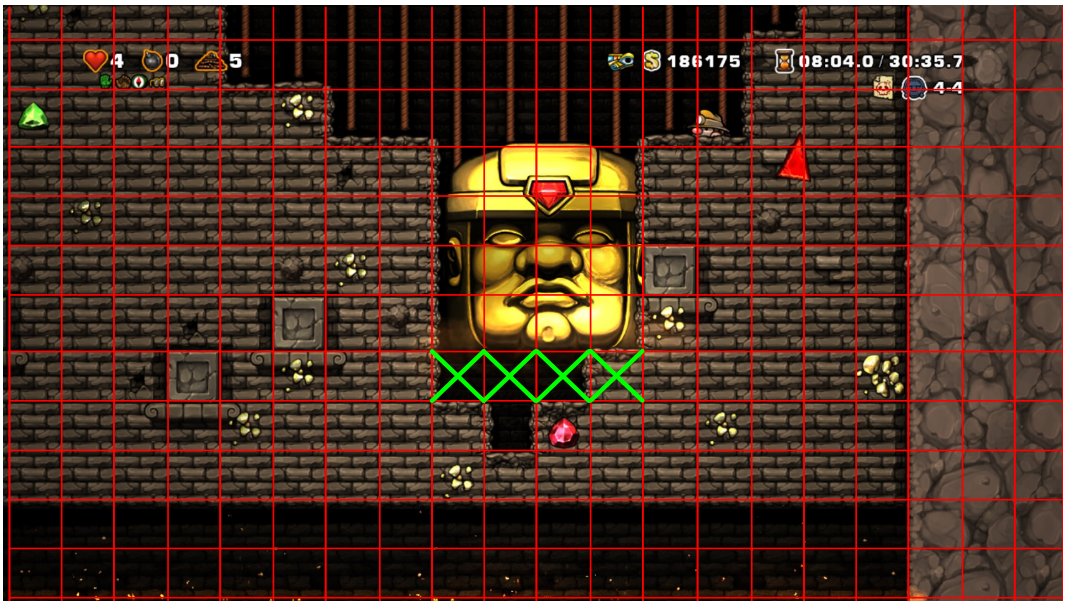
У Остапа всего 10 тугриков, и лучшее, что он может — это купить по одному билету каждого типа.

Задача 7. Olmec

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Olmec (Ольмек) — босс компьютерной игры Spelunky, которого нужно победить на последнем уровне игры. Он перемещается по игровому полю вслед за игроком, каждый раз поднимаясь высоко вверх и падая вертикально вниз. Когда игрок оказывается под Ольмеком, тот падает вниз с особым усилием, стремясь раздавить игрока. При таком сильном ударе он разрушает почву, на которую приземляется. Единственный способ одолеть Ольмека — это заставить его продолбить яму до самого низа уровня, где расположена лава, и позволить ему туда упасть.

Игровой мир Spelunky — это прямоугольное клеточное поле, которое представляет собой вид сбоку. Каждая клетка поля либо заполнена почвой, либо пуста. Ольмек является квадратом размера $K \times K$ клеток, хотя его высота для нас значения не имеет.



Будем считать, что удар Ольмека работает следующим образом. Находясь где-то высоко над землёй, где все клетки пустые, Ольмек выбирает вертикаль, по которой он будет падать — тем самым он определяет свою позицию по горизонтали. Далее он падает вертикально вниз до первого столкновения с клеткой почвы. В момент столкновения K клеток под Ольмеком, которые непосредственно граничат с ним по стороне, подвергаются действию удара — все они становятся пустыми. Далее Ольмек поднимается обратно на большую высоту и готовится к следующему удару.

На картинке изображён момент удара Ольмека размера $K = 4$. Подверженные удару клетки перечёркнуты крестом. Все они станут пустыми в результате удара, хотя лишь в одной из них есть почва перед ударом.

Вам дано состояние игрового поля и набор прямоугольников-запросов. Будем полагать, что мы полностью управляем действиями Ольмека, и можем сделать так, чтобы он выполнил любую наперёд заданную последовательность ударов. Для каждого запроса нужно определить, какое минимальное количество ударов необходимо выполнить, чтобы все клетки заданного прямоугольника стали пустыми.

Формат входных данных

В первой строке входного файла заданы четыре целых числа H, W, K и Q, где H — высота игрового поля, W — ширина игрового поля, K — размер Ольмека и Q — количество запросов ($1 \leqslant H \leqslant 12$, $1 \leqslant K \leqslant W \leqslant 10^5$, $1 \leqslant Q \leqslant 10^5$).

Далее записано игровое поле в H строках, по W символов в каждой. Буква `'X'` (ASCII 88) обозначает клетку с почвой, а символ `'.'` (ASCII 46) обозначает пустую клетку. Строки игрового поля перечислены в порядке сверху вниз.

В оставшихся Q строках заданы запросы, по одному в строке. В каждой строке записано три целых числа: D — глубина прямоугольника, L — номер первого столбца в прямоугольнике и R — номер последнего столбца прямоугольника ($1 \leqslant D \leqslant H$, $1 \leqslant L \leqslant R \leqslant W$). Клетка поля, находящаяся в строке r и в столбце c, входит в заданный прямоугольник, если $r \leqslant D$ и $L \leqslant c \leqslant R$. Гарантируется, что ширина прямоугольника не меньше ширины Ольмека, то есть $R - L + 1 \geqslant K$.

Следует считать, что запросы носят теоретический характер, никакие действия над полем на самом деле не выполняются, и каждый запрос анализируется независимо от остальных.

Формат выходных данных

В выходной файл требуется вывести Q целых чисел, по одному в строке. Каждое число — это минимальное количество ударов, которое требуется совершить, чтобы полностью опустошить соответствующий прямоугольник. Ответы для запросов следует выводить в порядке их перечисления во входном файле.

Пример

input.txt	output.txt
12 20 4 5 XXXX..........XXXXX XXXX..........XXXXXX XXXXXX........XXXXXX XXXXXXXX....XXXXXXXX XXXXXXXX....XXXXXXXX XXXXXXXX....XXXXXXXX XXXXXXXX....XXXXXXXX XXXXXXXX...XXXXXXXXX XXXXXXXXX.XXXXXXXXXX XXXXXXXXXXXXXXXXXXXX ....XXX ....XXX 12 9 12 3 6 11 1 9 13 5 4 10 10 1 20	3 1 0 7 41

Пояснение к примеру

В примере записано игровое поле, которое изображено на картинке в условии.

Задача 8. Игра в строки

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Алиса, Борис и Константин играют в строки. Правила игры таковы:

1. Алиса загадывает строку A.
2. Борис загадывает строку B.
3. Константин, который выступает в роли ведущего, придумывает натуральное число k.
4. В строке A случайным образом выбирается подстрока X длины k. Позиция начала подстроки выбирается равновероятно среди всех возможных вариантов.
5. Аналогичным образом в строке B выбирается случайная подстрока Y длины k.
6. Исход игры определяется в зависимости от того, какая из подстрок лексикографически меньше. Если подстрока X оказалось лексикографически меньше подстроки Y, то побеждает Алиса. Если подстрока Y лексикографически меньше подстроки X, то побеждает Борис. Если подстроки равны, то побеждает дружба.

Алиса и Борис уже загадали свои строки A и B. Константину стало интересно: с какой вероятностью получится каждый исход игры? Требуется вычислить эти вероятности для всех разумных значений числа k.

Формат входных данных

В первой строке входного файла записана строка A, а во второй — строка B. Строка A состоит из n символов, а строка B состоит из m символов ($1 \leqslant n, m \leqslant 2 \cdot 10^5$). В строках могут быть только маленькие латинские буквы.

Формат выходных данных

В выходной файл требуется вывести $\min(n, m)$ строк, в каждой строке по три вещественных числа. Результаты для случая, когда Константин загадывает число k, должны находиться в k-ой строке. Первое число в строке определяет вероятность того, что выиграет Алиса, второе — вероятность того, что победит дружба, и третье — вероятность победы Бориса.

Отклонение каждого выведенного числа от верного значения не должно превышать 10^{-12}.

Пример

input.txt
abac ababa
output.txt
0.2 0.4 0.4 0.33333333333333 0.333333333334 0.333333333333333 0.1666666666666 0.3333333333333 0.500000 0.5 0 0.5

Задача 9. Перфекционизм Стива

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Стив — известный перфекционист. В последнее время на глаза Стиву частенько начали попадаться массивы натуральных чисел, но далеко не всегда они ему нравятся. Массив нравится Стиву тогда и только тогда, когда он строго возрастает. Массивы чисел, которые не нравятся Стиву нужно срочно поменять, иначе он очень сильно расстроится. Новый массив устроит Стива только если он останется той же длины, а его элементы будут кратны соответствующим элементам старого массива. Нужно заметить, что массив может с самого начала устроить Стива, в таком случае менять его не обязательно (но если очень хочется, то можно).

Порадуйте Стива, напишите программу, которая позволит быстро получить из массива новый, который ему понравится.

Формат входных данных

В первой строке входного файла записано одно целое число n — количество элементов массива, попавшегося на глаза Стиву ($2 \leqslant n \leqslant 1000$).

Вторая строка содержит n целых положительных чисел a_i, разделенных пробелами — элементы массива ($1 \leqslant a_i \leqslant 10^6$).

Формат выходных данных

В выходной файл необходимо вывести n целых положительных чисел a_i— новый массив, который понравится Стиву ($1 \leqslant a_i \leqslant 10^9$) .

Вариантов ответа может быть несколько, разрешается вывести любой подходящий ответ.

Пример

input.txt	output.txt
7 1 10 5 4 3 2 7	1 10 15 20 30 32 35

Задача 10. Деревянный трубопровод

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

В тридевятом царстве, тридесятом государстве, правил премудрый царь N городами. Решил однажды царь улучшить жизнь в своей земле, и призвал всех бояр заниматься инновацией, модернизацией и нанотехнологиями. Бояре думали-думали, что бы им сотворить такого инновационного, и построили трубопровод. Нанотрубки мужики пока делают «не очень», поэтому трубопровод сделали из дерева.

Деревянный трубопровод соединяет все города в единую сеть, и от каждого города к каждому можно дойти вдоль него. Состоит он из $N - 1$ магистралей. Каждая магистраль ведёт напрямую из одного города в другой, без каких-либо ответвлений. Для надёжности в смету закладывали дополнительные магистрали, но ближе в концу строительства не хватило дерева.

Известна пропускная способность каждой магистрали в каждую сторону, то есть какое максимальное количество жидкости можно пропускать в единицу времени. Пропускная способность магистрали в противоположных направлениях может отличаться, ибо славится тридесятое царство своим мастерством!

В глубокой печали взирает царь на творение своих подданых. Что качать через трубопровод — он не знает. Молоко будет портиться, а мёду и нету столько. С другой стороны, можно воду перекачивать, если где засуха приключится. Пожелал царь узнать, насколько эффективным будет трувопровод в случае засухи.

Допустим, в городе U случится засуха. Тогда все остальные города делятся на два типа: конечные города и промежуточные. Город называется промежуточным, если из него есть магистраль, ведущая в более удалённый от U город (расстояние считается вдоль магистралей). Все остальные города считаются конечными. Разрешается забирать воду из всех конечных городов в любом объёме, и перекачивать её по магистралям в город U. Из промежуточных городов воду брать нельзя.

Требуется определить максимальный объём воды в единицу времени, который можно перекачать из конечных городов в город U по трубопроводу. Засуха может случиться в любом городе, поэтому нужно посчитать ответ отдельно для каждого города U.

Формат входных данных

В первой строке входного файла записано одно целое число N — количество городов ($1 \leqslant N \leqslant 3 \cdot 10^5$).

В остальных $N - 1$ строках описываются магистрали, по одной в строке. Каждая магистраль описывается четырьмя целыми числами: a — номер города, в котором магистраль начинается, b — номер города, в которым магистраль заканчивается, C_{ab} — пропускная способность магистрали из города a в город b, C_{ba} — пропускная способность магистрали из города b в город a ($1 \leqslant a \neq b \leqslant N$, $1 \leqslant C_{ab}, C_{ba} \leqslant 10^5$).

Гарантируется, что вдоль трубопровода можно добраться из любого города в любой.

Формат выходных данных

В выходной файл требуется вывести N целых чисел, по одному в строке. Каждое k-ое число определяет максимальный объём воды в единицу времени, который можно доставлять в город с номером k, если там приключится засуха.

Пример

input.txt	output.txt
5 1 2 2 4 5 2 2 6 2 3 2 3 3 4 5 5	4 7 7 2 5

Задача 11. Кубические полиномы

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Два старых друга Карданьё и Феррариньё любят устраивать математические состязания: один из них присылает другому по электронной почте математическую задачу, которую надо решить за неделю. Если корреспондент не справляется за отведённое время, то он считается проигравшим.

Феррариньё придумал очередную проблему и уже выслал её другу. Карданьё только что прочитал новый вызов — ему прислали кучу кубических многочленов, то есть выражений вида $ax^3 + bx^2 + cx + d$. Вызов заключается в том, чтобы перемножить все эти многочлены, и найти целый корень наибольшей кратности полученного произведения.

Карданьё очень обеспокоен, потому что он проиграл уже три раза подряд. В этот раз проигрывать уже точно нельзя, поэтому он просит Вас написать программу, которая найдёт ответ на поставленный вызов, чтобы он смог раньше отведённого срока огорчить своего друга правильным ответом.

Формат входных данных

В первой строке входного файла записано одно целое число n — количество уравнений ($1 \leqslant n \leqslant 10^5$).

Далее следуют n строк, каждая i-ая из которых содержит четыре целых числа a_i, b_i, c_i и d_i — коэффициенты i-го уравнения $a_i x^3 + b_i x^2 + c_i x + d_i = 0$ ($0 \leqslant |a_i|, |b_i|, |c_i|, |d_i| \leqslant 10^9$). Гарантируется, что все a_i ненулевые.

Формат выходных данных

Если ни у одного из уравнений нет целого корня, то выведите единственную строку `NO`. Иначе выведите в первую строку входного файла строку `YES`. Во вторую строку выведите два целых числа, разделённых пробелом. Первое число — найденный корень, второе число — кратность этого корня (она должна быть максимально возможной).

Если ответов несколько, то выведите любой из них.

Примеры

input.txt	output.txt
2 1 3 3 1 2 2 0 0	YES -1 4
1 1 1 1 2	NO

Пояснение к примеру

В первом примере при перемножении получается:

$$(x^3 + 3x^2 + 3x + 1) \cdot (2x^3 + 2x^2) = 2x^2(x + 1)^4$$

Число 0 является корнем кратности 2 полученного произведения, а число -1 является корнем кратности 4.

Во втором примере целочисленных корней нет.