Задача 1. Барсик

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

На прямоугольном клеточном поле размера $N \times M$ в одном из углов, в клетке с координатами $(1, 1)$, сидит голодный кот Барсик. А миска с едой для Барсика находится в противоположном углу поля — в клетке с координатами (N, M). Барсик может ходить по полю, перемещаясь между клетками, у которых есть общая сторона.

Однако, на пути к еде у Барсика есть препятствие — злая собака Тузик, который сидит в своей будке в клетке с координатами (R, C). Тузик привязан к будке прочной цепью, и поэтому может добраться только до тех клеток, до которых можно дойти от будки за S перемещений в соседнюю по стороне клетку. Все такие клетки усеяны костями преждевременно ушедших барсиков, и наш Барсик категорически отказывается через них ходить.

Может ли Барсик добраться до своей еды, не проходя через владения Тузика?

Формат входных данных

Первая строка входного файла содержит целое число T — ~~количество Барсиков, которых надо накормить~~ количество тестов в задаче ($1 \leqslant T \leqslant 2\,000$).

Следующие T строк содержат описания тестов, по одному на строке.

Каждый тест состоит из пяти целых чисел N, M, R, C и S записанных через пробел ($1 \leqslant R \leqslant N \leqslant 10^9$, $1 \leqslant C \leqslant M \leqslant 10^9$, $1 \leqslant S \leqslant 10^9$).

Гарантируется, что клетка с будкой Тузика расположена таким образом, что он не может добраться ни до клетки, на которой изначально находится Барсик, ни до клетки с едой для Барсика.

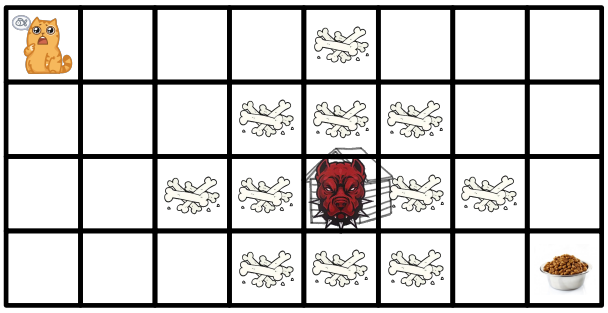
Формат выходных данных

Для каждого теста требуется вывести ответ на него на отдельной строке. Выведите `Barsik`, если Барсик может добраться до еды, не ходя по костям. В противном случае требуется вывести `Tuzik`.

Примеры

input.txt	output.txt
2 8 4 5 3 2 8 4 3 3 1	Tuzik Barsik

Иллюстрация для первого теста из примера:



Задача 2. Кто работает — тот не ест

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

Несмотря на то, что «Согласие — есть продукт при полном непротивлении сторон», монтера Мечникова все-таки припахали таскать театральный реквизит.

В тот момент, когда он проделал p процентов работы, руководство театра поняло, что имеющегося у них довольствия не хватит, чтобы прокормить труженика до окончания работ. Мечников удосужился стрескать q процентов выделенного ему провианта, где q больше p.

Тут-то директор, слегка переиначив известный лозунг «кто не работает — тот ест» пришел к лозунгу «кто работает — тот не ест» (ну или почти не ест). А потому он решил прикомандировать к пролетарию некоторое количество почти дармовой рабочей силы — кодеров, которые работают за еду. Каждый из них работает в a раз быстрее монтера, а ест в b раз медленнее.

Запас дармовых кодеров практически безграничен, но привлекать их слишком много — тоже не резон. Ещё как прибежит их великое полчище, как растрезвонят про дармовой хавчик... Помогите директору, определите минимальное необходимое количество кодеров, достаточное для того, чтобы имеющегося продовольствия хватило до завершения работы.

Формат входных данных

В первой строке входного файла записано одно число t ($1 \leqslant t \leqslant 3 \cdot 10^4$) — количество тестов. В каждой из следующих t строк входного файла записано по четыре целых числа $0 < p, q, a, b < 100$ — соответственно процент уже выполненной работы, процент съеденного продовольствия, во сколько раз кодер работает быстрее монтера и во сколько раз медленнее он ест. Гарантируется, что $q > p$.

Формат выходных данных

В ответ на каждый из t тестов выведите по одному целому числу в отдельной строке — минимальное количество кодеров, которое необходимо привлечь. Если же работу выполнить никак нельзя, необходимо вывести число `-1`.

Примеры

input.txt	output.txt
3 1 2 99 1 30 60 2 2 30 80 1 2	1 10 -1

Задача 3. Бомбы

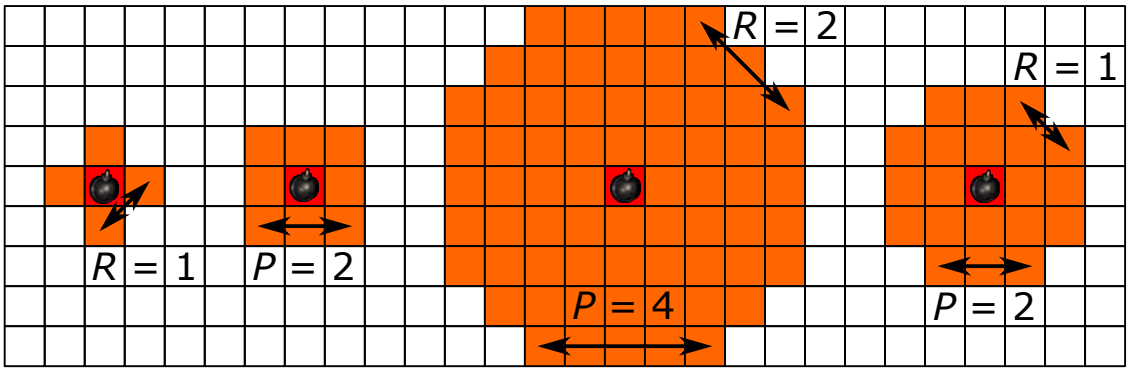
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Геннадий играет в свою любимую игру.

Игровой уровень представляет собой клеточное поле. Каждая клетка поля либо пустая, либо заполнена землёй. В пустой клетке может располагаться вход на уровень или выход из него. Игрок может ходить по пустым клеткам, на каждом шаге переходя из клетки в соседнюю по стороне клетку. Выходить за пределы уровня нельзя.

Некоторые игровые уровни не получается пройти по-простому, и тогда в дело включаются бомбы. Бомбу можно прилепить на землю, и когда она взорвётся, близкие клетки очистятся от земли. В данной задаче будем полагать, что игрок может заложить бомбу в той клетке, где он находится, если она граничит по стороне с клеткой с землёй. После взрыва бомбы все клетки с землёй в области взрыва превращаются в пустые клетки, по которым можно ходить. В область взрыва могут попадать вход на уровень, выход с уровня и сам игрок — им от взрыва ничего не делается.

Область взрыва бомбы имеет восьмиугольную форму с центром в той клетке, где была заложена бомба. У области взрыва есть два параметра P и R, и её точная форма определяется следующим образом. Чтобы обойти её граничные клетки по периметру, нужно P раз сдвинуться вверх, потом R раз сдвинуться вверх и вправо, затем P раз сдвинуться вправо, R раз сдвинуться вправо и вниз, затем P раз сдвинуться вниз, и так далее, до момента возвращения в начальную клетку. Форма области взрыва при некоторых значениях параметров показана на картинке:



Бомбы в игре — конечный и очень ценный ресурс, поэтому тратить их надо по минимуму. Помогите Геннадию определить, как пройти очередной уровень, затратив минимальное количество бомб.

Формат входных данных

В первой строке входного файла записано два целых числа N и M — размер поля по вертикали и по горизонтали соответственно ($1 \leqslant N, M \leqslant 1\,500$). Во второй строке записано два целых числа P и R — параметры области взрыва ($0 \leqslant P, R \leqslant 1\,000$). Гарантируется, что число P чётное, и что $P + R > 0$.

Далее записана карта уровня: N строк по M символов в каждой. Каждый символ описывает содержимое соответствующей клетки:

- `@` — клетка с землёй.
- `.` — пустая клетка.
- `S` — пустая клетка со входом на уровень.
- `E` — пустая клетка с выходом из уровня.

Гарантируется, что на карте одна клетка со входом на уровень и одна клетка с выходом из него.

Формат выходных данных

В первой строке выведите одно целое число B — минимальное достаточное количество бомб ($B \geqslant 0$). В остальных B строках выведите клетки, в которые нужно закладывать бомбы, в порядке их подрыва. В каждой строке выведите два целых числа u и v — номер строки и столбца соответственно ($1 \leqslant u \leqslant N$, $1 \leqslant v \leqslant M$).

Примеры

input.txt	output.txt
9 10 0 1 @@......@@ @@S@@@@... @@@@@...@. @@@@@@@@@@ @@@@@@@@@@ ....@ @@@.@@@@.. @...@@@E@. @@@@@@@@@@	3 3 7 4 7 8 10
4 5 2 1 S@@@@ ..@@@ @@@@E @@@@@	1 2 2
7 5 4 0 @E@@@ @@.@@ @@@@@ @@@@@ @@... @@... @@..S	2 5 5 2 3

Пояснение к примеру

В первом примере взрыв имеет форму креста и затрагивает соседние по стороне клетки. Две бомбы нужно, чтобы пробить стену в строках 4 и 5, и ещё одна бомба, чтобы откопать выход. Во втором примере взрыв достаточно мощный, что пробить проход одной бомбой. В третьем примере необходимо две бомбы. Заметим, что закладывать первую бомбу в клетке (6, 4) нельзя, потому что эта клетка не граничит с землёй.

Задача 4. Починка кучи 8-бит

Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
	3 секунды (для Java)
Ограничение по памяти:	**64** мегабайта

Петя написал кучу для динамического выделения памяти в его собственном языке программирования Ц. Корректная куча устроена следующим образом. Это непрерывный отрезок памяти, состоящий из N ячеек. В каждой ячейке хранится беззнаковое **8-битное** целое число, которое может принимать любое значение **от 0 до 255** включительно. Вся куча разбивается на B подотрезков, называемых блоками. При этом каждая ячейка кучи принадлежит ровно одному блоку.

Каждый блок состоит из двух или более ячеек. В первой и в последней ячейках блока хранится полезный размер блока — это количество ячеек в нём, исключая первую и последнюю ячейки. В остальных ячейках блока могут быть записаны произвольные данные, независимо от того, свободный блок или занятый.

Язык программирования Ц очень низкоуровневый, его пользователи («цэшные» программисты) часто ошибаются и портят кучу, записав случайно свои данные не в ту ячейку. Пользователи просят Петю написать функцию восстановления целостности кучи после такого рода происшествия.

Функция должна проанализировать содержимое N ячеек памяти, в которых должна располагаться куча, и изменить содержимое нескольких ячеек так, чтобы этот отрезок памяти стал представлять собой корректную кучу. Количество изменённых ячеек должно быть минимально возможным.

Формат входных данных

Содержимое анализируемого отрезка памяти из N ячеек записано в бинарном файле, который имеет размер N байт. Каждый байт файла представляет собой одну ячейку памяти. Выполняется $2 \leqslant N \leqslant 2^{24}$, то есть размер файла не меньше двух байт и не больше 16 мегабайт.

Формат выходных данных

Требуется вывести найденное множество из K изменений ячеек в бинарный файл размером $4K$ байт.

Каждое изменение описывается четырьмя байтами. Первые три байта трактуются как беззнаковое 24-битное целое число и определяют адрес (номер) изменённой ячейки. Разумеется, это число должно быть меньше N — размера входного файла. Последний четвёртый байт определяет новое содержимое этой ячейки.

24-битные адреса требуется записывать с порядком байтов little-endian, то есть первый байт младший, а последний — старший. Если есть несколько ответов с одинаковым количеством изменений, разрешается вывести любой из них. Порядок вывода изменений значения не имеет.

Примеры

Для удобства восприятия ниже показано содержимое входного и выходного файлов в шестнадцатеричном виде. В системе тестирования файлы будут в бинарном виде! Примеры в бинарном виде можно скачать на вкладке «Новости» рядом с условиями.

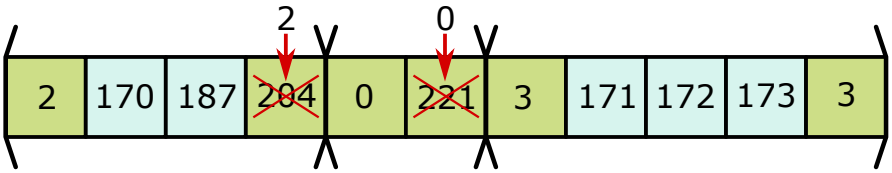
Для быстрого чтения входного файла рекомендуется использовать один вызов: `Files.readAllBytes` на Java и `fread` на C++. Не забудьте, что файлы на C++ нужно открывать в бинарном режиме.

input.bin	output.bin
02 AA BB 02 00 00 03 AB AC AD 03	
02 AA BB CC 00 DD 03 AB AC AD 03	03 00 00 02 05 00 00 00

Пояснение к примеру

В обоих примерах $N = 11$. В первом примере куча уже корректная, и содержит три блока с полезными размерами 2, 0 и 3 соответственно. Раз изменений не требуется, то выходной файл должен быть пустым. Во втором примере куча некорректная, но её можно починить за два изменения, сделав из неё в точности кучу из первого примера. Для этого предлагается изменить ячейку с адресом 3 с `CC` на `02`, и изменить ячейку с адресом 5 с `DD` на `00`.

Иллюстрация для второго примера (числа записаны в десятичном виде):



Задача 5. Починка кучи 32-бит

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Петя написал кучу для динамического выделения памяти в его собственном языке программирования Ц. Корректная куча устроена следующим образом. Это непрерывный отрезок памяти, состоящий из N ячеек. В каждой ячейке хранится беззнаковое **32-битное** целое число, которое может принимать любое значение **от 0 до 4 294 967 295** включительно. Вся куча разбивается на B подотрезков, называемых блоками. При этом каждая ячейка кучи принадлежит ровно одному блоку.

Каждый блок состоит из двух или более ячеек. В первой и в последней ячейках блока хранится полезный размер блока — это количество ячеек в нём, исключая первую и последнюю ячейки. В остальных ячейках блока могут быть записаны произвольные данные, независимо от того, свободный блок или занятый.

Язык программирования Ц очень низкоуровневый, его пользователи («цэшные» программисты) часто ошибаются и портят кучу, записав случайно свои данные не в ту ячейку. Пользователи просят Петю написать функцию восстановления целостности кучи после такого рода происшествия.

Функция должна проанализировать содержимое N ячеек памяти, в которых должна располагаться куча, и изменить содержимое нескольких ячеек так, чтобы этот отрезок памяти стал представлять собой корректную кучу. Количество изменённых ячеек должно быть минимально возможным.

Протокол взаимодействия

Количество ячеек памяти может быть очень большим, и прочитать полностью весь отрезок ваша программа никак не сможет. Поэтому 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 на стандартный поток ввода вашей программе подаётся целое число N — количество ячеек памяти в анализируемом отрезке ($2 \leqslant N \leqslant 2^{32}$). Далее ваша программа может делать запросы, чтобы узнавать содержимое различных ячеек памяти. Чтобы сделать запрос, нужно вывести в стандартный поток вывода слово `read`, а затем через пробел целое число A — адрес/номер ячейки ($0 \leqslant A < N$). В ответ на стандартный поток ввода придёт одно целое число V — содержимое A-ой ячейки памяти ($0 \leqslant V < 2^{32}$).

В конце концов ваша программа должна вместо очередного запроса вывести ответ на поставленную задачу. В первой строке ответа нужно вывести слово `fix`, а затем через пробел целое число K — сколько ячеек потребуется изменить ($K \geqslant 0$). В остальных K строках следует вывести предлагаемые изменения, по одной в строке. Для каждого изменения требуется вывести два целых числа: A — адрес/номер ячейки, которую нужно изменить и V — новое значение, которое следует записать в эту ячейку ($0 \leqslant A < N$, $0 \leqslant V < 2^{32}$).

Гарантируется, что существует оптимальный ответ, в котором получается куча с не более чем 25 000 блоками. Вашей программе разрешается сделать не более 120 000 запросов, иначе решение получит вердикт Wrong Answer.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого выведенного запроса. Иначе решение может получить вердикт Timeout. Все числа записываются в десятеричном виде в текстовом формате.

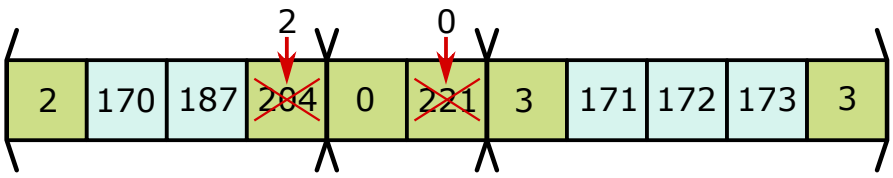
Примеры

Для удобства чтения команды в примере разделены строками с символом минуса. Ваша программа **не** должна выводить никаких минусов.

стандартный поток ввода	стандартный поток вывода
11 - 2 - 170 - 187 - 204 - 0 - 221 - 3 - 171 - 172 - 173 - 3 - - -	- read 0 - read 1 - read 2 - read 3 - read 4 - read 5 - read 6 - read 7 - read 8 - read 9 - read 10 - fix 2 3 2 5 0

Пояснение к примеру

Пример полностью повторяет второй пример из другой задачи «Починка кучи 8-бит»: совпадает как изначальное содержимое ячеек в анализируемом отрезке, так и предлагаемый способ восстановления целостности. Иллюстрация:



Задача 6. Дробь

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

Поначалу Берлага в сумасшедшем доме изображал из себя вице-короля Индии. Поразмыслив, пришел к выводу, что это может быть чревато — посадят на слона и заставят им рулить. А потому он решил переменить свой бред. Правда, пока что не придумал окончательно, кем бы стать. Поэтому он предается своему любимому занятию — раскладывает пасьянс «Косынка».

Время от времени он задается вопросом — какая у него доля выигрышей? Поскольку сам пасьянс выдает ответ только с двумя знаками после запятой, а он любит абсолютно точные цифры, то иногда он считает долю выигрышей самостоятельно. При этом ответ у него может быть как бесконечной периодической дробью, так и конечной — это зависит от самой дроби и от основания системы счисления. Да-да, у бухгалтеров в сумасшедшем доме может быть какая угодно система счисления, а не только десятичная. Так, например, в десятичной системе $1/3$ — бесконечная периодическая дробь, а $4/10$ — конечная, но зато в троичной системе все будет наоборот: $1/3$ будет конечной дробью “0.1”, а $4/10$ — бесконечной дробью “$0.101210121012\ldots$”.

Естественно, он не любит бесконечные дроби (как все бухгалтеры) и предпочитает считать эту самую долю выигрышей только тогда, когда он абсолютно уверен, что независимо от числа выигрышей ответ будет хороший, то есть будет конечной дробью. Для этого иногда ему надо сыграть дополнительно еще некоторое количество игр.

Помогите Берлаге узнать, какое минимальное количество игр ему надо еще сыграть. При этом общее количество игр не должно превышать M.

Формат входных данных

В первой строке входного файла дано три целых числа: B — основание системы счисления, M — максимально допустимое количество игр и N — количество запросов ($2 \leqslant B \leqslant 5 \cdot 10^6$, $1 \leqslant M \leqslant 10^{18}$, $1 \leqslant N \leqslant 10^5$).

Далее следует N строк, описывающих запросы. В каждой строке указано одно целое число a_i — сколько игр уже сыграл Берлага ($1 \leqslant a_i \leqslant M$).

Формат выходных данных

В выходном файле должно быть N строк, в каждой строке ответ на соответствующий запрос. Каждый ответ должен быть целым числом A — сколько игр нужно сыграть дополнительно, чтобы доля выигрышей гарантированно записывалась конечной дробью ($A \geqslant 0$). Если при этом общее количество игр превышает M, то выведите -1 в качестве ответа.

Примеры

input.txt	output.txt
100 120 3 5 117 13	0 -1 3

Задача 7. Придорожная оптимизация

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

В губернии G некоторые города соединены дорогами, дороги всегда идут в двух направлениях. Бюджет на поддержание дорожной сети почти не выделяется, поэтому принято решение оставить минимальное количество дорог. Однако, если из города A можно доехать до города B, то и после сокращения количества дорог такая возможность должна остаться.

Определите минимальное число дорог, которое необходимо оставить.

Формат входных данных

В первой строке входного файла записано единственное целое число T — количество тестов ($1 \leqslant T \leqslant 50\,000$).

Далее следует описание T тестов.

В первой строке теста под номером t записано единственное целое число N_t — количество городов ($1 \leqslant N_t \leqslant 200$).

Далее в N_t строках записано N_t целых чисел, каждое из которых равно 0 либо 1. Если в строке с номером i на j-м месте находится 0, значит проезда между городами с номерами i и j нет, даже с использованием других городов в качестве промежуточных, если 1 — проехать можно. Считается, что из города в него же проехать можно, поэтому в строке i в позиции i всегда будет находиться 1 ($1 \leqslant i \leqslant N_t$).

Гарантируется, что сумма N_t^2 по всем тестам не превышает 50 000.

Формат выходных данных

Для каждого теста в отдельной строке выведите единственное целое число — минимальное количество дорог, которое необходимо оставить.

Примеры

input.txt	output.txt
1 1 1	0
2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1	3 4

Задача 8. Деревянный трубопровод

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

В тридевятом царстве, в тридесятом государстве правил когда-то премудрый царь. Решил 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 делали «не очень», поэтому трубопровод сделали из дерева.

Деревянный трубопровод соединяет все N городов 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. Известна пропускная способность каждой магистрали, то есть какое максимальное количество жидкости можно пропускать через него в единицу времени.

Прошли годы, правление мудрого царя подошло к концу. Его сын перевёл государство на капиталистические рельсы. Внуку досталось современное государство с пустой казной и бесконечными долгами. Трубопровод всё это время стоял без дела. И ничего ему не было, ибо на века строили! Ну разве что его приватизировали по частям, так — на всякий случай. Наконец, в городе U приключилась засуха, и внук вспомнил о плане деда: теперь через трубопровод будут качать воду в город U.

Все города, кроме города U, 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.

К сожалению, есть недобрые люди, которые решили нажиться на бедствии и стали взимать плату за прокачивание воды по своим магистралям. Прижучить их царь не может, потому что частная собственность, свободный рынок и так далее. В противодействие этим стервятникам, верные монархическим идеалам люди заявили, что они сочтут за честь, если по их магистралям будут качать воду в город U, и даже будут платить за это деньги. Таким образом, для каждой магистрали известно, сколько нужно платить за каждую единицу воды, пропущенную по ней, при этом это число может быть и отрицательным.

Денег у царя нет, в казне тоже нет. Помогите ему составить такой план, чтобы в город U приходило как можно больше воды, но чтобы для этого не требовалось никаких денежный вливаний извне.

Формат входных данных

В первой строке входного файла записано одно целое число N — количество городов ($2 \leqslant N \leqslant 200\,000$).

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, M_{ab} — пропускная способность магистрали, C_{ab} — стоимость прокачивания единицы воды по магистрали ($1 \leqslant a \neq b \leqslant N$, $1 \leqslant M_{ab} \leqslant 10^6$, $|C_{ab}| \leqslant 10^7$).

Гарантируется, что вдоль трубопровода можно добраться из любого города в любой. Город U имеет номер 1.

Формат выходных данных

Выведите одно вещественное число F — максимальный объём воды, который можно поставлять в город U в единицу времени. Абсолютная или относительная погрешность ответа не должна превышать 10^{-12}.

Примеры

input.txt	output.txt
2 1 2 10 -15	10
6 1 3 5 -4 1 2 14 2 4 2 6 -1 5 2 3 5 6 2 6 1	15.666666666666666667

Пояснение к примеру

В первом примере одна магистраль с отрицательной стоимостью. Можно просто пустить по ней максимум воды, ещё и денег на этом заработать.

Во втором примере можно получить оптимальный ответ следующим образом. По магистрали из 1 в 3 пропустим 5 единиц воды, получая с этого 20 монет прибыли. По магистрали из 1 в 2 пропустим $10\frac{2}{3}$ единиц воды, затратив на это $21\frac{1}{3}$ монет. Их них 6 единиц пропустим из 2 в 4, получив 6 монет прибыли, а остальные $4\frac{2}{3}$ единиц воды пропустим из 2 в 6, затратив на это $4\frac{2}{3}$ монет.

Задача 9. Плащ и ружьё

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

Геннадий придумал новый челлендж в его любимой игре: проходить уровни, не касаясь земли.

Игровой уровень представляет собой клеточное поле. Каждая клетка поля либо пустая, либо заполнена землёй. В пустой клетке может стоять монстр, а также может располагаться вход на уровень или выход из него. Все клетки за пределами поля заполнены землёй. Монстры не перемещаются.

Несмотря на то, что уровень представляется клеточным полем, время в игре течёт непрерывно, и игрок перемещается по уровню непрерывно. Будем считать, что игрок является точкой. Благодаря имеющемуся у него плащу, игрок постоянно опускается вниз с маленькой скоростью по вертикали. Плащ позволяет игроку парить: в любой момент времени он может выбирать горизонтальную компоненту скорости по собственному желанию, в дополнение к вертикальной компоненте скорости, которую контролировать нельзя. Игрок не может проходить через клетки с землёй, и в рамках челленджа не может даже касаться земли.

Из ружья можно стрелять только влево и вправо. Пули вылетают из ружья и летят горизонтально в выбранном направлении до первого контакта с землёй. Во всех клетках, через которые пролетают пули, монстры умирают.

Чтобы пройти уровень, нужно пролететь из клетки, в которой расположен вход на уровень, в клетку, в которой расположен выход. Начинать можно в любой точке клетки со входом, и заканчивать можно в любой точке клетки с выходом. Геннадий хочет узнать, сможет ли он это сделать, и если сможет, то какое максимальное количество монстров он сможет при этом убить.

Игроку разрешается сколь угодно быстро перемещаться по горизонтали. Патроны у ружья не заканчиваются, скорострельность не ограничена. Игрок может пролетать через клетку с монстром, и монстр при этом умирает.

Формат входных данных

В первой строке входного файла записано два целых числа N и M — размер поля по вертикали и по горизонтали соответственно ($1 \leqslant N, M \leqslant 2\,000$).

Далее записана карта уровня: N строк по M символов в каждой. Каждый символ описывает содержимое соответствующей клетки:

- `@` — клетка с землёй.
- `.` — пустая клетка.
- `m` — пустая клетка с монстром.
- `S` — пустая клетка со входом на уровень.
- `E` — пустая клетка с выходом из уровня.

Гарантируется, что на карте одна клетка со входом на уровень и одна клетка с выходом из него.

Формат выходных данных

Если пролететь уровень нельзя, выведите число -1, а если можно, то выведите максимальное количество убитых при этом монстров.

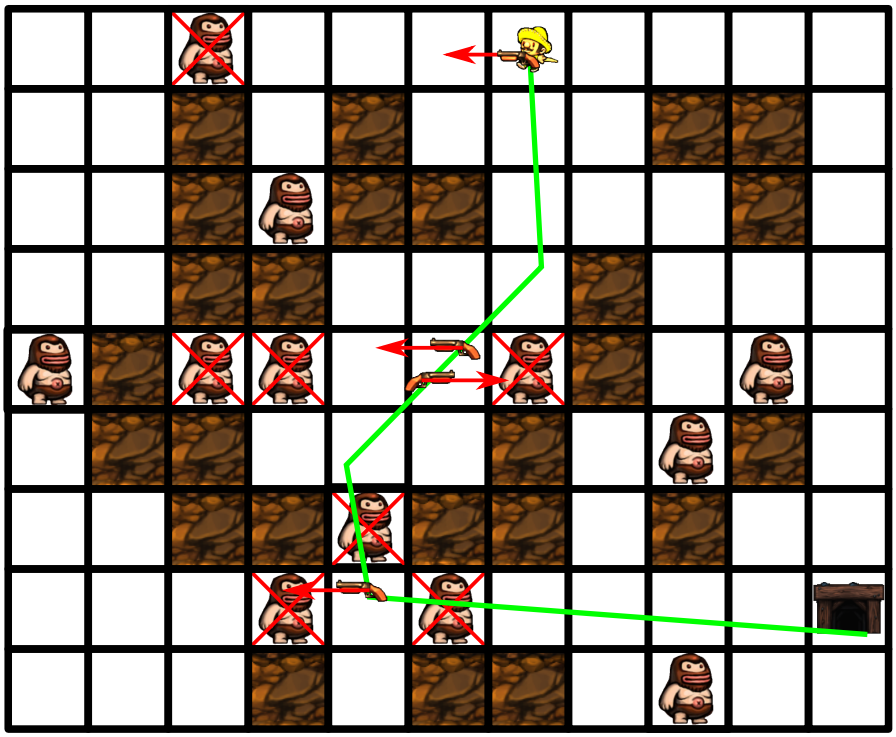
Примеры

input.txt	output.txt
9 11 ..m...S.... ..@.@...@@. ..@m@@...@. ..@@...@... m@mm..m@.m. .@@...@.m@. ..@@m@@.@.. ...m.m....E ...@.@@.m..	7
3 1 S @ E	-1

Пояснение к примеру

Оптимальный план прохождения уровня из первого примера приведён ниже. Заметим, что пролетать по диагонали между двумя клетками с землёй нельзя.

Во втором примере путь между входом и выходом заблокирован, так что пролететь уровень нельзя.



Задача 10. Остап и стулья

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

В новой компьютерной игре «Остап и стулья» есть один весьма волнующий момент, когда много маленьких Остапчиков бегут каждый к своему стулу. При этом графика этого события уже отрисована: имеются две картинки — на первой нарисованы Остапы (их координаты x_i заданы), а на второй стулья (их координаты y_i также известны).

Перед стартом игры нельзя перемещать ни стулья, ни Остапчиков, однако можно изменить масштаб второй картинки со стульями с помощью линейного преобразования $y_i \to k * y_i + b$. После этого сначала первый Остап бежит к первому стулу, потом второй ко второму и так далее, а их время суммируется. Игроку необходимо сделать это суммарное время как можно меньше, то есть минимизировать общее суммарное расстояние.

Ваша задача — найти минимально возможное значение следующего выражения:

$$\sum_{i=1}^{N} |x_i - (ky_i + b)|$$

Формат входных данных

Во входном файле в первой строке записано целое число N — количество Остапов и стульев ($2 \leqslant N \leqslant 300$). Далее в двух строках находится по N целых чисел: во второй строке файла x_i — координаты Остапов, в третьей строке y_i — координаты стульев ($1 \leqslant i \leqslant N$, $|x_i|, |y_i| \leqslant 10^3$). Все x_i различные, все y_i различные.

Формат выходных данных

В ответе необходимо вывести три вещественных числа: D — минимально возможное значение общего суммарного расстояния и коэффициенты K и B, при которых это расстояние достигается.

Относительное или абсолютное отклонение расстояния D от оптимального не должно превышать 10^{-9}. Вычисленное по коэффициентам K и B расстояние должно совпадать с D с той же точностью.

Примеры

input.txt	output.txt
3 0 3 -5 4 1 -2	5.5 0.8333333333 -3.3333333333
2 -7 12 -7 12	0 1 0