

Для всех задач:

Имя входного файла: *input.txt*
 Имя выходного файла: *output.txt*
 Ограничение по памяти: *256 Мб*

Задача 1. Измерение Вселенной

Ограничение по времени на 1 тест: *1 сек.*

Остап Ибрагимович и Ипполит Матвеевич обзавелись каждый по астролябии и решили чего-нибудь такое померить. А чего и не мерить-то — сама меряет, было бы что мерить. Ну и направили они каждый свою астролябию на северный полюс Вселенной, да и меряют все, что ни попадя. Ну, а прибор сей устроен так, что меряет он все подряд, но только строго внутри области, точки которой удалены от наблюдателя не более, чем на R парсек, и луч, на них направленный из точки наблюдения, отклоняется от луча, направленного строго вверх параллельно оси OZ не более, чем на α градусов. И, конечно же, им интересно узнать объём области, каждая точка которой обзревается хотя бы одной из астролябий.


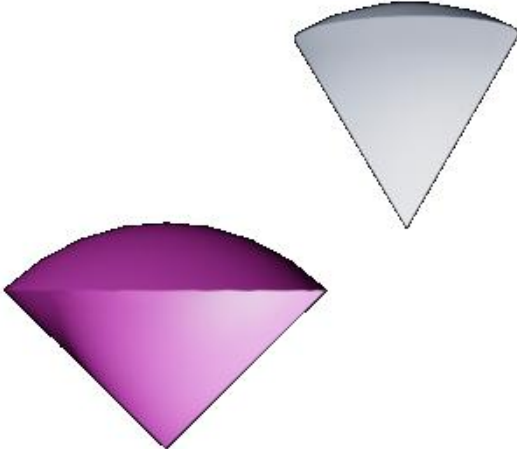
Входные данные

Во входном файле содержатся две строки. В каждой строке записано по пять вещественных чисел x , y , z , α , R — координаты наблюдателей, угол, внутри которого меряет астролябия, и радиус области наблюдений, соответственно ($0 \leq x, y, z, R \leq 30$, $0 \leq \alpha \leq 90$). При этом x , y , z и R измеряются в парсеках, а угол α — в градусах.

Выходные данные

В выходной файл необходимо вывести одно вещественное число — объём измеренной области в кубических парсеках, с точностью до 0.1 кубопарсека.

Примеры

<i>input.txt</i>	<i>output.txt</i>	<i>Схематическое изображение измеренной области</i>
0.0 0.0 0.0 90.0 10.0 0.0 0.0 5.0 45.0 5.0	2094.4	
5.1 5.4 5.6 45.5 15.7 20.8 20.1 20.3 30.3 13.1	3067.3	

Задача 2. Цапля на болотах

Ограничение по времени на 1 тест: **2 сек.**

для задач на Java: **3 сек.**

В некоторой болотистой местности поселилась цапля. Она любит облетать близлежащие болота и путешествовать по кочкам. Прилетев на какое-то болото, цапля выбирает себе кочку, встает на нее левой ногой, размышляет о жизни, затем выбирает себе другую кочку, до которой шагает по промежуточным кочкам, начиная с правой ноги, вставая на каждую кочку одной ногой. Она выбирает себе маршрут таким образом, чтобы на конечную кочку попасть снова левой ногой, причем не проходить одну кочку дважды.

Цапля попалась довольно ленивая. Она хотела бы сделать как можно меньше шагов по кочкам. Помогите ей.

Входные данные

Входной файл содержит описание нескольких болот.

В первой строке описания одного болота через пробел даны числа n , m , s и t , где n — количество кочек на этом болоте, m — количество пар кочек, между которыми цапля может сделать шаг, s — номер начальной кочки, t — номер конечной кочки. Все кочки на болоте занумерованы числами от 1 до n , $s \neq t$.

В следующих m строках записано по два различных целых числа — номера кочек, между которыми цапля может сделать один шаг.

Сумма n по всем болотам не превосходит 20. В каждом болоте $n \geq 2$.

Входной файл заканчивается строкой **0 0 0 0**.

Выходные данные

Для каждого болота на отдельной строке нужно вывести в выходной файл через пробел последовательность номеров кочек, по которым цапля может пройти от начальной кочки до конечной, сделав минимальное количество шагов, причем начинать движение с правой ноги и попасть на последнюю кочку левой ногой. Если таких путей несколько, то вывести любой. Если же такого пути между заданными кочками не существует, то нужно вывести фразу **No path**.

Пример

<i>input.txt</i>	<i>output.txt</i>
3 3 1 2	1 3 2
3 1	No path
3 2	
2 1	
4 4 1 2	
1 2	
1 3	
3 4	
4 1	
0 0 0 0	

Задача 3. Труба

Ограничение по времени на 1 тест: **1 сек.**

Труба́ (итал. tromba, фр. trompette, нем. Trompete, англ. trumpet) — медный духовой музыкальный инструмент альтово-сопранового регистра, самый высокий по звучанию среди медных духовых. В качестве сигнального инструмента натуральная труба использовалась с древнейших времён, примерно с XVII века вошла в состав оркестра. С изобретением механизма вентилей труба получила полный хроматический звукоряд и с середины XIX века стала полноценным инструментом классической музыки. Инструмент обладает ярким, блестящим тембром, используется как сольный инструмент, в симфоническом и духовом оркестрах, а также в джазе и других жанрах.

Из Википедии — свободной энциклопедии

Рассмотрим для начала строение натуральной трубы. По сути своей натуральная труба представляет длинную трубку, которая изгибается исключительно для компактности. Она слегка сужается у мундштука, расширяется у раструба, а на остальных участках имеет цилиндрическую форму. В рамках данной задачи будем считать для простоты, что натуральная труба имеет форму цилиндра длины L безо всяких сгибов, сужений и расширений.

Звучащим телом трубы является столб воздуха, заполняющего канал ее трубки и отделенного от окружающей среды корпусом инструмента. Этот столб воздуха служит резонатором, отвечающим на звуковые колебания определенных частот в зависимости от его размера.

Посредством специального механизма возбуждения колебаний в основной части трубы генерируются стоячие волны, пучности которых находятся у двух концов трубы; в случае извлечения основного тона (первой гармоники) один узел стоячей волны находится посередине (а).

Узкая и длинная форма столба воздуха обеспечивает легкость так называемого «передувания», то есть разделения столба воздуха на ряд участков, колеблющихся в противоположных фазах; при этом образуются новые узлы и пучности, причем на краях всегда остаются пучности. Такое явление можно наблюдать на картинках (б), (в) и (г), на которых показано извлечение второй, третьей и четвертой гармоники соответственно.

Таким образом, на натуральной трубе можно извлечь лишь звуки, длина волны которых нацело делит удвоенную длину трубки. Набор всех таких звуков по аналогии с математикой называется гармоническим звукорядом или натуральным звукорядом. k -ый звук ряда называется k -ой гармоникой и имеет длину волны:

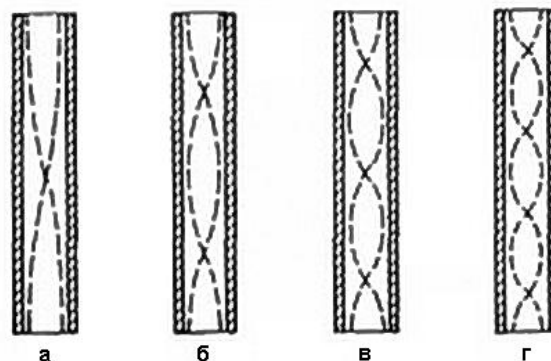
$$\lambda_k = \frac{2L}{k}$$

В виду технических особенностей на трубе основной тон (первая гармоника) не извлекается. Начиная со второй все остальные гармоники прекрасно звучат по описанной выше теории. Таким образом, извлечь можно звуки с длинами волн λ_k при $k \geq 2$.

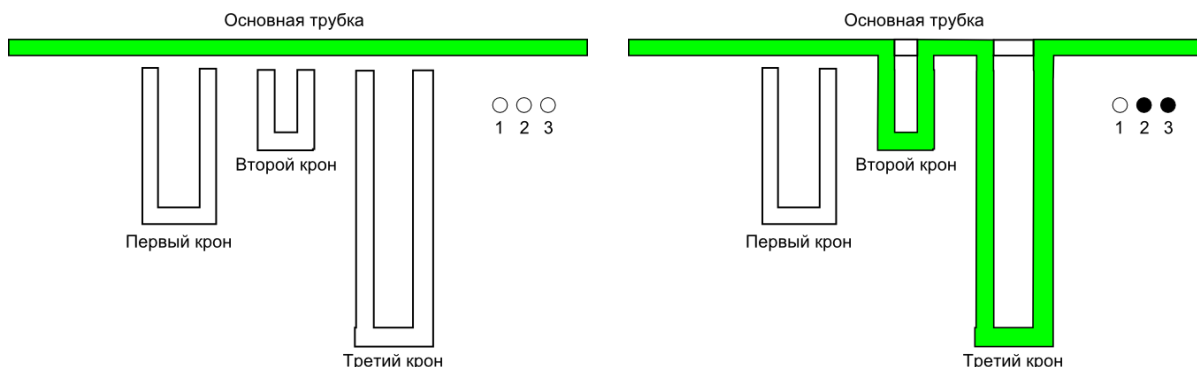
Неспособность натуральной трубы извлекать звуки, не лежащие в натуральном звукоряде, сильно ограничивает её репертуар. Например, каждая натуральная труба может играть произведения только в одной тональности, которая соответствует её основному тону. К тому же большая часть современной музыки написана с использованием хроматического звукоряда, который гораздо полнее натурального.

Для расширения возможностей трубы до полного хроматического ряда была изобретена система вентилей. Данная система позволяет увеличить длину столба воздуха трубы, понижая таким образом натуральный звукоряд.

Кроме основной трубки на вентильной трубе



также имеется N кронов. Крон — дополнительная трубка некоторой длины. Каждый крон соединён с основной трубкой через индивидуальный вентиль. При нажатии вентиля соответствующий крон мгновенно подключается к основной трубке, благодаря чему длина колеблющегося столба воздуха увеличивается на длину подключенного крона. Вентили можно нажимать независимо друг от друга. При нажатии нескольких вентилях длина столба воздуха увеличивается на сумму длин всех кронов, им соответствующих.



На рисунке зеленым цветом показан колеблющийся столб воздуха. Слева не нажат ни один вентиль, справа нажаты второй и третий вентиля.

Вентильный механизм современной трубы позволяет с достаточно малой погрешностью извлекать все звуки хроматической гаммы, используемые в европейской музыке.

Учитывая то, что скорость звука в воздухе постоянна и равна $c = 340$ метров в секунду, можно однозначно определить частоту колебаний по длине волны и наоборот. Напоминаем, что произведение длины волны на частоту колебаний равно скорости звука:

$$F\lambda = c$$

В период классической музыки основным клавишным инструментом стало фортепиано. Как известно, на фортепиано последовательность всех клавиш слева направо образует хроматический ряд. Теоретически идеально настроенное фортепиано должно соответствовать равномерно темперированному строю. В таком строе интервал между любыми двумя соседними звуками хроматического ряда одинаков и равен одному полутону. Если два звука имеют частоты F_1 и F_2 , то **интервал** между ними в полутонах определяется по формуле:

$$I(F_1, F_2) = 12 \left| \log_2 \frac{F_2}{F_1} \right|$$

Например, интервал в один полутон означает отношение частот, равное $\sqrt[12]{2}$, а интервал в двенадцать полутонов (октава) соответствует отношению частот, равному 2.

Все звуки (ноты) хроматической гаммы имеют обозначения. Обозначение состоит из имени ноты и номера октавы. Имена нот повторяются через каждые 12 звуков. Номер октавы увеличивается на 1 через каждые 12 звуков. Ниже представлена таблица части хроматического ряда.

Номер	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2
Обозначение	C1	C#1 Db1	D1	D#1 Eb1	E1	F1	F#1 Gb1	G1	G#1 Ab1	A1	A#1 Bb1	B1
Номер	3	4	5	6	7	8	9	10	11	12	13	14
Обозначение	C2	C#2 Db2	D2	D#2 Eb2	E2	F2	F#2 Gb2	G2	G#2 Ab2	A2	A#2 Bb2	B2

Заметьте, что некоторые ноты можно обозначить несколькими способами. Эту таблицу можно продолжать бесконечно как в сторону уменьшения номера ноты, так и в сторону его увеличения. В данной задаче все ноты будут лежать в пределах от ДО малой октавы ($C0 = -21$) до СИ пятой октавы ($B5 = 50$) включительно.

Канонически считается, что частота ноты ЛЯ первой октавы ($A_1 = 0$) равна 440 герц. Учитывая то, каждая следующая нота имеет в $\sqrt[3]{2}$ большую частоту, а каждая предыдущая в $\sqrt[3]{2}$ раз меньшую частоту, можно определить частоты всех нот.

Нетрудно видеть, что вентильная труба с небольшим количеством вентиляей не способна идеально извлечь все ноты равномерно темперированной хроматической гаммы. Техника исполнения подразумевает нажатие такой комбинации вентиляей, чтобы ошибка была минимальной. То есть из всех 2^N вариантов нажатий вентиляей нужно выбрать один. Этот вариант однозначно определит длину колеблющегося столба воздуха. Вместе с этим нужно выбрать натуральный номер гармоники. Вместе эти два параметра однозначно определяют частоту звука. Требуется минимизировать *интервал* между этой частотой и эталонной частотой ноты, которую пытается сыграть исполнитель.

В данной задаче вам будет дано описание конфигурации трубы: количество вентиляей, длины основной трубки и кронов; а также последовательность нот, которую нужно сыграть. Вам требуется определить оптимальное исполнение нот, а также допускаемую при этом ошибку.

Входные данные

Первая строка входного файла содержит два целых числа: N — количество вентиляей на трубе ($1 \leq N \leq 10$) и L_0 — длину основной трубки трубы в миллиметрах ($100 \leq L_0 \leq 10000$). Вторая строка содержит N целых чисел — длины всех кронов в миллиметрах. Длина каждого крона лежит в пределах от 20 до 2000 миллиметров. На этом описание трубы заканчивается.

Следующая строка содержит число M — количество нот, которые требуется сыграть исполнителю ($1 \leq M \leq 100$). Затем идёт M строк, в каждой из которых записано два или три символа — обозначение ноты. Символ диеза '#' имеет ASCII-код 35. Ограничения на высоту ноты описаны выше в тексте условия.

Выходные данные

В выходном файле должно быть ровно M строк. Каждая строка должна содержать описание оптимального исполнения соответствующей ноты из входного файла.

В начале описания идёт слово из N символов — комбинация нажатия вентиляей. Каждый символ означает состояние соответствующего вентиля: символ '.' (ASCII 46) означает ненажатое состояние, а символ 'o' (ASCII 111) означает нажатое состояние. Второе, что нужно вывести, это номер используемой гармоники. Номер является целым числом не менее двух (причина описана в тексте условия). Последнее, что нужно вывести — это интервал между требуемой частотой и частотой, получаемой при оптимальном исполнении. Интервал следует выдавать в полутонах как вещественное число с абсолютной погрешностью не более 10^{-6} . Гарантируется, что оптимальное исполнение единственно.

Примеры

<i>input.txt</i>	<i>output.txt</i>
3 1300 159 77 261 1 C0	ooo 2 6.389290
3 1300 159 77 261 11 G1 C2 G1 A1 B1 E1 E1 A1 G1 F1 G1	... 3 0.013785 ... 4 0.005765 ... 3 0.013785 oo. 4 0.106236 .o. 4 0.001968 oo. 3 0.125786 oo. 3 0.125786 oo. 4 0.106236 ... 3 0.013785 o.. 3 0.016166 ... 3 0.013785

3 1300	ooo 2 0.389290
159 77 261	o.o 2 0.147472
31	.oo 2 0.006850
F#0	oo. 2 0.106236
G0	o.. 2 0.003384
G#0	.o. 2 0.001968
A0	... 2 0.005765
A#0	ooo 3 0.408840
B0	o.o 3 0.167022
C1	.oo 3 0.012700
C#1	oo. 3 0.125786
D1	o.. 3 0.016166
D#1	.o. 3 0.017582
E1	... 3 0.013785
F1	.oo 4 0.006850
F#1	oo. 4 0.106236
G1	o.. 4 0.003384
G#1	.o. 4 0.001968
A1	... 4 0.005765
A#1	oo. 5 0.030627
B1	o.. 5 0.140247
C2	.oo 6 0.012700
C#2	ooo 7 0.077549
D2	o.. 6 0.016166
D#2	.o. 6 0.017582
E2	... 6 0.013785
F2	.oo 8 0.006850
F#2	oo. 8 0.106236
G2	o.. 8 0.003384
G#2	.o. 8 0.001968
A2	... 8 0.005765
A#2	
B2	
C3	

Комментарии

Во всех тестах из примеров дана конфигурация стандартной вентиляльной трубы.

На этой трубе основной тон попадает на ноту ДО малой октавы (C0). Таким образом, можно не нажимая вентилях сыграть с относительно малой погрешностью ноты C1, G1, C2, E2, G2, Bb2, C3 и так далее.

Длины кронов подобраны таким образом, что при нажатии первого вентиля высота ноты понижается на 2 полутона, при нажатии второго – на 1 полутона, а при нажатии третьего – на 3 полутона. При нажатии нескольких вентилях одновременно все снижения суммируются. Например, поскольку G1 играется без нажатия вентилях, то можно нажав первый вентиль сыграть F1, а нажав второй и третий вентиль сыграть Eb1. Все эти факты верны лишь с некоторой точностью.

В первом тесте требуется извлечь основной тон трубы. Поскольку это невозможно, то лучшее решение – нажать все вентилях и выбрать минимальную гармонику. При этом ошибка очень большая: примерно 6 полутонов.

Во втором тесте даны первые 11 нот из гимна России.

В третьем тесте записана полная хроматическая гамма в классическом диапазоне трубы: от ФА-диез малой октавы (F#0) до ДО третьей октавы (C3). В ответе на этот тест записана аппликатура, с которой знаком любой трубач. Именно так нажимаются вентилях при игре на трубе. Единственное отличие ответа на тест от классической аппликатуры — МИ второй октавы (E2), которое обычно играется без нажатия вентилях.

Задача 4. Ресторан

Ограничение по времени на 1 тест: 3 сек.

В этом году в курортном городе X очень жарко. Настолько жарко, что люди, оказавшиеся на открытом солнце, в мгновение ока получают тепловой удар.

Предприимчивый дядя Коля решил открыть в городе ресторан. В данный момент он тщательно планирует местоположение ресторана. Дядя Коля оптимист, поэтому он уверен, что его ресторан будет самым лучшим, и не учитывает возможную конкуренцию. Вот что он действительно вынужден учитывать, так это палящее солнце.

В модели дяди Коли город представлен в виде графа: вершины представляют районы города, а рёбра – крытые переходы между районами. В любой момент времени в зависимости от положения солнца каждый район либо будет полностью освещён солнцем, либо будет полностью находиться в тени. Переходы всегда находятся в тени. Дяде Коле нужно выбрать район, в котором лучше всего открыть ресторан.

Дядя Коля уверен, что ни один отдыхающий не пойдёт в его ресторан, если по пути ему придётся идти по открытому солнцу, в частности, если под солнцем находится ресторан или район, из которого начинается путь отдыхающий. Можно считать, что отдыхающие перемещаются по затенённым местам города мгновенно.

Дядя Коля составил карту «людности» города. Для каждого района он оценил «людность» — количество людей в единицу времени, у которых появляется желание сходить в ресторан. Когда отдыхающему приходит в голову мысль сходить в ресторан, он определяет, сможет ли он до него дойти, всегда оставаясь в тени. Если это возможно, то отдыхающий приходит в ресторан, и дядя Коля имеет с него прибыль в 1 рубль. Если это невозможно, то отдыхающий переключается на другие дела.

Дядя Коля уже определил время работы своего будущего ресторана. Никаких перерывов в работе ресторана не будет. Беспощадное солнце, терроризирующее город, в течение дня перемещается. Из-за этого районы города могут менять освещённость. Какие-то районы уходят в тень, другие оказываются на солнце. Смена освещённости в каждом районе происходит мгновенно.

В начале дня все районы находятся в тени. Во время восхода солнца некоторые районы становятся освещёнными. Через некоторое время солнце начинает опускаться, и освещённый район уходит в тень. В конце дня все районы вновь оказываются в тени. Обратите внимание, что район не может уйти в тень в процессе восхода солнца, также как район не может стать освещённым в процессе захода солнца.

Дядя Коля хочет максимизировать дневной заработок. Определите по данным дяди Коли, в каком районе города нужно открыть ресторан, чтобы прибыль ресторана за день была максимальна.

Входные данные

В первой строке входного файла записано четыре целых числа: N – количество районов города, M – количество переходов между ними, T_O – время открытия ресторана и T_C – время закрытия ресторана ($1 \leq N, M \leq 100000, 0 \leq T_O < T_C \leq 10^7$).

В следующей строке записаны числа людности районов города – N целых чисел, лежащих в диапазоне от 0 до 10^6 включительно.

Затем идёт M строк, описывающих переходы между районами. Каждая строка содержит по два целых числа A_i и B_i – номера районов, соединённых переходом ($1 \leq A_i \neq B_i \leq N$). Никакой переход не повторяется во входных данных дважды. Отдыхающие могут перемещаться по каждому переходу в обоих направлениях.

Затем описано расписание, по которому меняется освещённость районов города. Оно начинается строкой, содержащей одно целое число Q – количество изменений освещённости за день ($0 \leq Q \leq 2N$). В следующих Q строках описаны смены освещённости районов в строгом хронологическом порядке.

Каждая строка содержит символ (плюс или минус) и два целых числа. Первое число T_j – момент времени, в который происходит смена освещённости ($0 \leq T_j \leq 10^7$). Второе число V_j – номер района, в котором меняется освещённость ($1 \leq V_j \leq N$). Символ плюс означает, что район

становится освещённым, а символ минус – что район уходит в тень. В расписании любой район либо не упоминается вообще, либо упоминается ровно два раза – по одному разу с каждым знаком. Все смены освещённости со знаком плюс происходят раньше всех смен освещённости со знаком минус.

Выходные данные

В выходной файл необходимо вывести два целых числа: номер района, в котором лучше всего разместить ресторан, и получающийся при этом дневной заработок. Если оптимальное решение не единственно, выведите любое из них.

Пример

<i>input.txt</i>	<i>output.txt</i>
3 2 0 10 1 1 1 1 2 2 3 6 + 1 3 + 2 2 + 3 1 - 4 1 - 5 2 - 6 3	1 21

Комментарий

В примере указаны три района, соединённых переходами линейным образом. В каждом районе появляется один отдыхающий в единицу времени. Можно нарисовать состояние районов города на протяжении дня.

Отрезок времени	Состояние районов
0-1	###
1-2	##.
2-3	#..
3-4	...
4-5	#..
5-6	##.
6-7	###
7-8	###
8-9	###
9-10	###

Очевидно, что если открыть ресторан в районе 1, то все отдыхающие смогут до него добраться в любой момент времени за исключением промежутка 3-4. Если же открыть ресторан в районе 3, то в отрезок времени 1-6 ресторан будет освещен, а значит, не будет иметь посетителей, и прибыль при этом составит всего 15 рублей.

Задача 5. Суперкомпьютер

Ограничение по времени на 1 тест: **2 сек.**

для задач на Java: **3 сек.**

International Brotherhood of Magicians (IBM) разработали новейший суперкомпьютер. Отличительной особенностью этого суперкомпьютера является то, что он содержит бесконечное количество процессоров и неограниченный объем памяти! Более того, благодаря применению архитектуры гиперкуб и тахионной шины, время доступа процессоров к памяти равно нулю. Тем не менее, вычисления с плавающей запятой все же требуют, как правило, некоторых временных затрат. В частности, время сложения и вычитания равно t_a , время умножения — t_m , деления — t_d , вычисления квадратного корня из числа — t_s . Время операции смены знака равно нулю. Однако операции сравнения чисел с плавающей запятой пока работают неэффективно и поэтому заблокированы. Кроме того, за счет аппаратной реализации библиотеки *libastral*, операции в целочисленной арифметике, включая сравнения, а также все операции перехода выполняются мгновенно. К сожалению, реализация ее все же не полна и не позволяет сразу узнавать ответ на любую задачу.

Одним из первых покупателей этого суперкомпьютера оказалась некая организация MJ12. Задача, которую требуется решить им, формулируется следующим образом. Пусть имеется симметричная матрица A размера $N \times N$ с действительными коэффициентами, для которой известно, на каких позициях находятся ее ненулевые коэффициенты. *Разложением Холецкого* матрицы A называется такая нижне-треугольная матрица L , что $A = LL^T$, где L^T — транспонированная матрица L . Для того чтобы L существовала, необходимо, чтобы коэффициенты матрицы A удовлетворяли определенным условиям. Мы предполагаем, что эти условия выполняются. Тогда коэффициенты L могут быть вычислены по следующим рекуррентным формулам:

$$L_{jj} = \sqrt{A_{jj} - \sum_{k < j} L_{jk}^2}$$

$$L_{ij} = \frac{A_{ij} - \sum_{k < j} L_{ik} L_{jk}}{L_{jj}}, i > j$$

MJ12 требуется вычислять по заданной матрице A соответствующую ей матрицу L . Ясно, что поскольку некоторые коэффициенты матрицы A являются нулевыми, то некоторые коэффициенты матрицы L также обязательно будут нулевыми, и их вычислять не требуется. Все остальные коэффициенты матрицы L вычислять необходимо. Перед покупкой суперкомпьютера MJ12 хотят узнать, насколько быстро он сможет решать эту задачу на их сверхсекретных матрицах. Чтобы не предоставлять их IBM, они попросили предоставить им программу, которая по заданным позициям ненулевых элементов матрицы A вычисляет минимальное время, за которое суперкомпьютер сможет вычислить коэффициенты матрицы L . Так как инженеры IBM заняты «допиливанием» *libastral*, они поручили написать такую программу вам.

MJ12 требуют, чтобы вычисление L проводилось в точном соответствии с указанными формулами. Однако это может быть неэффективно, и вам удалось убедить их разрешить использовать следующие тождества для оптимизации вычислений:

1. $a + b = b + a$, $a - b = -(b - a)$
2. $(a + b) + c = a + (b + c)$, $(a - b) + c = a - (b - c)$, $(a - b) - c = a - (b + c)$
3. $0 + a = a + 0 = a$, $a - 0 = a$, $0 - a = -a$
4. $0 \cdot a = a \cdot 0 = 0$

Тождества (1) – (2), фактически, разрешают складывать слагаемые под знаком корня и в числителе дроби в выражениях в произвольном порядке, а тождества (3) – (4) — разрешают учитывать структуру матриц L и A и не складывать с тождественным нулем и не умножать на тождественный ноль.

Не забудьте, что суперкомпьютер имеет бесконечно много процессоров. Модель параллельных вычислений на бесконечном числе процессоров подразумевает, что если есть арифметическая операция, которую требуется вычислить, то она начнет выполняться сразу, как только будут вычислены все ее аргументы.

Входные данные

В первой строке входного файла записано два целых числа N и M ($1 \leq N \leq 50000$, $0 \leq M \leq 200000$) — размер матрицы A и количество внедиагональных элементов в нижней треугольной ее половине. В следующих M строках содержатся пары чисел r_j и c_j — строка и столбец, в котором находится внедиагональный элемент j ($1 \leq c_j < r_j \leq N$). Гарантируется, что каждый элемент будет указан ровно один раз. Ненулевые элементы на диагонали матрицы во входном файле не задаются, так как предполагается, что вся диагональ отлична от нуля. Последняя строка входного файла содержит четверку целых неотрицательных чисел t_a , t_m , t_d и t_s — времена вычисления соответствующих операций. Эти числа не превосходят 10^9 .

Выходные данные

В единственную строку выходного файла необходимо вывести одно целое число — минимальное время, требуемое для вычисления разложения Холецкого матрицы, заданной во входном файле. Гарантируется, что количество ненулевых элементов матрицы L , включая элементы на диагонали, не превзойдет 300000.

Примеры

<i>input.txt</i>	<i>output.txt</i>
2 0 1 1 1 1	1
3 1 3 2 1 2 3 4	14

Задача 6. Клонирование

Ограничение по времени на 1 тест: **2 сек.**

В Институте магического копипаста (ИМК) все было спокойно, ученые-волшебники изучали методы копирования, писали статьи, получали гранты, жили счастливо и не ведали бед, пока не пришла инспекция. Инспекторы оценили состояние института как плачевное и дали свой вердикт: "В ИМК мало сотрудников, популярность этого института низка, а финансирование слишком высоко. Следует урезать финансирование". Однако ректору института Копирайко И.В. удалось убедить инспекторов, что это всего лишь временная проблема, связанная с демографической ямой, и что ИМК скоро приобретет популярность. Он уговорил инспекторов провести повторную проверку через некоторое время. После ухода инспекторов сотрудники задумались над способами разрешения сложившейся ситуации.

Молодой специалист по магии копирования живых существ Людоклонов А.В. предложил свой вариант решения проблемы, который всем сразу же пришелся по душе. Он предложил использовать его новое, только разработанное, заклинание для клонирования домашнего скота на себе, чтобы получить столько сотрудников, сколько нужно. Однако все было не так просто. Ректор обратился в Институт Идеальных Чисел (ИИЧ), в котором ему подсчитали, что для того, чтобы финансирование не урезали, нужно, чтобы стало ровно Y сотрудников (Y — это новое изобретенное в ИИЧ магическое число). Не будем забывать, что дело происходит в очень далеком 30000 году и для института в порядке вещей иметь 10^8 сотрудников. Заклинание Людоклонова работает следующим образом. Собирается какое-то количество из всех имеющихся на данный момент сотрудников ИМК и читается заклинание. Для каждого чтения заклинания может быть собрано различное количество сотрудников. В результате каждый сотрудник института клонируется в такое количество людей, которое было в группе, читающей заклинание. В любой момент времени читается только одно заклинание. Теперь перед сотрудниками встала задача, как наклонировать Y сотрудников максимально быстро, ведь чтение заклинания один только раз занимает целую неделю, а инспекторы не любят ждать слишком долго. Тогда они попросили Вас помочь им решить данную задачу.

Входные данные

В единственной строке входного файла задано два целых числа X и Y , где X — реальное количество сотрудников института, а Y — нужное количество сотрудников ($0 < X < Y \leq 10^8$).

Выходные данные

В единственную строку выходного файла необходимо вывести одно число — минимальное количество чтений заклинания, которое понадобится, чтобы получить Y сотрудников из имеющихся X . Если это невозможно, то вывести слово **Impossible**.

Пример

<i>input.txt</i>	<i>output.txt</i>
3 18	2

Комментарий

В примере сначала 3 человека читают заклинание, получается 9 сотрудников. Затем 2 человека читают заклинание, и количество всех сотрудников увеличивается в 2 раза.

Задача 7. Отдых на природеОграничение по времени на 1 тест: **1 сек.**

В одно лесничество любили приезжать горожане на отдых. До домика лесника их довозил автобус, а дальше они разбредались по окрестным полянкам, загорали, устраивали пикники. Но нередко бывало так, что обратную дорогу до домика лесника удавалось найти не сразу. Некоторые заблудившиеся горожане долго ходили от одной полянки до другой по многочисленным тропинкам. Лесник захотел помочь гражданам. Он решил на каждой полянке поставить номер, а на тропинке поставить стрелочку так, чтобы если бы горожанин шел от домика лесника до какой-либо полянки по направлениям стрелочек, проходя по другим полянкам не более одного раза, то номера полянок все время увеличивались бы. Если же он решит вернуться назад, то он должен идти против стрелочек. Тогда уменьшение номеров полянок, через которые он будет проходить, будет говорить ему, что он идет в правильном направлении. После перенумерации все полянки должны иметь различные номера от 1 до n .

У лесника есть план леса, на котором указаны полянки и соединяющие их тропинки. Любые две полянки могут соединяться более чем одной тропинкой.

Лесник стрелки расставил, но с номерами у него проблемы. Помогите леснику перенумеровать полянки в соответствии с его задумкой.

Входные данные

Во входном файле записано несколько тестовых наборов данных.

В первой строке каждого тестового набора через пробел записаны целые числа n , m и s — количество полянок, количество тропинок и номер полянки, на которой находится домик лесника. Все полянки пронумерованы числами от 1 до n .

В следующих m строках описаны тропинки — это два целых числа, обозначающих номера полянок, которые соединяет тропинка в направлении стрелки, указанной лесником. В некоторых случаях эти числа могут совпадать.

Сумма n по всем тестам не превосходит 1000. Сумма m по всем тестам не превосходит 1000.

Входной файл заканчивается строкой **0 0 0**.

Выходные данные

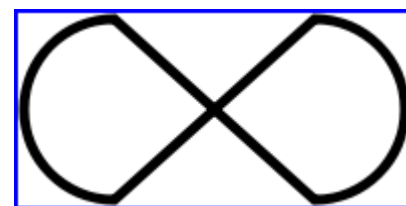
Для каждого теста на отдельной строке в выходной файл необходимо вывести нумерацию полянок. Если нужной нумерации не существует, то вывести фразу **No solution**.

Пример

<i>input.txt</i>	<i>output.txt</i>
3 3 1	1 2 3
1 2	2 3 4 1
2 3	No solution
1 3	
4 4 4	
4 1	
1 2	
2 3	
3 1	
4 6 1	
1 2	
1 3	
1 4	
2 3	
3 4	
4 2	
0 0 0	

Задача 8. Бег восьмеркамиОграничение по времени на 1 тест: **1 сек.**

Обычно, когда пользователь интернетов волнуется, он начинает забавно суетиться и бегать восьмёрками. При этом для бега он использует всё возможное пространство, заполняя своим волнением любой доступный объём. Чем больше восьмёрок пробежит пользователь, тем сильнее он успокоится.



ЖЖ-юзер Пётр очень волнуется в прямоугольной комнате.

Восьмёрка для бега выглядит следующим образом: две полуокружности, касающиеся трёх сторон комнаты и отрезки, соединяющие перекрёстно концы полуокружностей. Каждая полуокружность касается одной более короткой стороны комнаты и двух длинных.

У Петра есть достаточно свободного времени для бега, бежит он с постоянной скоростью, поэтому, если знать длину восьмерки, то вычислить, сколько полных восьмёрок он пробежит, не составит ничего сложного.

Входные данные

В первой строке входного файла задано два целых числа A и B — размеры комнаты в метрах ($1 \leq A, B \leq 10000$). Известно, что комната не является квадратной.

Выходные данные

В выходной файл необходимо вывести одно вещественное число — длину периметра восьмёрки в метрах с точностью до сантиметров.

Пример

	<i>input.txt</i>	<i>output.txt</i>
1	2	5.97

Задача 9. Полёт ладьиОграничение по времени на 1 тест: **1 сек.**

За пару минут до исторического броска ладьёй в одноглазого шахматиста Остап Бендер решает в уме непростую задачу: как бы ее так бросить, чтобы наверняка поразить цель?

Рассмотрим полет спрятанной в рукаве ладьи подробнее.

Первые X метров ладья летит с постоянным ускорением a . При этом, как известно (вам, а не комбинатору), ее скорость изменяется по закону $v = at$, а пройденное расстояние $l = \frac{at^2}{2}$, где t — время, прошедшее от начала броска.

Следующие Y метров ладья летит в плотных слоях атмосферы, сопротивление которой прямо пропорционально скорости ладьи с коэффициентом k . Здесь ее скорость изменяется по закону: $v = v_0 \cdot e^{-kt}$, а пройденное расстояние $l = \frac{v_0}{k} - \frac{v_0 e^{-kt}}{k}$, где v_0 — скорость, с которой ладья входит в плотные слои, и она совпадает с конечной скоростью, приданной великим комбинатором, k — коэффициент сопротивления, t — время с момента входа в атмосферу.

При этом, если время, за которое ладья преодолевает все это расстояние $X+Y$, больше наперед заданной величины T_0 , то бросок не достигает своей цели — одноглазый шахматист успевает уклониться. Естественно, Остап не должен этого допустить. Необходимо сделать так, чтобы общее время полета ладьи к цели не превышало T_0 , и при этом придать ладье минимально возможное ускорение, чтобы не тратить лишние силы — они ещё пригодятся Остапу во время побега от разъяренной толпы любителей шахмат.

Но если в шахматы великий комбинатор все-таки хоть раз, но играл, то уж такие-то мудреные формулы он точно не видел и с поставленной задачей он заведомо не справится. Помогите ему.

Входные данные

Во входном файле на одной строке записаны через пробел четыре вещественных числа X , Y , k и T_0 , где X и Y — длины первого и второго участков полета ладьи, k — коэффициент сопротивления и T_0 — время реакции одноглазого шахматиста-любителя ($0.1 \leq X, Y \leq 100$, $0.1 \leq k \leq 1$, $0.1 \leq T_0 \leq 10$).

Выходные данные

В выходной файл необходимо вывести одно вещественное число — необходимое минимальное ускорение a с точностью до 10^{-6} . Гарантируется, что это число не превосходит 100.

Пример

<i>input.txt</i>	<i>output.txt</i>
2.0 3.0 1.0 4.0	2.567201

Задача 10. Рогалик

Ограничение по времени на 1 тест: **2 сек.**
 для задач на Java: **4 сек.**

Рогаликами сокращённо называют жанр компьютерных игр в стиле rogue-like. Эти игры всегда работают в текстовом режиме, поэтому игровое поле изображается как прямоугольное поле, заполненное ASCII-символами. Каждый символ в клетке поля характеризует то, что находится в данной клетке. Например, в этой задаче будут использоваться лишь два вида клеток: сплошная стена ('#') и свободная клетка ('.').

Текстовое изображение карты накладывает некоторые интересные ограничения на игровой процесс. Рассмотрим в качестве примера механику работы заклинания «смертельный луч» из одного рогалика. Это заклинание создаёт луч, который летит по прямой и отражается от стен. Однако в качестве направлений выстрела можно выбрать лишь один из восьми вариантов, которые соответствуют цифрам 1 (вниз влево), 2 (вниз), 3 (вниз вправо), 6 (вправо), 9 (вверх вправо), 8 (вверх), 7 (вверх влево), 4 (влево):

7	8	9
4	@	6
1	2	3

Теперь рассмотрим подробнее процесс прочтения заклинания. Игрок находится в некоторой свободной клетке и запускает луч в одном из восьми указанных выше направлений. Полёт луча моделируется пошагово. На первом шаге луч находится в клетке, в которой стоит игрок. На каждом следующем ходу луч сдвигается на одну клетку по направлению полёта. При попадании в стенку луч отражается по следующим правилам:

1. Если край стены в окрестности точки контакта является прямой линией, то отражение происходит по закону "угол падения равен углу отражения". Примеры:

####	####	####	####
####	####	####	####
....	.↑..	.↓..
.↑..↓..
Шаг 1	Шаг 2	Шаг 3	Шаг 4

####	####	####	####
####	####	####	####
....	.↗..	..↘.
↗....↘
Шаг 1	Шаг 2	Шаг 3	Шаг 4

2. Если произошло касание, то луч летит дальше. Например:

##..	##..	##..	##.↗
##..	##..	##↗.	##..
....	.↗..
↗....
Шаг 1	Шаг 2	Шаг 3	Шаг 4

##..	##..	##..	##.↗
##..	##..	##↗.	##..
..##	.↗##	..##	..##
↗.##	..##	..##	..##
Шаг 1	Шаг 2	Шаг 3	Шаг 4

3. Если произошло попадание в угол, которое не является касанием, то луч разворачивается в обратную сторону. Примеры:

####	####	####	####
####	####	####	####
..##	.↗##	.↙##	..##
↗.##	..##	..##	↙.##
Шаг 1	Шаг 2	Шаг 3	Шаг 4

..##	..##	..##	..##
..##	..##	..##	..##
....	.↗..	.↙..
↗....	↙....
Шаг 1	Шаг 2	Шаг 3	Шаг 4

После $(K+1)$ -го хода луч исчезает. В данном случае K – дальность действия заклинания. Если луч пролетает через клетку, в которой кто-то находится, то этот несчастный умирает. Это не оказывает никакого влияния на дальнейший полёт луча.

В нашей задаче вам дано одно игровое поле размера M на N клеток, а также Q запросов для него. В каждом запросе указано положение игрока, положение противника, а также дальность действия луча смерти. Нужно для каждого из восьми возможных направлений выстрела определить результат запуска луча. Результат обозначается одной буквой:

M (iss) – луч никого не задел.

W (in) – луч убил противника, не задев игрока.

L (oss) – луч убил игрока, не задев противника.

D (eath) – луч убил и игрока, и противника.

Входные данные

В первой строке входного файла записаны размеры поля: M – количество строк и N – количество столбцов ($1 \leq M, N \leq 800$). Следующие M строк содержат по N символов каждая – карта поля. Первый символ первой строки поля имеет координаты $(1, 1)$. В следующей строке содержится одно число Q – количество запросов ($1 \leq Q \leq 100000$). Описание каждого запроса занимает одну из последующих Q строк. Каждое описание содержит пять целых чисел, записанных через пробел: R_P и C_P – позицию игрока, R_E и C_E – позицию противника и D – дальность полёта луча смерти ($1 \leq R_P, R_E \leq M; 1 \leq C_P, C_E \leq N; 1 \leq D \leq 10^9$). И для игрока и для противника R обозначает номер строки, а C – номер столбца. Гарантируется, что игрок и противник находятся в различных пустых клетках. Считается, что за пределами заданной карты все клетки заполнены стенами ('#').

Выходные данные

Каждая из Q строк выходного файла должна содержать ровно восемь символов – ответ для соответствующего запроса. Для каждого из направлений в порядке **12369874** выведите по одному символу **M**, **W**, **L** или **D**.

Пример

<i>input.txt</i>	<i>output.txt</i>
3 3	MMLWMLLL
...	MMLMMLLL
.#. .	DLLLDLLL
...	
3	
1 1 1 3 2	
1 1 1 3 1	
1 1 3 3 10	

Задача 11. Серверы

Ограничение по времени на 1 тест: **3 сек.**
 для задач на Java: **4 сек.**

Компьютерная сеть в некотором доме строилась по принципу присоединения нового компьютера к последнему из уже подключенных. Никакие два компьютера, будучи подключенными в сеть, между собой дополнительно никак не связывались. Таким образом, в сеть были объединены последовательно N компьютеров. Соседи обменивались информацией между собой, но в какой-то момент поняли, что им нужны прокси-серверы. Компьютерное сообщество дома решило установить прокси-серверы ровно на K компьютеров. Осталось только решить, какие именно компьютеры выбрать для этой цели. Главным критерием является ежемесячная стоимость обслуживания серверами всех компьютеров.

Для каждого компьютера установлен тариф его обслуживания, выраженный в рублях за метр провода. Стоимость обслуживания одного компьютера каким-то сервером равна тарифу компьютера, умноженному на суммарную длину провода от этого компьютера до сервера, которым он обслуживается.

Ваша задача написать программу, которая выберет такие K компьютеров, чтобы установить на них прокси-серверы, что общие затраты на обслуживание всех компьютеров были бы минимальными.

Входные данные

В первой строке входного файла записано два целых числа N и K — количество компьютеров в сети и количество прокси-серверов, которые нужно установить ($1 \leq K \leq N \leq 2000$).

Все компьютеры в сети пронумерованы числами от 1 до N по порядку подключения.

Во второй строке записано одно целое число T_1 — тариф обслуживания первого компьютера.

В следующих $N - 1$ строках записано через пробел по два целых неотрицательных числа L_i , T_i — информация об остальных компьютерах в сети по порядку номеров. L_i — длина провода, соединяющего i -ый компьютер с соседним с меньшим номером, T_i — тариф обслуживания данного компьютера ($2 \leq i \leq N$). Все L_i и T_i не превышают 10^6 .

Выходные данные

В первую строку выходного файла необходимо вывести одно целое число — минимальную стоимость обслуживания всех компьютеров всеми серверами. Во второй строке должны быть записаны через пробел K номеров компьютеров, на которые необходимо установить серверы. При существовании нескольких вариантов размещения разрешается вывести любой.

Примеры

<i>input.txt</i>	<i>output.txt</i>
3 1 10 2 2 3 3	19 1
3 2 10 2 2 3 3	4 1 3