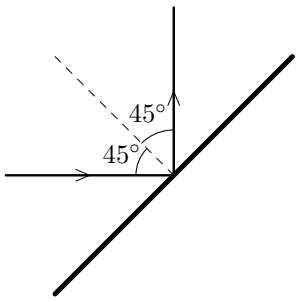
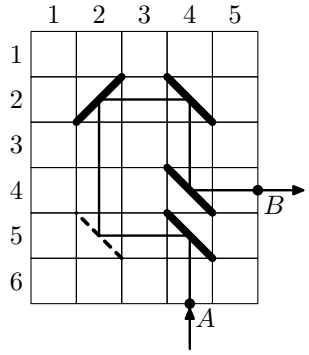
Задача 4. Pinball

Имя входного файла:	`pinball.in`
Имя выходного файла:	`pinball.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка:	100 баллов

Поле в Pinball представляет собой прямоугольник без стенок, состоящий из $n \times m$ квадратных клеток, (n клеток по вертикали, m клеток по горизонтали). Клетки по вертикали нумеруются сверху вниз, по горизонтали — слева направо. В каждой клетке можно установить одну отражающую пластинку в одном из двух положений: в положении 1 — от левого верхнего угла к правому нижнему или в положении 2 — от левого нижнего к правому верхнему. Летящий шарик при столкновении с пластинкой изменяет свою траекторию, при этом угол падения шарика всегда равен углу отражения и составляет 45° (см. рисунок).

На границе прямоугольника заданы две точки A и B, являющиеся серединами сторон некоторых клеток поля. Пластинки расставляются таким образом, чтобы шарик, запущенный из точки A, попал в точку B. При этом шарик начинает движение внутрь поля перпендикулярно стороне клетки, на которой находится точка A.





Изначально на поле были расставлены k пластинок таким образом, чтобы шарик попал из точки A в точку B. После этого одну из пластинок удалили. Необходимо определить, куда и как можно поставить удаленную пластинку, чтобы шарик, выпущенный из точки A, попал в точку B. При этом требуется, чтобы длина пути шарика была минимальной. Пластинку нужно поставить на некоторую свободную клетку даже в том случае, если шарик попадает в точку B и без нее.

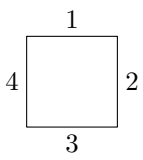
Требуется написать программу, устанавливающую пластинку таким образом, чтобы шарик попадал из точки A в точку B и длина его пути была минимальна.

Формат входных данных

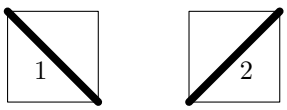
Первая строка входного файла содержит три числа: n, m ($1 \le n, m \le 1000$) и k, где k — общее количество пластинок, которые были исходно расставлены.

Во второй строке указываются номера клетки по вертикали и по горизонтали, на границе которой лежит точка A, и номер стороны, на которой она находится. Стороны клетки пронумерованы целыми числами от 1 до 4, при этом верхней стороне присвоен номер 1, далее по часовой стрелке нумеруются остальные стороны.

Третья строка содержит описание точки B в том же формате.



Номера сторон клеток



Возможные положения пластинок

Следующие $k-1$ строк описывают пластинки, оставшиеся на поле. В каждой строке записаны по три числа: первое — номер клетки по вертикали, второе — номер клетки по горизонтали, третье — положение пластинки в клетке (число 1 или 2).

Формат выходных данных

Выходной файл должен содержать три числа: номера клетки, в которую следует поставить пластинку, по вертикали и горизонтали и ее положение. Если решений несколько, выведите любое.

Пример

pinball.in	pinball.out
6 5 5 6 4 3 4 5 2 2 4 1 4 4 1 2 2 2 5 4 1	5 2 1

Задача 5. Великолепный Гоша

Имя входного файла:	`gosha.in`
Имя выходного файла:	`gosha.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка:	100 баллов

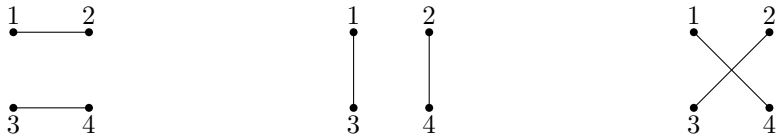
Телефонист Гоша получил задание соединить дома оптоволоконным кабелем по заданной схеме. На схеме указаны дома и их соединения между собой. При этом, согласно схеме может требоваться соединить два дома несколькими кабелями.

Когда Гоша приехал на работу, он обнаружил, что схему оставил дома. Однако, у него оказались записи, фиксирующие, сколько кабелей должно быть подведено к каждому из домов. Он попытался восстановить схему, соединяя дома с использованием только имеющихся записей. По окончании работ оказалось, что получившаяся сеть схеме не соответствует. На рис. 1 показаны пример схемы и сети, реализованной Гошей.

Схема Сеть Гоши

Рис. 1

На следующий день Гоша стал исправлять свою работу, глядя на верную схему. По техническим причинам он вынужден ограничиться последовательным выполнением операций только одного типа. В результате одной такой операции удаляются два кабеля, соединяющих некоторые две пары различных домов (см. рис. 2а), и эти же четыре дома соединяются либо как показано на рис. 2б, либо так, как на рис. 2в. Заметим, что после любой из таких операций количество кабелей, подведенных к каждому из домов, не меняется.



а) б) в)

Рис. 2

Требуется написать программу, которая определяет, сможет ли Гоша за один рабочий день перестроить сеть согласно схеме, если за один день он выполняет не более 48 000 операций. Если такая перестройка возможна, то программа должна выдать соответствующую последовательность операций.

Формат входных данных

В первой строке входного файла задано число N ($4 \le N \le 1\,000$) — количество домов на схеме.

Во второй строке записано число M ($2 \le M \le 20\,000$) — количество кабелей в схеме.

В следующих M строках расположены пары номеров домов, соединенных кабелями на схеме. Дома имеют номера от 1 до N.

Затем в M строках указаны пары номеров домов, которые Гоша соединил кабелями.

Формат выходных данных

Если перестроить сеть согласно схеме невозможно, или для перестройки сети требуется более 48 000 операций, то выходной файл должен содержать только число "`-1`". В противном случае

выходной файл должен содержать описание последовательности операций, по одной операции в строке.

Каждая операция описывается начальным и конечным положением кабелей. Сначала четырьмя целыми числами V_1, V_2, V_3, V_4 записывается исходное положение кабелей, при этом один из кабелей соединяет дома с номерами V_1, V_2, а другой — V_3, V_4. Затем в таком же формате описывается конечное положение.

Пример

gosha.in	gosha.out
6 6 1 2 2 3 1 4 2 5 5 4 5 6 1 4 1 4 2 5 3 6 2 5 5 2	1 4 5 2 1 2 5 4 2 5 3 6 2 3 5 6

Задача 6. Красная Шапочка

Имя входного файла:	`redhat.in`
Имя выходного файла:	`redhat.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка:	100 баллов

В непроходимом лесу имеется N полянок и M тропинок между ними. Каждая тропинка соединяет две различные полянки. Две полянки могут быть соединены несколькими тропинками.

На двух разных полянках живут Красная Шапочка и ее бабушка. Домик Красной Шапочки находится на полянке с номером 1, а домик бабушки — на полянке с номером N. Красная Шапочка хорошо ориентируется в лесу и знает, какое минимальное время ей потребуется для прохождения каждой тропинки. Когда Красная Шапочка идет по лесу, она переходит с тропинки на тропинку только на полянках. На каждой полянке есть укрытие, в котором Красная Шапочка может спрятаться на некоторое время.

В этом же лесу живет Волк. Время, за которое Волк пробегает какую-либо тропинку, может отличаться от времени, за которое по ней проходит Красная Шапочка. Кроме того, если Волк пробегает по одной и той же тропинке несколько раз, то каждый раз он может тратить на это разное время.

С края полянки, где живет Красная Шапочка, Волк увидел, что она собирается нести пирожки бабушке и побежал по тропинкам привычного ему пути от дома Красной Шапочки к дому бабушки. Волк начинает бежать от домика Красной Шапочки в тот момент, когда она решила выйти из дома, его путь заканчивается как только он окажется на полянке с домиком бабушки. Ни на одной полянке Волк не задерживается.

Чтобы застать бабушку в целости и сохранности, Красной Шапочке необходимо обогнать Волка. При этом ей нельзя оказаться с Волком на одной тропинке, даже если Волк уже покидает ее, а она только появляется на ней, или наоборот. Чтобы избежать встречи с Волком на полянке, Красная Шапочка использует имеющееся там укрытие. Красной Шапочке нельзя появляться на полянке одновременно с Волком или покидать укрытие на полянке в тот момент, когда на ней появляется Волк. При необходимости Красная Шапочка может идти по тропинке дольше минимально возможного времени, а также выйти из дома позже, чем она исходно решила.

Необходимо написать программу, которая поможет Красной Шапочке добраться к бабушке раньше Волка, если известна последовательность тропинок, по которым побежал Волк.

Формат входных данных

Первая строка входного файла содержит числа N, M и K ($2 \le N \le 2\,000$, $1 \le M \le 100\,000$, $1 \le K \le 100\,000$). Следующие M строк содержат по три числа: B_i, E_i — номера полянок, которые соединяет i-я тропинка, и T_i — минимальное время, за которое Красная Шапочка может по ней пройти ($1 \le T_i \le 10\,000$).

В следующих K строках находится последовательное описание пути Волка, по два числа в строке: P_i — номер тропинки, по которой он побежит, и V_i — время, которое он на это затратит ($1 \le V_i \le 10\,000$). Путь волка всегда начинается на полянке 1 и заканчивается на полянке N.

Все числа во входном файле целые и в пределах одной строки разделены пробелами.

Формат выходных данных

В том случае, если Красная Шапочка не может добраться до домика бабушки быстрее Волка, выходной файл должен содержать слово “`NO`”.

Если Красная Шапочка сможет добраться до домика бабушки быстрее волка, в первой строке выходного файла должно быть слово “`YES`”. Во второй строке в этом случае должно содержаться число тропинок в пути Красной Шапочки. В третью строку следует вывести номера тропинок в

том порядке, в котором Красная Шапочка должна по ним пройти. Числа должны быть разделены пробелами.

Информацию о времени прохождения по тропинкам и остановках на полянках в выходной файл выводить не нужно.

Пример

redhat.in	redhat.out
4 4 5 1 3 6 1 2 2 2 3 2 3 4 1 2 1 2 2 2 1 3 4 4 1	YES 2 1 4
4 3 4 1 2 2 2 3 1 2 4 3 1 2 2 1 2 2 3 5	NO