Задача A. Столица

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

Страна Фландия представляет собой n городов, соединённых дорогами с односторонним движением. Президент Фландии хочет разместить свою резиденцию в центре страны для удобства совершения различных деловых поездок. Центром он считает тот город, от которого сумма длин кратчайших путей до всех остальных городов минимальна. Напишите программу, находящую такой город.

Формат входных данных

В первой строке записаны числа n и m ($1 \leqslant n \leqslant 100$, $1 \leqslant m \leqslant n \cdot (n-1)$). Далее следуют m строк с описаниями дорог в виде «a_i b_i l_i» — дорога из a_i в b_i, которая имеет длину l_i ($1 \leqslant a_i, b_i \leqslant n$, $1 \leqslant l_i \leqslant 10^6$). Всегда верно, что из любого города Фландии можно попасть в любой другой.

Формат выходных данных

Выведите номер города, в котором нужно разместить резиденцию президента, и сумму кратчайших расстояний до всех городов. Если оптимальных решений несколько, разрешается выводить любое из них.

Пример

capital.in	capital.out
3 4 1 2 10 2 1 10 2 3 9 3 1 11	2 19

Задача B. Соединение точек

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

Даны N точек на плоскости. Требуется провести отрезки между некоторыми парами точек таким образом, чтобы, во-первых, из любой данной точки в любую можно было пройти по этим отрезкам, а во-вторых, суммарная длина проведённых отрезков была минимальна.

Формат входных данных

В первой строке входных данных задано число N — количество точек ($1 \leqslant N \leqslant 200$). Следующие N строк содержат по два числа X_i Y_i каждая через пробел — координаты i-й точки ($-1000 \leqslant X_i, Y_i \leqslant 1000$). Никакие две данные точки не совпадают, никакие три не лежат на одной прямой. Все числа во входных данных целые.

Формат выходных данных

В первой строке выведите L — суммарную длину проведённых отрезков с точностью не менее шести знаков после десятичной точки. Во второй строке выведите K — их количество. В следующих K строках выведите по два числа A_j B_j через пробел в каждой — номера точек, соединённых j-м отрезком ($1 \leqslant A_j, B_j \leqslant N$, $A_j \neq B_j$). Точки нумеруются с единицы в том порядке, в котором они даны во входных данных. Если ответов с минимальным L несколько, разрешается выводить любой из них.

Примеры

connect-points.in	connect-points.out
4 0 0 0 1 1 0 1 1	3 3 1 2 2 4 4 3
5 0 0 0 2 1 1 3 0 3 2	7.064495 4 3 1 3 2 3 4 4 5

Задача C. Диаметр графа

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

Дан связный взвешенный неориентированный граф.

Рассмотрим пару вершин, расстояние между которыми максимально среди всех пар вершин. Расстояние между ними называется *диаметром графа*. *Эксцентриситетом вершины* v называется максимальное расстояние от вершины v до других вершин графа. *Радиусом графа* называется наименьший из эксцентриситетов вершин. Найдите диаметр и радиус графа.

Формат входных данных

В первой строке задано единственное число N — количество вершин графа ($1 \leqslant N \leqslant 100$). В следующих N строках задано по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы всегда нули; веса рёбер не превышают 1000.

Формат выходных данных

Выведите два числа — диаметр и радиус графа.

Пример

diameter.in	diameter.out
4 0 -1 1 2 -1 0 -1 5 1 -1 0 4 2 5 4 0	8 5

Задача D. Дейкстра

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

Дан ориентированный взвешенный граф. Найдите кратчайшее расстояние от одной заданной вершины до другой.

Формат входных данных

В первой строке даны три числа: N, S и F ($1 \leqslant N \leqslant 1000$, $1 \leqslant S, F \leqslant N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках задано по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы всегда стоят нули. Веса всех рёбер целые и не превосходят $10\,000$.

Формат выходных данных

Вывести искомое расстояние или -1, если пути не существует.

Пример

dijkstra.in	dijkstra.out
3 1 2 0 -1 2 3 0 -1 -1 4 0	6

Задача E. Флойд

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

Дан ориентированный взвешенный граф. В нём вам необходимо найти пару вершин, кратчайшее расстояние от одной из которых до другой максимально среди всех пар вершин.

Формат входных данных

В первой строке запиасно целое число N — количество вершин графа ($1 \leqslant N \leqslant 100$). В каждой из следующих N строк задано по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса.

На главной диагонали матрицы всегда стоят нули. Все числа во входных данных не превосходят 100. Гарантируется, что хотя бы одно ребро в графе присутствует.

Формат выходных данных

Вывести искомое максимальное кратчайшее расстояние.

Пример

floyd.in	floyd.out
4 0 5 9 -1 -1 0 2 8 -1 -1 0 7 4 -1 -1 0	16

Задача F. Лабиринт знаний

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

Участникам тренировки подарили билеты на аттракцион «Лабиринт знаний». Лабиринт представляет собой N комнат, пронумерованных целыми числами от 1 до N, между некоторыми из которых есть двери. Когда человек проходит через дверь, показатель его знаний изменяется на определённую величину, фиксированную для данной двери. Вход в лабиринт находится в комнате 1, выход — в комнате N. Каждый участник сборов проходит лабиринт ровно один раз и наибрает некоторое количество знаний (при входе в лабиринт этот показатель равен нулю). Ваша задача — показать наилучший результат.

Формат входных данных

Первая строка содержит целые числа N и M — количество комнат и дверей ($1 \leqslant N \leqslant 2000$, $1 \leqslant M \leqslant 10\,000$). В каждой из следующих M строк содержится описание двери — номер комнаты, из которой она ведёт, и номер комнаты, в которую она ведёт (через дверь в лабиринте можно ходить только в одну сторону), а также целое число, которое прибавляется к количеству знаний при прохождении через дверь (это число по модулю не превышает $10\,000$). Двери могут вести из комнаты в неё саму, между двумя комнатами может быть более одной двери.

Формат выходных данных

Выведите «`:)`», если можно пройти лабиринт и получить неограниченно большой запас знаний, «`:(`», если лабиринт пройти нельзя, и максимальное количество набранных знаний в остальных случаях.

Пример

maze.in	maze.out
2 2 1 2 5 1 2 -5	5

Задача G. Цикл отрицательного веса

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

Дан ориентированный граф. Определите, есть ли в нём цикл отрицательного веса, и если да, то выведите его.

Формат входных данных

В первой строке задано число N — количество вершин графа ($1 \leqslant N \leqslant 100$). В следующих N строках находится по N чисел — матрица смежности графа. Все веса рёбер не превышают по модулю 10 000. Если ребра нет, то соответствующее число равно 100 000.

Формат выходных данных

В первой строке выведите «`YES`», если цикл существует или «`NO`» в противном случае. При его наличии выведите во второй строке количество вершин в искомом цикле и в третьей строке — вершины, входящие в этот цикл, в порядке обхода.

Пример

negcycle.in	**negcycle.out**
2 0 -1 -1 0	YES 2 1 2

Задача H. Порядок циклов

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

Дан неориентированный граф из n вершин, заданный матрицей смежности `a` (`a[u][u] = True`; `a[u][v] = a[v][u]`; `a[u][v] = True` тогда и только тогда, когда есть ребро между вершинами `u` и `v`). На нём запускают следующий алгоритм:

```
for x := 1 to n do
  for y := 1 to n do
    for z := 1 to n do
      if a[i][k] and a[k][j] then
        a[i][j] := True;
```

Перед запуском буквы `x`, `y` и `z` заменяют буквами `i`, `j` и `k` в некотором порядке. Утверждается, что после работы этого алгоритма `a[u][v] = True` тогда и только тогда, когда в исходном графе существует путь между вершинами `u` и `v`. Выясните, верно ли это, и если нет, приведите пример исходного графа, на котором это неверно.

Формат входных данных

В первой строке ввода записаны через пробел три буквы — «`i`», «`j`» и «`k`» — в некотором порядке. Первая буква подставляется в программу вместо «`x`», вторая — вместо «`y`», третья — вместо «`z`».

Формат выходных данных

Если искомый граф существует, в первой строке выведите через пробел целые числа n и m — количество вершин и рёбер в графе, соответственно ($1 \leqslant n \leqslant 10$, $0 \leqslant m \leqslant 45$). В следующих m строках выведите пары вершин, соединённых рёбрами, по одной паре на строке. Номера вершин в паре должны быть упорядочены по возрастанию; вершины нумеруются с единицы. Кратные рёбра и петли не допускаются.

Если же программа с заданным порядком циклов корректно работает на любом графе, вместо n и m выведите в первой строке два нуля через пробел.

Пример

order.in	order.out
`k i j`	`0 0`

Задача I. Найдите путь

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

Дан неориентированный взвешенный граф без петель и кратных рёбер. Также дана последовательность чисел. Выведите путь в графе, длины рёбер в котором образуют заданную последовательность. Если таких путей несколько, выведите лексикографически минимальный. Если же таких путей не существует, выведите «`No solution`» (без кавычек).

Обратите внимание, что число в последовательности может не соответствовать ни одному ребру заданного графа.

Пути в графе задаются как последовательности номеров вершин графа. Любые две соседние вершины в пути должны быть соединены ребром.

Путь a лексикографически меньше пути b той же длины тогда и только тогда, когда в первой позиции, в которой пути различаются, номер вершины в a меньше номера вершины в b.

Формат входных данных

В первой строке ввода заданы через пробел три целых числа n, m и k — количество вершин графа, рёбер графа и количество рёбер в искомом пути соответственно ($2 \leqslant n \leqslant 500$, $1 \leqslant m \leqslant \frac{n(n-1)}{2}$, $1 \leqslant k \leqslant 500$).

В следующих m строках описаны рёбра графа: в каждой строке заданы через пробел три целых числа u, v и l — номера двух смежных вершин и длина соединяющего их ребра ($1 \leqslant u, v \leqslant n$, $1 \leqslant l \leqslant 10^9$).

В следующей строке заданы через пробел k целых чисел. Эти числа также лежат в пределах от 1 до 10^9.

Формат выходных данных

Если искомого пути не существует, выведите строку «`No solution`». В противном случае в первой строке выведите искомый путь: последовательность из $k+1$ целого числа, состоящую из номеров первой, второй, ..., $(k+1)$-й вершин пути. Вершины графа нумеруются с единицы. Выведенный путь должен быть лексикографически минимальным. Соседние числа в последовательности следует разделять пробелами.

Примеры

pathfind.in	pathfind.out
3 3 3 1 2 10 2 3 20 1 3 30 30 10 20	3 1 2 3
3 3 3 1 2 10 2 3 20 1 3 30 30 10 30	No solution
3 3 3 1 2 10 2 3 20 1 3 30 30 10 40	No solution
4 5 6 1 2 10 2 3 10 1 3 10 1 4 10 2 4 10 10 10 10 10 10 10	1 2 1 2 1 2 1

Задача J. Подграф

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

Дан неориентированный граф. Требуется удалить из него некоторые рёбра так, чтобы получился связный подграф, содержащий все вершины исходного графа, в котором суммарная длина всех рёбер минимальна.

Формат входных данных

В первой строке заданы два целых числа n и m через пробел — количество вершин и рёбер в исходном графе, соответственно ($1 \leqslant n \leqslant 100$, $m \geqslant 0$). Следующие m строк описывают рёбра; i-я из них содержит три целых числа u_i, v_i и w_i — номера концов ребра и его длину ($1 \leqslant u_i, v_i \leqslant n$, $u_i \neq v_i$, $|w_i| \leqslant 10\,000$). Гарантируется, что между каждой парой вершин есть не более одного ребра.

Формат выходных данных

Если невозможно удалить рёбра так, чтобы получился связный подграф, выведите «`Impossible`» (без кавычек). В противном случае выведите одно число — минимальную сумму длин рёбер подграфа.

Пример

subgraph.in	subgraph.out
3 3 1 2 3 2 3 4 3 1 5	7

Задача K. Сумма расстояний

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

Дан связный граф. Требуется найти сумму расстояний между всеми парами вершин.

Формат входных данных

Первая строка содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($1 \leqslant n \leqslant 1000$, $0 \leqslant m \leqslant 10\,000$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Гарантируется, что граф связен.

Формат выходных данных

Первая строка должна содержать одно натуральное число — сумму попарных расстояний между вершинами.

Пример

sumdist.in	sumdist.out
5 5 1 2 2 3 3 4 5 3 1 5	16

Задача L. Синхронизация

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

Учёные взялись за какую-то большую и сложную проблему. Для её решения написали распределённое приложение. Процессы приложения во время решения задачи должны обмениваться сообщениями. Каждый учёный запустил один или несколько процессов приложения на своём компьютере, и спустя некоторое время решение проблемы было найдено.

После того, как решение было получено, возник вопрос о том, как происходил обмен данными между компьютерами. Но вот незадача — времена на компьютерах не были синхронизированы, и осталась только информация о времени запуска приложения, времена вызовов функций пересылки и приёма данных и их продолжительность.

Ваша задача — найти, на сколько отличаются времена на компьютерах, на которых считалась задача. Но из-за того, что время отправления сигнала регистрировалось в момент окончания отправления, а время получения сигнала регистрировалось в момент начала приёма сигнала, может сложиться такая ситуация, что определение возможной разности времён на компьютерах невозможно.

Формат входных данных

В первой строке вводятся через пробел два числа N и M ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$), где N — количество компьютеров, а M — количество пересланных пакетов данных. В следующих M строках задаются параметры пакетов данных: в каждой строке по три числа, записанных через пробел, i, j и Δt ($1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant N$, $-10\,000 \leqslant \Delta t \leqslant 10\,000$). Здесь i — номер компьютера, получившего пакет данных, j — номер компьютера, пославшего пакет данных, а Δt — разница между временем начала получения i-м компьютером и временем окончания отправки j-м компьютером пакета данных. В Δt входят относительные времена, которые измерялись на i-м и j-м компьютерах соответственно. Поэтому разность реальных времён начала получения и окончания отправки данных могла быть отлична от Δt, если времена на компьютерах отличались.

Формат выходных данных

Выведите N чисел, по одному в каждой строке. В i-й строке должна содержаться возможная разность между временем i-го компьютера и сервера (сервер — компьютер с номером 1). При этом рассчитанные времена должны удовлетворять тому условию, что момент начала получения был не раньше момента окончания отправки каждого из пакетов данных. Если возможных наборов времён несколько, то нужно выдать любой из них. Если возможных наборов времён нет, то необходимо выдать строку «`NO SOLUTION`» (без кавычек).

Пример

synchro.in	synchro.out
3 6 1 2 5 2 1 5 1 3 1 3 1 1 2 3 1 3 2 1	0 2 1