Задача A. Флойд

Имя входного файла:	`floyd.in`
Имя выходного файла:	`floyd.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Полный ориентированный взвешенный граф задан матрицей смежности. Постройте матрицу кратчайших путей между его вершинами.

Гарантируется, что в графе нет циклов отрицательного веса.

Формат входных данных

В первой строке вводится единственное число N ($1 \leqslant N \leqslant 100$) — количество вершин графа. В следующих N строках по N чисел задается матрица смежности графа (j-е число в i-й строке — вес ребра из вершины i в вершину j). Все числа по модулю не превышают 100. На главной диагонали матрицы — всегда нули.

Формат выходных данных

Выведите N строк по N чисел — матрицу расстояний между парами вершин, где j-е число в i-й строке равно весу кратчайшего пути из вершины i в j.

Примеры

floyd.in	floyd.out
4 0 5 9 100 100 0 2 8 100 100 0 7 4 100 100 0	0 5 7 13 12 0 2 8 11 16 0 7 4 9 11 0

Задача B. Кратчайшие пути

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

Вам дан взвешенный ориентированный граф и вершина s в нём. Для каждой вершины графа u выведите длину кратчайшего пути от вершины s до вершины u.

Формат входных данных

Первая строка входного файла содержит три целых числа n, m, s — количество вершин и ребёр в графе и номер начальной вершины соответственно ($2 \leqslant n \leqslant 2\,000$, $1 \leqslant m \leqslant 5\,000$).

Следующие m строчек описывают рёбра графа. Каждое ребро задаётся тремя числами — начальной вершиной, конечной вершиной и весом ребра соответственно. Вес ребра — целое число, не превосходящее 10^{15} по абсолютной величине. В графе могут быть кратные рёбра и петли.

Формат выходных данных

Выведите n строчек — для каждой вершины u выведите длину кратчайшего пути из s в u. Если не существует пути между s и u, выведите «*». Если не существует кратчайшего пути между s и u, выведите «-».

Примеры

path.in	path.out
6 7 1 1 2 10 2 3 5 1 3 100 3 5 7 5 4 10 4 3 -18 6 1 -1	0 10 - - - *

Задача C. Лабиринт знаний (25 баллов)

Имя входного файла:	`maze.in`
Имя выходного файла:	`maze.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В ЛКШ построили аттракцион «Лабиринт знаний». Лабиринт представляет собой N комнат, занумерованных 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 ЛКШонок проходит лабиринт ровно один раз и попадает в группу в зависимости от набранных знаний (при входе в лабиринт этот показатель равен нулю). Ваша задача показать наилучший результат.

Формат входных данных

Первая строка входного файла содержит целые числа N ($1 \leqslant N \leqslant 2\,000$) — количество комнат и M ($0 \leqslant M \leqslant 10\,000$) — количество дверей. В каждой из следующих M строк содержится описание двери — номера комнат, из которой она ведёт и в которую она ведёт, 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

В выходной файл выведите «`:)`» — если можно получить неограниченно большой запас знаний, «`:(`» — если лабиринт пройти нельзя, и максимальное количество набранных знаний в противном случае.

Примеры

maze.in	maze.out
2 2 1 2 3 1 2 7	7

Задача D. Цикл отрицательного веса

Имя входного файла:	`negcycle.in`
Имя выходного файла:	`negcycle.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайта

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

Формат входных данных

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

Формат выходных данных

В первой строке выходного файла 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.

Примеры

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

Задача E. Диаметр графа

Имя входного файла:	`diameter.in`
Имя выходного файла:	`diameter.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	16 мегабайта

Дан связный взвешенный неориентированный граф.

Рассмотрим пару вершин, расстояние между которыми максимально среди всех пар вершин.

- Расстояние (длина кратчайшего пути) между ними называется *диаметром графа*.
- *Эксцентриситетом вершины* v называется максимальное из расстояний 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 входных данных содержится единственное число: n ($1 \leqslant n \leqslant 100$) — количество вершин графа. В следующих n строках по n чисел — матрица длин ребер графа, где -1 означает отсутствие ребра между вершинами, а любое целое неотрицательное число, не превосходящее 1/,000 — присутствие ребра данного веса. На главной диагонали матрицы всегда нули.

Гарантируется, что матрица симметрична, а граф — связный.

Формат выходных данных

Выведите два числа — диаметр и радиус графа.

Примеры

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

Задача F. Заправки

Имя входного файла:	`gasstation.in`
Имя выходного файла:	`gasstation.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В стране N городов, некоторые из которых соединены между собой дорогами. Для того, чтобы проехать по одной дороге, требуется один бак бензина. В каждом городе бак бензина имеет разную стоимость. Вам требуется добраться из первого города в N-й, потратив как можно меньшее количество денег.

Дополнительно имеется канистра для бензина, куда входит столько же бензина, сколько входит в бак. В каждом городе можно заправить бак, залить бензин в канистру, залить и туда и туда, или же перелить бензин из канистры в бак.

Формат входных данных

В первой строке входного файла записано N чисел ($1 \leqslant N \leqslant 25$), i-е из которых задает стоимость бензина в i-м городе (все числа целые в диапазоне от 0 до 100).

Во следующих строках описаны все дороги (по одной в строке). Каждая дорога задается двумя числами — номерами городов, которые она соединяет. Все дороги двухсторонние, между двумя городами существует не более одной дороги, не существует дорог, ведущих из города в себя.

Формат выходных данных

В выходной файл выведите одно число — суммарную стоимость маршрута или -1, если добраться до нужного города невозможно.

Примеры

gasstation.in	gasstation.out
1 10 2 15 1 2 1 3 4 2 4 3	2

Задача G. Рейсы во времени

Имя входного файла:	`time.in`
Имя выходного файла:	`time.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Между N населёнными пунктами совершаются пассажирские рейсы на машинах времени.

В момент времени 0 вы находитесь в пункте A. Вам дано расписание рейсов. Требуется оказаться в пункте B как можно раньше (то есть в наименьший возможный момент времени).

При этом разрешается делать пересадки с одного рейса на другой. Если вы прибываете в некоторый пункт в момент времени T, то вы можете уехать из него любым рейсом, который отправляется из этого пункта в момент времени T или позднее (но не раньше).

Формат входных данных

Первая строка входного файла содержит число N — количество населённых пунктов ($1 \leqslant N \leqslant 1\,000$). Вторая строка содержит два числа A и B — номера начального и конечного пунктов. Третья строка содержит число K — количество рейсов ($0 \leqslant K \leqslant 1000$). Следующие K строк содержат описания рейсов, по одному на строке. Каждое описание представляет собой четвёрку целых чисел. Первое число каждой четвёрки задаёт номер пункта отправления, второе — время отправления, третье — пункт назначения, четвёртое — время прибытия. Номера пунктов — натуральные числа из диапазона от 1 до N. Пункт назначения и пункт отправления могут совпадать. Время измеряется в некоторых абсолютных единицах и задаётся целым числом, по модулю не превышающим 10^9. Поскольку рейсы совершаются на машинах времени, то время прибытия может быть как больше времени отправления, так и меньше, или равным ему.

Гарантируется, что входные данные таковы, что добраться из пункта A в пункт B всегда можно.

Формат выходных данных

Выведите в выходной файл минимальное время, когда вы сможете оказаться в пункте B.

Примеры

time.in	time.out
2 1 1 2 1 1 2 10 1 10 1 9	0
1 1 1 3 1 3 1 -5 1 -5 1 -3 1 -4 1 -10	-10