# Задача A. Кратчайший путь

| | |
|---|---|
| Имя входного файла: | `distance.in` |
| Имя выходного файла: | `distance.out` |
| Ограничение по времени: | 3 секунды |
| Ограничение по памяти: | 64 мегабайта |

Дан неориентированный взвешенный граф.

Найти кратчайший путь между двумя данными вершинами.

## Формат входных данных

Первая строка входного файла содержит натуральные числа $N$ и $M$ ($N \leqslant 2\,000$, $M \leqslant 50\,000$) — количество вершин и ребер графа. Вторая строка входного файла содержит натуральные числа $S$ и $F$ ($1 \leqslant S, F \leqslant N$, $S \neq F$) — номера вершин, длину пути между которыми требуется найти Следующие $M$ строк по три натуральных числа $b_i$, $e_i$ и $w_i$ — номера концов $i$-ого ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100\,000$).

## Формат выходных данных

Первая строка должна содержать одно натуральное число — длина минимального пути между вершинами $S$ и $F$. Во второй строке через пробел выведите вершины на кратчайшем пути из $S$ в $F$ в порядке обхода. Если путь из $S$ в $F$ не существует, выведите $-1$.

## Примеры

| distance.in | distance.out |
|---|---|
| 4 4<br>1 3<br>1 2 1<br>2 3 2<br>3 4 5<br>4 1 4 | 3<br>1 2 3 |

# Задача B. 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 ($j$-ое число в $i$-ой 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 $j$-ое число в $i$-ой строке равно весу кратчайшего пути из вершины $i$ в $j$.

## Примеры

| floyd.in | floyd.out |
|---|---|
| 4<br>0 5 9 100<br>100 0 2 8<br>100 100 0 7<br>4 100 100 0 | 0 5 7 13<br>12 0 2 8<br>11 16 0 7<br>4 9 11 0 |

# Задача C. Pink Floyd

| | |
|---|---|
| Имя входного файла: | `floyd.in` |
| Имя выходного файла: | `floyd.out` |
| Ограничение по времени: | 3 секунды |
| Ограничение по памяти: | 64 мегабайта |

Группа Pink Floyd собирается отправиться в новый концертный тур по всему миру. По предыдущему опыту группа знает, что солист Роджер Уотерс постоянно нервничает при перелетах. На некоторых маршрутах он теряет вес от волнения, а на других — много ест и набирает вес.

Известно, что чем больше весит Роджер, тем лучше выступает группа, поэтому требуется спланировать перелеты так, чтобы вес Роджера на каждом концерте был максимально возможным.

Группа должна посещать города в том же порядке, в котором она дает концерты. При этом между концертами группа может посещать промежуточные города.

## Формат входных данных

Первая строка входного файла содержит три натуральных числа $n$, $m$ и $k$ — количество городов в мире, количество рейсов и количество концертов, которые должна дать группа соответственно ($n \leqslant 100$, $m \leqslant 10\,000$, $2 \leqslant k \leqslant 10\,000$). Города пронумерованы числами от 1 до $n$.

Следующие $m$ строк содержат описание рейсов, по одному на строке. Рейс номер $i$ описывается тремя числами $b_i$, $e_i$ и $w_i$ — номер начального и конечного города рейса и предполагаемое изменение веса Роджера в миллиграммах ($1 \leqslant b_i, e_i \leqslant n$, $-100\,000 \leqslant w_i \leqslant 100\,000$).

Последняя строка содержит числа $a_1, a_2, ..., a_k$ — номера городов, в которых проводятся концерты ($a_i \neq a_{i+1}$). В начале концертного тура группа находится в городе $a_1$.

Гарантируется, что группа может дать все концерты.

## Формат выходных данных

Первая строка выходного файла должна содержать число $l$ — количество рейсов, которые должна сделать группа. Вторая строка должна содержать $l$ чисел — номера используемых рейсов.

Если существует такая последовательность маршрутов между концертами, что Роджер будет набирать вес неограниченно, то первая строка выходного файла должна содержать строку “`infinitely kind`”.

## Примеры

| floyd.in | floyd.out |
|---|---|
| 4 8 5<br>1 2 -2<br>2 3 3<br>3 4 -5<br>4 1 3<br>1 3 2<br>3 1 -2<br>3 2 -3<br>2 4 -10<br>1 3 1 2 4 | 6<br>5 6 5 7 2 3 |

# Задача D. Цикл отрицательного веса

| | |
|---|---|
| Имя входного файла: | `negcycle.in` |
| Имя выходного файла: | `negcycle.out` |
| Ограничение по времени: | 3 секунды |
| Ограничение по памяти: | 64 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 $N$ ($1 \leqslant N \leqslant 80$) — 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<br>0 -1<br>-1 0 | YES<br>2<br>2 1 |

# Задача E. 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 ($1 \leqslant N \leqslant 25$), $i$-ое из которых задает стоимость бензина в $i$-ом 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<br>1 2<br>1 3<br>4 2<br>4 3 | 2 |

# Задача F. 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<br>1 1<br>2<br>1 1 2 10<br>1 10 1 9 | 0 |
| 1<br>1 1<br>3<br>1 3 1 -5<br>1 -5 1 -3<br>1 -4 1 -10 | -10 |