**Ограничение по времени работы программы: 4 секунды**

Неориентированный граф задан списком ребер. Найдите кратчайший путь от вершины a до вершины b.

ВХОДНЫЕ ДАННЫЕ

В первой строке входных данных идут целые числа n и m (1⩽n⩽50000, 1⩽m⩽100000) — количества вершин и ребер соответственно. Во второй строке идут целые числа a и b - начальная и конечная вершина соответственно. Далее идут m строк, описывающие ребра, каждая строка содержит номер начальной и конечной вершины.

ВЫХОДНЫЕ ДАННЫЕ

Если пути между a и b нет выведите единственное число -1. Иначе выведите в первой строке число l — длину кратчайшего пути между этими двумя вершинами в ребрах, а во второй строке выведите l+1 число — вершины этого пути.

ПРИМЕРЫ

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 4 5 1 4 1 3 3 2 2 4 2 1 2 3 | 2 1 2 4 |
| 4 4 2 3 2 1 2 4 4 3 1 3 | 2 2 1 3 |

**Ограничение по времени работы программы: 1 секунда**

На магическом калькуляторе есть три кнопки и табло. Табло может показывать не более чем четырехзначные числа. Каждая из кнопок меняет число некоторым образом: первая домножает его на 3, вторая прибавляет к нему сумму его цифр, а третья вычитает из него 2. В случае, если число становится отрицательным или превосходит 9999, калькулятор ломается. Сейчас на табло калькулятора горит число a. Определите, как получить на табло число b при помощи минимального числа нажатий.

ВХОДНЫЕ ДАННЫЕ

Единственная строка входных данных содержит два натуральных числа a и b, разделенные пробелом (1⩽a,b⩽9999).

ВЫХОДНЫЕ ДАННЫЕ

Выведите минимальное число действий, необходимых для получения из числа a числа b.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 14 45 | 3 |
| 18 12 | 3 |
| 14 29 | 2 |

Дана таблица, состоящая из N строк и M столбцов. В каждой клетке таблицы записано одно из чисел: 0 или 1. Расстоянием между клетками (x1, y1) и (x2, y2) назовем сумму |x1-x2|+|y1-y2|. Вам необходимо построить таблицу, в клетке (i, j) которой будет записано минимальное расстояние между клеткой (i, j) начальной таблицы и клеткой, в которой записана 1. Гарантируется, что хотя бы одна 1 в таблице есть.

## **Входные данные**

В первой строке вводятся два натуральных числа N и M, не превосходящих 500. Далее идут N строк по M чисел - элементы таблицы.

## **Выходные данные**

Требуется вывести N строк по M чисел - элементы искомой таблицы.

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 2 3  0 0 1  1 0 0 | 1 1 0  0 1 1 |

**Ограничение по времени работы программы: 1 секунда**

Ориентированный взвешенный граф задан так же, как в предыдущей задаче. Найдите кратчайший путь из вершины S в вершину F.

ВХОДНЫЕ ДАННЫЕ

В первой строке содержатся три числа: N, S и F (1⩽N⩽100, 1⩽S,F⩽N), где N — количество вершин графа, S — начальная вершина, а F – конечная. В следующих N строках вводится по N чисел, не превосходящих 100 — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы записаны нули.

ВЫХОДНЫЕ ДАННЫЕ

Требуется вывести последовательно все вершины одного (любого) из кратчайших путей, или одно число -1, если пути между указанными вершинами не существует.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 3 1 2 0 -1 2 3 0 -1 -1 4 0 | 1 3 2 |

**Ограничение по времени работы программы: 1 секунда**

В стране N городов, некоторые из которых соединены между собой дорогами. Для того, чтобы проехать по одной дороге, требуется один бак бензина. Помимо этого у вас есть канистра для бензина, куда входит столько же топлива, сколько входит в бензобак.

В каждом городе бак бензина имеет разную стоимость. Вам требуется добраться из первого города в N-й, потратив как можно меньшее денег.

В каждом городе можно заправить бак, заправить бак и канистру или же перелить бензин из канистры в бак. Это позволяет экономить деньги, покупая бензин в тех городах, где он стоит дешевле, но канистры хватает только на одну заправку бака!

ВХОДНЫЕ ДАННЫЕ

В первой строке вводится число N (1⩽N⩽100), в следующей строке идет N чисел, i-е из которых задает стоимость бензина в i-м городе (всё это целые числа из диапазона от 0 до 100). Затем идет число M – количество дорог в стране, далее идет описание самих дорог. Каждая дорога задается двумя числами – номерами городов, которые она соединяет. Все дороги двухсторонние (то есть по ним можно ездить как в одну, так и в другую сторону), между двумя городами всегда существует не более одной дороги, не существует дорог, ведущих из города в себя.

ВЫХОДНЫЕ ДАННЫЕ

Требуется вывести одно число – суммарную стоимость маршрута или -1, если добраться невозможно.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 4 1 10 2 15 4 1 2 1 3 4 2 4 3 | 2 |

**Ограничение по времени работы программы: 1 секунда (для языка Python - 2 секунды)**

Дан ориентированный граф, в котором могут быть кратные ребра и петли. Каждое ребро имеет вес, выражающийся целым числом (возможно, отрицательным). Гарантируется, что циклы отрицательного веса отсутствуют.

Требуется посчитать длины кратчайших путей от вершины номер 1 до всех остальных вершин.

ВХОДНЫЕ ДАННЫЕ

В первой строке входных данных записано число N (1⩽N⩽100) – количество вершин графа и число M (0⩽M⩽10000) – количество ребер. В следующих строках идет M троек чисел, описывающих ребра: начало ребра, конец ребра и вес (вес – целое число от -100 до 100).

ВЫХОДНЫЕ ДАННЫЕ

Программа должна вывести N чисел – расстояния от вершины номер 1 до всех вершин графа. Если пути до соответствующей вершины не существует, вместо длины пути выведите число 30000.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 4 5 1 2 10 2 3 10 1 3 100 3 1 -10 2 3 1 | 0 10 11 30000 |

**Ограничение по времени работы программы: 1 секунда**

Профессору Форду необходимо попасть на международную конференцию. Он хочет потратить на дорогу наименьшее количество денег, поэтому решил, что будет путешествовать исключительно ночными авиарейсами (чтобы не тратиться на ночевку в отелях), а днем будет осматривать достопримечательности тех городов, через которые он будет проезжать транзитом. Он внимательно изучил расписание авиаперелетов и составил набор подходящих авиарейсов, выяснив, что перелеты на выбранных направлениях совершаются каждую ночь и за одну ночь он не сможет совершить два перелета.

Теперь профессор хочет найти путь наименьшей стоимости, учитывая что до конференции осталось K ночей (то есть профессор может совершить не более K перелетов).

ВХОДНЫЕ ДАННЫЕ

В первой строке находятся числа N (количество городов), M (количество авиарейсов), K (количество оставшихся ночей), S (номер города, в котором живет профессор), F (номер города, в котором проводится конференция).

Ограничения: 2⩽N⩽100, 1⩽M⩽5000, 1⩽K⩽100, 1⩽S⩽N, 1⩽F⩽N.

Далее идет M строк, задающих расписание авиарейсов. i-я строка содержит три натуральных числа: Si, Fi и Pi, где Si - номер города, из которого вылетает i-й рейс, Fi - номер города, в который прилетает i-й рейс, Pi - стоимость перелета i-м рейсом. 1⩽Si⩽N, 1⩽Fi⩽N, 1⩽Pi⩽106.

ВЫХОДНЫЕ ДАННЫЕ

Выведите одно число - минимальную стоимость пути, подходящего для профессора. Если профессор не сможет за K ночей добраться до конференции, выведите число -1.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 4 5 2 1 4 1 2 1 2 3 1 3 4 1 1 3 3 1 4 5 | 4 |

Олимпиадная

**Цикл отрицательного веса**

**Ограничение по времени работы: 1 секунда (для языка Python - 5 секунд)**

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

ВХОДНЫЕ ДАННЫЕ

В первой строке содержится число N (1⩽N⩽100) – количество вершин графа. В следующих N строках находится по N чисел – матрица смежности графа. Веса ребер по модулю меньше 100000. Если ребра нет, соответствующее значение равно 100000.

ВЫХОДНЫЕ ДАННЫЕ

В первой строке выведите "YES", если цикл существует, или "NO", в противном случае. При наличии цикла выведите во второй строке количество вершин в нем (считая одинаковые – первую и последнюю), а в третьей строке – вершины, входящие в этот цикл, в порядке обхода. Если циклов несколько, то  выведите любой из них.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 2 0 -1 -1 0 | YES 3 1 2 1 |

Вводная

**Простой Флойд**

**Ограничение по времени работы программы: 1 секунда (для Python - 5 секунд)**

Полный ориентированный взвешенный граф задан матрицей смежности. Постройте матрицу кратчайших путей между его вершинами. Гарантируется, что в графе нет циклов отрицательного веса.

ВХОДНЫЕ ДАННЫЕ

В первой строке вводится единственное число N (1⩽N⩽100) – количество вершин графа. В следующих N строках по N чисел задается матрица смежности графа (j-е число в i-й строке соответствует весу ребра из вершины i в вершину j). Все числа по модулю не превышают 100. На главной диагонали матрицы стоят нули.

ВЫХОДНЫЕ ДАННЫЕ

Выведите N строк по N чисел – матрицу кратчайших расстояний между парами вершин. j-е число в i-й строке должно быть равно весу кратчайшего пути из вершины i в вершину j.

Гарантируется, что для хранения длины пути достаточно 32-битного целого числа.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 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 |

Сложная

**Обмен валют**

**Ограничение по времени работы программы: 1 секунда**

Банк одной страны работает с N различными валютами. В банке утверждены курсы обмена любой валюты на любую другую. А именно, значение P[i][j] равно сумме, которую банк выплачивает в валюте j в обмен на единицу валюты i. При этом P[i][i]=1.

Профессор Флойд хочет разорить банк, заработав неограниченно большую сумму денег используя только те курсы обмена валют, которые установлены самим банком.

Выясните, удастся ли ему это сделать.

ВХОДНЫЕ ДАННЫЕ

Программа получает на вход число N (1⩽N⩽50). Далее идет N строк по N действительных неотрицательных чисел в каждой строке — курсы обмена валют. j-е число i-й строке равно числу единиц валюты j, которое выдает банк в обмен на единицу валюты i.

ВЫХОДНЫЕ ДАННЫЕ

Выведите слово YES, если профессор Флойд может разорить банк и слово NO в противном случае.

ПРИМЕР

|  |  |
| --- | --- |
| **ввод** | **вывод** |
| 2 1 2 0.5 1 | NO |
| 3 1 2 3 0.5 1 2 0.3 0.5 1 | YES |

Во втором примере можно обменять единицу валюты 1 на 2 единицы валюты 2, затем — на 4 единицы валюты 3 и, наконец, на 1,2 единицы валюты 1. Продолжая этот процесс можно получить сколь угодно большую сумму.