
Задача А. Обрати меня!

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

Мальчик Вася очень любит разворачивать ориентированные графы. Помогите ему в этом.

Формат входных данных

Во входном файле записано число N ($1 \leq N \leq 50\,000$) — количество вершин в графе. В следующих N строках записан граф в виде списков смежности: в i -й строке записаны номера вершин, в которые идут рёбра из i -й вершины. Нумерация начинается с единицы. Гарантируется, что рёбер в графе не более 50 000.

Формат выходных данных

Выведите развёрнутый граф в том же формате, что и исходный.

Примеры

<code>reverse.in</code>	<code>reverse.out</code>
4	4
2 3	
3	1 4
	1 2
2	

Задача В. Кратчайшее расстояние

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

Дан ориентированный граф. Найдите расстояния от вершины x до всех остальных вершин графа.

Формат входных данных

В первой строке входного файла содержатся два натуральных числа N и x ($1 \leq N \leq 1000$, $1 \leq x \leq N$) — количество вершин в графе и стартовая вершина соответственно. Далее в N строках по N чисел — матрица смежности графа: в i -й строке на j -м месте стоит «1», если вершины i и j соединены ребром, и «0», если ребра между ними нет. На главной диагонали матрицы стоят нули.

Формат выходных данных

Выведите через пробел числа d_1, d_2, \dots, d_n , где d_i — это -1 , если путей между x и i нет, и минимальное расстояние между x и i в противном случае.

Примеры

<code>mindist.in</code>	<code>mindist.out</code>
6 5 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0	2 2 1 1 0 -1

Задача С. Кратчайший путь

Имя входного файла: mindist2.in
Имя выходного файла: mindist2.out
Ограничение по времени: 3 секунды
Ограничение по памяти: 64 мегабайта

Вам дан неориентированный граф. Найдите кратчайший путь от вершины a до вершины b .

Формат входных данных

В первой строке входного файла идут целые числа n и m ($1 \leq n \leq 50\,000$, $1 \leq m \leq 100\,000$) — количества вершин и рёбер соответственно. Во второй строке идут целые числа a и b — стартовая и конечная вершины соответственно. Далее идут m строк, описывающих рёбра.

Формат выходных данных

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 рёбрах, а во второй строке выведите $l + 1$ число — вершины этого пути.

Примеры

mindist2.in	mindist2.out
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

Задача D. Шайтан-машинка

Имя входного файла: `crazycalc.in`
Имя выходного файла: `crazycalc.out`
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

У Ибрагима есть магическая чёрная шайтан-машинка. На ней есть три кнопки и табло. Табло может показывать не более чем четырёхзначные 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 9 999, шайтан-машинка ломается. Ибрагим может нажимать кнопки в любом порядке. Он хочет узнать, как получить на табло число b после некоторой последовательности нажатий, если сейчас шайтан-машинка показывает a . Помогите ему найти минимальное необходимое число нажатий.

Формат входных данных

Единственная строка входного файла содержит два натуральных числа a и b , разделённых пробелом ($1 \leq a, b \leq 9\,999$).

Формат выходных данных

Выведите одно число — минимальное необходимое количество действий.

Примеры

<code>crazycalc.in</code>	<code>crazycalc.out</code>
14 45	3
18 12	3
14 29	2

Задача Е. Детали

Имя входного файла: `details.in`
Имя выходного файла: `details.out`
Ограничение по времени: 4 секунды
Ограничение по памяти: 64 мегабайта

Предприятие «Авто-2005» выпускает двигатели для известных во всём мире автомобилей. Двигатель состоит ровно из n деталей, пронумерованных от 1 до n , при этом деталь с номером i изготавливается за p_i секунд. Специфика предприятия «Авто-2005» заключается в том, что там одновременно может изготавливаться лишь одна деталь двигателя. Для производства некоторых деталей необходимо иметь предварительно изготовленный набор других деталей.

Генеральный директор «Авто-2005» поставил перед предприятием амбициозную задачу — за наименьшее время изготовить деталь с номером 1, чтобы представить её на выставке.

Требуется написать программу, которая по заданным зависимостям порядка производства между деталями найдёт наименьшее время, за которое можно произвести деталь с номером 1.

Формат входных данных

Первая строка содержит n ($1 \leq n \leq 100\,000$) натуральных чисел p_1, p_2, \dots, p_n , определяющих время изготовления каждой детали в секундах.

Каждая из последующих n строк входного файла описывает характеристики производства деталей. Здесь i -я строка содержит список деталей, которые требуются для производства детали с номером i . В списке нет повторяющихся номеров деталей. Список может быть пустым: тогда ему будет соответствовать пустая строка! Сумма длин всех списков не превосходит 200 000.

Известно, что не существует циклических зависимостей в производстве деталей.

Формат выходных данных

В единственной строке выходного файла должно содержаться одно число: минимальное время (в секундах), необходимое для скорейшего производства детали с номером 1.

Примеры

details.in	details.out
100 200 300 2	300
2 1	

Долой списывание!

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

Во время теста Михаил Дмитриевич заметил, что некоторые лкшата обмениваются записками. Сначала он хотел поставить им всем двойки, но в тот день Михаил Дмитриевич был добрым, а потому решил разделить лкшат на две группы: списывающих и дающих списывать, и поставить двойки только первым.

У Михаила Дмитриевича записаны все пары лкшат, обменявшихся записками. Требуется определить, сможет ли он разделить лкшат на две группы так, чтобы любой обмен записками осуществлялся от лкшонка одной группы лкшонку другой группы.

Формат входных данных

В первой строке находятся два числа N и M — количество лкшат и количество пар лкшат, обменивающихся записками ($1 \leq N \leq 100$, $0 \leq M \leq \frac{N(N-1)}{2}$). Далее в M строках расположены описания пар лкшат: два различных числа, соответствующие номерам лкшат, обменивающихся записками (нумерация лкшат идёт с 1). Каждая пара лкшат перечислена не более одного раза.

Формат выходных данных

Необходимо вывести ответ на задачу Павла Олеговича. Если возможно разделить лкшат на две группы, выведите «YES»; иначе выведите «NO».

Примеры

bipartite.in	bipartite.out
3 2 1 2 2 3	YES

Лесопосадки

Имя входного файла: `tree.in`
Имя выходного файла: `tree.out`
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Дан неориентированный невзвешенный граф. Необходимо определить, является ли он деревом.

Формат входных данных

В первой строке входного файла содержится одно натуральное число N ($N \leq 100$) — количество вершин в графе. Далее в N строках по N чисел — матрица смежности графа: в i -ой строке на j -ом месте стоит 1, если вершины i и j соединены ребром, и 0, если ребра между ними нет. На главной диагонали матрицы стоят нули. Матрица симметрична относительно главной диагонали.

Формат выходных данных

Вывести «YES», если граф является деревом, «NO» иначе.

Примеры

tree.in	tree.out
6 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0	NO
3 0 1 0 1 0 1 0 1 0	YES

Компоненты связности

Имя входного файла: `components.in`
Имя выходного файла: `components.out`
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Дан неориентированный невзвешенный граф. Необходимо посчитать количество его компонент связности.

Формат входных данных

В первой строке входного файла содержится одно натуральное число N ($N \leq 100$) — количество вершин в графе. Далее в N строках по N чисел — матрица смежности графа: в i -й строке на j -м месте стоит «1», если вершины i и j соединены ребром, и «0», если ребра между ними нет. На главной диагонали матрицы стоят нули. Матрица симметрична относительно главной диагонали.

Формат выходных данных

Вывести одно целое число — искомое количество компонент связности графа.

Примеры

<code>components.in</code>	<code>components.out</code>
6 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0	3

Компоненты связности - 2

Имя входного файла: `matrix2.in`
Имя выходного файла: `matrix2.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Дан неориентированный невзвешенный граф. Необходимо посчитать количество его компонент связности и вывести их.

Формат входных данных

Во входном файле записано два числа N и M ($0 < N \leq 100\,000$), $0 \leq M \leq 100\,000$). В следующих M строках записаны по два числа i и j ($1 \leq i, j \leq N$), которые означают, что вершины i и j соединены ребром.

Формат выходных данных

В первой строчке выходного файла выведите количество компонент связности. Далее выведите сами компоненты связности в следующем формате: в первой строке количество вершин в компоненте, во второй — сами вершины в произвольном порядке.

Примеры

matrix2.in	matrix2.out
6 4	3
3 1	3
1 2	1 2 3
5 4	2
2 3	4 5
	1
	6

TopSort

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

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

В первой строке входного файла даны два натуральных числа N и M ($1 \leq N \leq 10^5, 1 \leq M \leq 10^5$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходных данных

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, требуется вывести -1 .

Примеры

topsort.in	topsort.out
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5
3 3 1 2 2 3 3 1	-1

Дейкстра

Имя входного файла: `dijkstra.in`
Имя выходного файла: `dijkstra.out`
Ограничение по времени: 4 секунды
Ограничение по памяти: 64 мегабайта

Дан ориентированный взвешенный граф.

Найдите кратчайшее расстояние от одной заданной вершины до другой.

Формат входных данных

В первой строке входного файла три числа: N , S и F ($1 \leq N \leq 1000, 1 \leq S, F \leq N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое целое неотрицательное число, не превосходящее $10\,000$ — присутствие ребра данного веса. На главной диагонали матрицы всегда нули.

Формат выходных данных

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

Примеры

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

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 ($N \leq 2000$, $M \leq 50\,000$) — количество вершин и ребер графа. Вторая строка входного файла содержит натуральные числа S и F ($1 \leq S, F \leq N$, $S \neq F$) — 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 ($1 \leq b_i, e_i \leq n$, $0 \leq w_i \leq 100\,000$).

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 1 3 1 2 1 2 3 2 3 4 5 4 1 4	3 1 2 3

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 ($1 \leq N \leq 100$) — 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	0 5 7 13
0 5 9 100	12 0 2 8
100 0 2 8	11 16 0 7
100 100 0 7	4 9 11 0
4 100 100 0	

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 ($1 \leq N \leq 80$) — 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

<code>negcycle.in</code>	<code>negcycle.out</code>
2	YES
0 -1	2
-1 0	2 1

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 ($1 \leq N \leq 25$), 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

<code>gasstation.in</code>	<code>gasstation.out</code>
1 10 2 15 1 2 1 3 4 2 4 3	2