Содержание

Пример работы с файлами.

Если вы не умеете читать/выводить данные, или открывать файлы, воспользуйтесь примерами. `http://acm.math.spbu.ru/~sk1/algo/sum/`

Пример работы с файлами.

В некоторых задачах большой ввод и вывод. Про ввод-вывод в C++:
`http://acm.math.spbu.ru/~sk1/algo/input-output/cpp_common.html`
Имеет смысл пользоваться супер быстрым вводом-выводом. Две версии:
`http://acm.math.spbu.ru/~sk1/algo/input-output/io_export.cpp.html`
`http://acm.math.spbu.ru/~sk1/algo/input-output/fread_write_export.cpp.html`

Выделение памяти.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу:
`http://acm.math.spbu.ru/~sk1/algo/memory.cpp.html`

Обязательные задачи

Задача 10A. Матрица инцидентности [0.3 sec, 256 mb]

Вершина графа u называется *инцидентной* ребру e, если u является одним из концов ребра e.

Аналогично, ребро e называется *инцидентным* вершине u, если один из концов e — это вершина u.

Матрицей инцидентности графа $G = (V, E)$ называется прямоугольная таблица из $|V|$ строк и $|E|$ столбцов, в которой на пересечении i-ой строки и j-го столбца записана единица, если вершина i инцидентна ребру j, и ноль в противном случае.

Дан неориентированный граф. Выведите его матрицу инцидентности.

Формат входных данных

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i. Рёбра нумеруются в том порядке, в котором они даны во входном файле, начиная с единицы.

Формат выходных данных

Выведите в выходной файл N строк, по M чисел в каждой. j-ый элемент i-ой строки должен быть равен единице, если вершина i инцидентна ребру j, и нулю в противном случае. Разделяйте соседние элементы строки одним пробелом.

Примеры

incident.in	incident.out
3 2 1 2 2 3	1 0 1 1 0 1
2 2 1 1 1 2	1 1 0 1

Задача 10B. От матрицы смежности к списку ребер [0.3 sec, 256 mb]

Простой неориентированный граф задан матрицей смежности, выведите его представление в виде списка ребер.

Формат входных данных

Входной файл содержит число N ($1 \leqslant N \leqslant 100$) — число вершин в графе, и затем N строк по N чисел, каждое из которых равно 0 или 1 — его матрицу смежности.

Формат выходных данных

Выведите в выходной файл список ребер заданного графа. Ребра можно выводить в произвольном порядке.

Пример

m2e.in	m2e.out
3 0 1 1 1 0 1 1 1 0	1 2 2 3 1 3

Задача 10C. Дерево [0.3 sec, 256 mb]

Дан неориентированный граф. Проверьте, является ли он деревом.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и m — количество вершин и рёбер в графе, соответственно ($1 \leqslant n \leqslant 100$). В следующих m строках заданы рёбра; i-я из этих строк содержит два целых числа u_i и v_i через пробел — номера концов i-го ребра ($1 \leqslant u_i, v_i \leqslant n$). Граф не содержит петель и кратных рёбер.

Формат выходных данных

В первой строке выходного файла выведите "`YES`", если граф является деревом, и "`NO`" в противном случае.

Примеры

tree.in	tree.out
3 2 1 2 1 3	YES
3 3 1 2 2 3 3 1	NO

Задача 10D. Связанность графа [0.3 sec, 256 mb]

Дан граф, содержащий N вершин и M ребёр ($1 \leqslant N \leqslant 1000, 1 \leqslant M \leqslant 7000$). Требуется найти наименьшее число рёбер и эти рёбра, которые нужно добавить, чтобы граф стал связным.

Формат входных данных

Во входном файле записаны сначала числа N и M, затем идёт описание рёбер графа — M пар чисел, где каждая пара описывает начало и конец ребра.

Формат выходных данных

В первую строку вывести единственное число K — минимальное количество рёбер, которое нужно добавить. В следующих K строках выведите по 2 числа — начало и конец нового ребра.

edges.in	edges.out
3 1 2 1	1 1 3

Задача 10E. Компоненты связности [0.3 sec, 256 mb]

Вам задан неориентированный граф с N вершинами и M ребрами ($1 \leqslant N \leqslant 20\,000$, $1 \leqslant M \leqslant 200\,000$). В графе отсутствуют петли и кратные ребра.

Определите компоненты связности заданного графа.

Формат входных данных

Граф задан во входном файле следующим образом: первая строка содержит числа N и M. Каждая из следующих M строк содержит описание ребра — два целых числа из диапазона от 1 до N — номера концов ребра.

Формат выходных данных

На первой строке выходного файла выведите число L — количество компонент связности заданного графа. На следующей строке выведите N чисел из диапазона от 1 до L — номера компонент связности, которым принадлежат соответствующие вершины. Компоненты связности следует занумеровать от 1 до L произвольным образом.

Пример

connect.in	connect.out
4 2 1 2 3 4	2 1 1 2 2

Задача 10F. TopSort. Топологическая сортировка [0.3 sec, 256 mb]

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

В первой строке входного файла даны два натуральных числа N и M ($1 \leqslant N \leqslant 100\,000, M \leqslant 100\,000$) — 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, вывести -1.

Пример

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

Задача 10G. Поиск пути на гриде [1 sec, 256 mb]

Дано прямоугольное поле $W \times H$. Некоторые клетки проходимы, через некоторые ходить нельзя. Из клетки можно ходить в соседние по ребру (слева, справа, сверху, снизу).

Нужно из клетки (x_1, y_1) найти любой (не обязательно кратчайший, даже не обязательно простой) путь в клетку (x_2, y_2).

Формат входных данных

На первой строке W, H, x_1, y_1, x_2, y_2 ($1 \leqslant x_1, x_2 \leqslant W \leqslant 1000$, $1 \leqslant y_1, y_2 \leqslant H \leqslant 1000$). Далее H строк, в каждой из которых по W символов. Символ "." означает, что клетка проходима, а символ "*" означает, что по ней ходить нельзя.

Клетки (x_1, y_1) и (x_2, y_2) не совпадают и обе проходимы.

Формат выходных данных

Если пути не существует, выведите `NO`.

Иначе выведите `YES` и последовательность клеток (x_i, y_i), в которой первая совпадает с клеткой (x_1, y_1), а последняя с клеткой (x_2, y_2).

Пример

dfsongrid.in	dfsongrid.out
4 2 1 1 4 2	YES 1 1 2 1 3 1 4 1 3 1 3 2 4 2
4 2 1 1 4 2 ..*. .*..	NO
4 2 1 1 4 2 ..*. *...	YES 1 1 2 1 2 2 3 2 4 2

Задача 10H. Поиск цикла [0.5 sec, 256 mb]

Дан ориентированный невзвешенный граф без петель и кратных рёбер. Необходимо определить есть ли в нём циклы, и если есть, то вывести любой из них.

Формат входных данных

В первой строке входного файла находятся два натуральных числа N и M ($1 \leqslant N \leqslant 100\,000$, $M \leqslant 100\,000$) — 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

Если в графе нет цикла, то вывести «NO», иначе — «YES» и затем перечислить все вершины в порядке обхода цикла.

Примеры

cycle.in	cycle.out
2 2 1 2 2 1	YES 1 2
3 3 1 2 2 3 1 3	NO

Задача 10I. Unique Topsort [1 sec, 256 mb]

Дан ориентированный ацикличный граф G. Проверить, что существует единственный топологический порядок вершин графа.

Формат входных данных

Первая строка входных данных содержит число вершин графа n ($1 \leqslant n \leqslant 100\,000$) и число ребер графа m ($0 \leqslant m \leqslant 100\,000$). Следующие m строк содержат пары чисел от 1 до n, задающие начало и конец соответствующего ребра. Гарантируется, что граф не содержит циклов.

Формат выходных данных

Если топологический порядок единственный, выведите на первой строке `YES`, а на второй номера вершин в топологическом порядке, иначе выведите `NO`.

Примеры

unitopsort.in	unitopsort.out
1 0	YES 1
2 1 2 1	YES 2 1
4 2 1 2 4 3	NO

Задача 10J. Condense 2. Конденсация графа [0.3 sec, 256 mb]

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 10\,000,\ m \leqslant 100\,000$). Следующие m строк содержат описание ребер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — началом и концом ребра соответственно ($1 \leqslant b_i,\ e_i \leqslant n$). В графе могут присутствовать кратные ребра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество ребер в конденсации графа.

Пример

condense2.in	condense2.out
4 4 2 1 3 2 2 3 4 3	2

Задача 10K. Bridges. Мосты [0.3 sec, 256 mb]

Дан неориентированный граф. Требуется найти все мосты в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 20\,000$, $m \leqslant 200\,000$).

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

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество мостов в заданном графе. На следующей строке выведите b целых чисел — номера ребер, которые являются мостами, в возрастающем порядке. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Пример

bridges.in	bridges.out
6 7 1 2 2 3 3 4 1 3 4 5 4 6 5 6	1 3

Задача 10L. Мосты и компоненты [0.6 sec, 256 mb]

Дан неориентированный граф (не обязательно связный). Граф может содержать петли и кратные ребра.

Выведите все компоненты реберной двусвязности графа (максимальные подмножества вершин, такие что подграф на них не теряет связность при удалении любого ребра).

Формат входных данных

Первая строка содержит числа n и m ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 100\,000$) — количество вершин и ребер в графе.

Следующие m строк задают ребра графа.

Формат выходных данных

В первой строке выведите количество компонент, в следующих за ней строках выведите сами компоненты, по одной на строку.

Вершины в каждой компоненте должны идти в возрастающем порядке, компоненты нужно вывести в лексикографическом порядке.

Примеры

bridges.in	bridges.out
3 2 1 2 2 3	3 1 2 3
3 3 1 2 2 3 3 1	1 1 2 3
2 2 1 2 1 2	1 1 2
7 8 1 5 5 6 1 6 5 4 4 3 4 2 3 2 7 2	3 1 5 6 2 3 4 7

Дополнительные задачи

Задача 10M. Points. Точки сочленения [0.3 sec, 256 mb]

Дан неориентированный граф без петель а кратных рёбер. Требуется найти все точки сочленения в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 20\,000$, $m \leqslant 200\,000$).

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

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество точек сочленения в заданном графе. На следующей строке выведите b целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

Пример

points.in	points.out
9 12 1 2 2 3 4 5 2 6 2 7 8 9 1 3 1 4 1 5 6 7 3 8 3 9	3 1 2 3

Задача 10N. Компоненты вершинной двусвязности [0.5 sec, 256 mb]

Компонентой вершинной двусвязности графа $\langle V, E \rangle$ называется максимальный по включению подграф (состоящий из вершин и ребер), такой что любые два ребра из него лежат на вершинно простом цикле.

Дан неориентированный граф без петель. Требуется выделить компоненты вершинной двусвязности в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и ребер графа соответственно ($n \leqslant 20\,000$, $m \leqslant 200\,000$).

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

Формат выходных данных

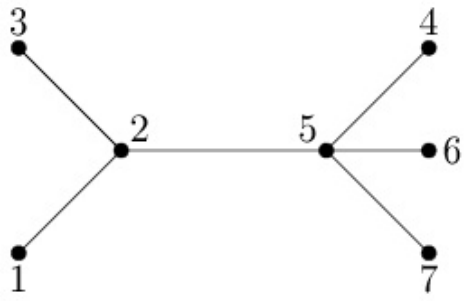
В первой строке выходного файла выведите целое число k — количество компонент вершинной двусвязности графа. Во второй строке выведите m натуральных чисел $a_1, a_2, \ldots, a_m$, не превосходящих k, где a_i — номер компоненты вершинной двусвязности, которой принадлежит i-е ребро. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Примеры

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

Задача 10O. Autotourism [2 sec, 256 mb]

В Байтландии существуют n городов, соединённых $n-1$ дорогами с двусторонним движением таким образом, что из каждого города можно проехать в любой другой по сети дорог. Длина каждой дороги равна 1 километру.



Бензобак автомобиля позволяет проехать без заправки m километров. Требуется выбрать маршрут, позволяющий посетить наибольшее количество различных городов без дозаправки. При этом начинать и заканчивать маршрут можно в произвольных городах.

Формат входных данных

В первой строке входного файла заданы два целых числа n и m ($2 \le n \le 500\,000$, $1 \le m \le 200\,000\,000$) — количество городов в стране и количество километров, которое автомобиль может проехать без дозаправки. В последующих $n-1$ строках описаны дороги. Каждая дорога задаётся двумя целыми числами a и b ($1 \leqslant a, b \leqslant n$) — номерами городов, которые она соединяет. Длина каждой дороги равна 1 км.

Формат выходных данных

Выведите одно число — максимальное количество городов, которое можно посетить без дозаправки.

Пример

autotourism.in	autotourism.out
7 6 1 2 2 3 2 5 5 6 5 7 5 4	5

Пояснение к примеру

5 городов можно посетить, например, по схеме $4 \to 5 \to 7 \to 5 \to 6 \to 5 \to 2$ или по схеме $3 \to 2 \to 1 \to 2 \to 5 \to 6 \to 5$.

Задача 10P. King's Assassination [1 sec, 256 mb]

Дан граф из n вершин и m ребер. Граф ориентированный. Нужно определить число вершин, содержащихся на всех путях из s в t (сами s и t учитывать не нужно).

Формат входных данных

Первая строка содержит n, m, s и t ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 300\,000$, $1 \leqslant s, t \leqslant n$, $s \neq t$).

Следующие m строк содержат пары чисел x_i и y_i — индексы вершин от 1 до n. Это означает что есть дорога из вершины с номером x_i в вершину с номером y_i.

Формат выходных данных

Число вершин k. Далее k чисел — номера вершин в возрастающем порядке.

Примеры

assassination.in	assassination.out
4 3 1 4 1 2 2 3 3 4	2 2 3
4 4 1 4 1 2 2 3 3 4 1 3	1 3
4 5 1 4 1 2 2 3 3 4 1 3 2 4	0

Задача 10Q. Сервера [2 sec, 256 mb]

Компьютерная сеть в некотором доме строилась по принципу присоединения нового компьютера к последнему из уже подключенных. Никакие два компьютера, будучи подключенными в сеть, между собой дополнительно никак не связывались. Таким образом, в сеть были объединены последовательно N компьютеров. Соседи обменивались информацией между собой, но в какой-то момент поняли, что им нужны прокси-серверы. Компьютерное сообщество дома решило установить прокси-серверы ровно на компьютеров. Осталось только решить, какие именно компьютеры выбрать для этой цели. Главным критерием является ежемесячная стоимость обслуживания серверами всех компьютеров.

Для каждого компьютера установлен тариф его обслуживания, выраженный в рублях за метр провода. Стоимость обслуживания одного компьютера каким-то сервером равна тарифу компьютера, умноженному на суммарную длину провода от этого компьютера до сервера, которым он обслуживается.

Ваша задача написать программу, которая выберет такие компьютеров, чтобы установить на них прокси-серверы, что общие затраты на обслуживание всех компьютеров были бы минимальными

Формат входных данных

В первой строке входного файла записано два целых числа N и K — количество компьютеров в сети и количество прокси-серверов, которые нужно установить ($1 \le K \le N \le 2000$).

Все компьютеры в сети пронумерованы числами от 1 до N по порядку подключения.

Во второй строке записано одно целое число T_1 — тариф обслуживания первого компьютера.

В следующих $N - 1$ строках записано через пробел по два целых неотрицательных числа L_i, T_i — информация об остальных компьютерах в сети по порядку номеров. L_i — длина провода, соединяющего $i-$ компьютер с соседним с меньшим номером, T_i — тариф обслуживания данного компьютера ($2 \le i \le N$). Все L_i и T_i от 0 до 10^6.

Формат выходных данных

В первую строку выходного файла необходимо вывести одно целое число — минимальную стоимость обслуживания всех компьютеров всеми серверами. Во второй строке должны быть записаны через пробел K номеров компьютеров, на которые необходимо установить серверы. При существовании нескольких вариантов размещения разрешается вывести любой.

Пример

server.in	server.out
3 1 10 2 2 3 3	19 1
3 2 10 2 2 3 3	4 1 3