Задача A. Топологическая сортировка

Имя входного файла:	`topsort.in`
Имя выходного файла:	`topsort.out`
Ограничение по времени:	2 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 ($1 \leqslant N \leqslant 100\,000$, $0 \leqslant M \leqslant 100\,000$) — 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 B. Конденсация графа

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

Дан ориентированный невзвешенный граф. Необходимо выделить в нем компоненты сильной связности и топологически их отсортировать.

Формат входных данных

В первой строке входного файла находятся два натуральных числа N и M ($1 \leqslant N \leqslant 20\,000, 1 \leqslant M \leqslant 200\,000$) — 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

Первая строка выходного файла должна содержать целое число k — количество компонент сильной связности в графе. Вторая строка выходного файла должна содержать n чисел — для каждой вершины выведите номер компоненты сильной связности, которой она принадлежит. Компоненты должны быть занумерованы таким образом, чтобы для каждого ребра (u, v) номер компоненты, которой принадлежит u не превосходил номер компоненты, которой принадлежит v.

Пример

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

Задача C. Кратчайший путь

Имя входного файла:	`shortpath.in`
Имя выходного файла:	`shortpath.out`
Ограничение по времени:	2 секунды

Дан ориентированный взвешенный ациклический граф. Требуется найти в нем кратчайший путь из вершины s в вершину t.

Формат входных данных

Первая строка входного файла содержит четыре целых числа n, m, s и t — количество вершин, дуг графа, начальная и конечная вершина соответственно. Следующие m строк содержат описания дуг по одной на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — началом, концом и длиной дуги соответственно ($1 \leqslant b_i, e_i \leqslant n$, $|w_i| \leqslant 1000$).

Входной граф не содержит циклов и петель.

$1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 200\,000$.

Формат выходных данных

Первая строка выходного файла должна содержать одно целое число — длину кратчайшего пути из s в t. Если пути из s в t не существует, выведите «Unreachable».

Пример

shortpath.in	shortpath.out
2 1 1 2 1 2 -10	-10
2 1 2 1 1 2 -10	Unreachable

Задача D. Игра

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

Дан ориентированный невзвешенный ациклический граф. На одной из вершин графа стоит «фишка». Двое играют в игру. Пусть «фишка» находится в вершине u, и в графе есть ребро (u, v). Тогда за ход разрешается перевести «фишку» из вершины u в вершину v. Проигрывает тот, кто не может сделать ход.

Формат входных данных

В первой строке входного файла находятся три натуральных числа N, M и S ($1 \leqslant N, S, M \leqslant 100\,000$) — количество вершин рёбер и вершина, в которой находится «фишка» в начале игры соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин.

Формат выходных данных

Если выигрывает игрок, который ходит первым, выведите «First player wins», иначе — «Second player wins».

Пример

game.in	game.out
3 3 1 1 2 2 3 1 3	First player wins
3 2 1 1 2 2 3	Second player wins

Задача E. Поиск цикла

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

Дан ориентированный невзвешенный граф. 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.

Пример

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

Задача F. Гамильтонов путь

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

Дан ориентированный граф без циклов. Требуется проверить, существует ли в нем путь, проходящий по всем вершинам.

Формат входных данных

Первая строка входного файла содержит два целых числа n и m — количество вершин и дуг графа соответственно. Следующие m строк содержат описания дуг по одной на строке. Ребро номер i описывается двумя натуральными числами b_i и e_i — началом и концом дуги соответственно ($1 \leqslant b_i, e_i \leqslant n$).

Входной граф не содержит циклов и петель.

$1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 200\,000$.

Формат выходных данных

Если граф удовлетворяет требуемому условию, то выведите `YES`, иначе `NO`.

Пример

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

Задача G. Компоненты связности

Имя входного файла:	`components.in`
Имя выходного файла:	`components.out`
Ограничение по времени:	2 секунды

Дан неориентированный граф. Требуется выделить компоненты связности в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 200\,000$).

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

Формат выходных данных

В первой строке выходного файла выведите целое число k — количество компонент связности графа. Во второй строке выведите n натуральных чисел $a_1, a_1, \ldots, a_n$, не превосходящих k, где a_i — номер компоненты связности, которой принадлежит i-я вершина.

Пример

components.in	components.out
3 1 1 2	2 1 1 2
4 2 1 3 2 4	2 2 1 2 1

Задача H. Мосты

Имя входного файла:	`bridges.in`
Имя выходного файла:	`bridges.out`
Ограничение по времени:	2 секунды

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 I. Точки сочленения

Имя входного файла:	`points.in`
Имя выходного файла:	`points.out`
Ограничение по времени:	2 секунды

Дан неориентированный граф. 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 1 3 2 3 1 4 4 5 1 5 2 6 6 7 2 7 3 8 8 9 3 9	3 1 2 3

Задача J. Двудольный граф

Имя входного файла:	`bipartite.in`
Имя выходного файла:	`bipartite.out`
Ограничение по времени:	2 секунды

Двудольным называется неориентированный граф $\langle V, E\rangle$, вершины которого можно разбить на два множества L и R, так что $L \cap R = \emptyset$, $L \cup R = V$ и для любого ребра $(u, v) \in E$ либо $u \in L, v \in R$, либо $v \in L, u \in R$.

Дан неориентированный граф. Требуется проверить, является ли он двудольным.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 200\,000$).

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

Формат выходных данных

В единственной строке выходного файла выведите «YES», если граф является двудольным и «NO» в противном случае.

Пример

bipartite.in	bipartite.out
4 4 1 2 1 3 2 4 4 2	YES
3 3 1 2 2 3 3 1	NO

Задача K. Компоненты реберной двусвязности

Имя входного файла:	`bicone.in`
Имя выходного файла:	`bicone.out`
Ограничение по времени:	2 секунды

Компонентой реберной двусвязности графа $\langle V, E \rangle$ называется подмножество вершин $S \subset V$, такое что для любых различных u и v из этого множества существует не менее двух реберно не пересекающихся пути из u в v.

Дан неориентированный граф. Требуется выделить компоненты реберной двусвязности 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

В первой строке выходного файла выведите целое число k — количество компонент реберной двусвязности графа. Во второй строке выведите n натуральных чисел $a_1, a_1, \ldots, a_n$, не превосходящих k, где a_i — номер компоненты реберной двусвязности, которой принадлежит i-я вершина.

Пример

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

Задача L. Компоненты вершинной двусвязности

Имя входного файла:	`biconv.in`
Имя выходного файла:	`biconv.out`
Ограничение по времени:	2 секунды

Компонентой вершинной двусвязности графа $\langle V, E\rangle$ называется подмножество ребер $S \subset E$, такое что любые два ребра из него лежат на вершинно простом цикле.

Дан неориентированный граф. Требуется выделить компоненты вершинной двусвязности 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

В первой строке выходного файла выведите целое число k — количество компонент вершинной двусвязности графа. Во второй строке выведите m натуральных чисел $a_1, a_1, \ldots, a_m$, 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.

Пример

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 M. Племя тив

Имя входного файла:	`tiv.in`
Имя выходного файла:	`tiv.out`
Ограничение по времени:	1 секунды
Ограничение по памяти:	256 мегабайт

Каждый год профессор Иванов ездит в Африку с целью изучить племена, которые там проживают. В этом году он ездил в гости к племени тив. Профессор довольно быстро научился понимать их язык, выучил многие их обряды, однако, он никак не мог понять записанные цифрами тив числа. Как и мы, члены племени используют позиционную систему счисления с основанием 26. Но цифры в племени тив обозначают символами, не похожими на обычные цифры от 0 до 25.

Профессор обозначил эти символы буквами от ‘a’ до ‘z’, но не может понять, какой цифре соответствует какой символ. Тогда вождь племени дал ему список из n неотрицательных чисел, записанных без ведущих нулей, и сказал, что числа в нем отсортированы строго по возрастанию.

Помогите профессору восстановить по этому списку какое-нибудь соответствие символов цифрам.

Формат входных данных

В первой строке входного файла дано одно натуральное числа n ($2 \leqslant n \leqslant 1000$) — количество слов в списке. Следующие n строк содержат выданные вождем числа племени тив, по одному числу в строке. Длина каждого числа не превышает 20.

Формат выходных данных

В первой строке файла выведите «`Yes`», если ответ существует, в этом случае в следующей строке выведите цифры, которые соответствуют символам, обозначенным ‘a’..‘z’, в этом порядке. Если существует несколько ответов, то выведете любой из них.

Если профессор понял что-то неправильно, и ответа не существует, выведете «`No`».

Примеры

tiv.in	tiv.out
4 a da dd cc	Yes 1 0 3 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
4 a j jb ac	No

Задача N. Авиаперелеты

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

Главного конструктора Петю попросили разработать новую модель самолета для компании «Air Бубундия». Оказалось, что самая сложная часть заключается в подборе оптимального размера топливного бака.

Главный картограф «Air Бубундия» Вася составил подробную карту Бубундии. На этой карте он отметил расход топлива для перелета между каждой парой городов.

Петя хочет сделать размер бака минимально возможным, для которого самолет сможет долететь от любого города в любой другой (возможно, с дозаправками в пути).

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 1000$) — число городов в Бубундии.

Далее идут n строк по n чисел каждая. j-ое число в i-ой строке равно расходу топлива при перелете из i-ого города в j-ый. Все числа не меньше нуля и меньше 10^9. Гарантируется, что для любого i в i-ой строчке i-ое число равно нулю.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — оптимальный размер бака.

Пример

avia.in	avia.out
4 0 10 12 16 11 0 8 9 10 13 0 22 13 10 17 0	10

Задача O. Установка ЧИПа

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

Новый ЧИП скоро установят в новый летательный апарат, недавно выпущенной компанией `Airtram`. ЧИП имеет форму диска. Есть n проводов, которые нужно подсоединить к ЧИПу.

Каждый провод можно подсоединить в один из двух разъемов, допустимых для этого провода. Все $2n$ разъемов расположены на границе диска. По кругу. Каждый провод имеет свой цвет. Для повышения безопасности два провода одного цвета не могут быть подсоединены к соседним разъемам.

Дана конфигурация разъемов на ЧИПе, найдите способ подсоединить все провода, не нарушающий условия про цвета.

Формат входных данных

Первая строка содержит число n — количество проводов ($1 \leqslant n \leqslant 50\,000$). Вторая строка содержит n целых чисел от 1 до 10^9 — цвета проводов. Третья строка содержит $2n$ целых чисел от 1 до n описывающих разъемы. Число обозначает номер провода, который может быть подсоединен к данному разъему. Каждое число от 1 до n встречается ровно дважды. Разъемы перечислены порядке "по кругу". 1-й разъем является соседним со 2-м и так далее, не забудьте, что n-й является соседним с 1-м.

Формат выходных данных

Если не существует способа подключить все провода, выведите одно слово “`NO`”.

Иначе выведите “`YES`” и n целых чисел. Для каждого провода выведите номер разъема, к которому нужно подключить этот провод. Разъемы нумеруются числами от 1 до $2n$ в том порядке, в котором они даны во входном файле.

Примеры

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

Задача P. P = NP

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

Мы рассмотрим переменные, которые принимают значения из множества $\{0, 1\}$. Литерал — это вхождение переменной с отрицанием или без. Литерал с отрицанием удовлетворяется тогда и только тогда, когда значение переменной равно 0. Литерал без отрицания удовлетворяется тогда и только тогда, когда значение переменной равно 1. Элементарная дизъюнкция — это множество литералов, и она дважды удовлетворяется тогда и только тогда, когда хотя бы два из ее литералов удовлетворены. КНФ формула — это множество элементарных дизъюнкций, и она дважды удовлетворяется тогда и только тогда, когда все ее элементарные дизъюнкции дважды удовлетворены.

Вам задана КНФ формула, в которой каждая элементарная дизъюнкция содержит ровно 3 различных литерала, и вы должны узнать, правда ли, что формула дважды удовлетворима.

Формат входных данных

Первая строка содержит два целых числа n ($1 \leqslant n \leqslant 100$) и m ($1 \leqslant m \leqslant 50\,000$). n — это число переменных в формула, а m — это число элементарных дизъюнкций. Переменные нумеруются от 1 до n.

Следующие m строк описывают каждую элементарную дизъюнкцию, каждая строка содержит три целых числа. Абсолютное значение каждого числа задает, какую переменную этот литерал представляет, а знак задает наличие отрицания (отрицательное число представляет литерал с отрицанием).

Формат выходных данных

Первая строка должна содержать "`YES`" или "`NO`", определяющее дважды удовлетворимость формулы. Если ответ "`YES`", то следующая строка должна содержать значения переменных, чтобы дважды удовлетворить формулу. Выведите n целых чисел, где i-е число равно i, если значение i-й переменной равно 1, и $-i$, если значение i-й переменной равно 0.

Пример

cnf.in	cnf.out
4 4 1 2 3 1 2 -3 3 4 1 3 4 -1	YES 1 2 3 4

Задача Q. Правда или Ложь?

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

Среди n коров ($2 \leqslant n \leqslant 1000$) Фермера Джона некоторые всегда говорят правду, а некоторые всегда лгут.

ФД выслушал m утверждений ($1 \leqslant m \leqslant 10\,000$) своих коров, каждое в форме «x y T», обозначающей, что корова x утверждает, что корова y всегда говорит правду или в форме «x y L», обозначающей, что корова x утверждает, что корова y всегда лжет.

Каждое утверждение вовлекает пару различных коров и одна и та же пара коров может появится во множестве утверждений.

К несчастью, не все записи утверждений были сделаны правильно. И теперь ФД хочет вычислить наибольшее a такое, что существует корректное распределение коровам статуса «лжеца» и «правдоруба», так, чтобы оказались корректными первые a высказываний из списка ФД.

Формат входных данных

Первая строка содержит два разделенных пробелом целых числа, n и m ($2 \leqslant n \leqslant 1000$; $1 \leqslant m \leqslant 10\,000$). Каждая следующая строка имеет форму «x y L» или «x y T», описывающее утверждение сделанное коровой x о корове y.

Формат выходных данных

Выведите максимальное значение a такое, что первые a утверждений из списка могут быть корректны при определенном назначении n коровам статусов «лжец», «правдец».

Примеры

truth.in	truth.out
4 3 1 4 L 2 3 T 4 1 T	2

Замечание

Утверждения 1 и 3 противоречивы всегда, а утверждения 1 и 2 могут быть непротиворечивы, если мы назначим коров 1…3 — «правдецами», а корову 4 — «лжецом».

Задача R. Тренировка

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

Вася и Вася усиленно тренируются к тандемному велосипедному марафону. Им нужно выбрать маршрут, на котором они будут тренироваться.

Вася и Вася живут в маленькой стране с n городами и m двусторонними дорогами. Тренировочный маршрут начинается в каком-то городе, проходит по некоторым дорогам и заканчивается в том же городе, откуда начался. Вася и Вася очень любят видеть новые места, поэтому они придумали правила: не посещать любой город дважды и не проезжать любую дорогу дважды. Тренировочный маршрут может стартовать в любом городе и не должен посетить все города. Ехать на заднем сиденье проще, потому что велосипедист защищен от ветра впереди сидящим партнером. Из-за этого Вася и Вася меняются местами в каждом городе, куда они приезжают. Для того, чтобы они одинаково потренировались, им требуется выбрать маршрут с четным числом дорог.

Помогите Васе и Васе найти такой тренировочный маршрут.

Формат входных данных

Входные данные содержать один или более тестов. Первая строка содержит T — число тестов. Далее описываются тесты.

Первая строка каждого теста состоит из двух целых чисел n и m — число городов и число дорог, соответственно ($2 \leqslant n \leqslant 100\,000$; $1 \leqslant m \leqslant 300\,000$). Следующие m содержат пары номеров городов, соединенных дорогой. Города пронумерованы от 1 до n. Никакая дорога не соединяет город с самим собой, никакие две дороги не соединяют одинаковую пару городов.

Гарантируется, что сумма n по всем тестам не превышает $100\,000$, и сумма m по всем тестам не превышает $300\,000$.

Формат выходных данных

Выведите ответы на все тесты.

Для каждого теста в первой строке выведите длину тренировочного маршрута или `-1`, если такого не существует. Если маршрут существует, в следующей строке выведите последовательность номеров городов в том порядке, в котором они идут в маршруте.

Если существует несколько решений, выведите любое. Обратите внимание, что длина маршрута может быть любым четным положительным целым числом.

Пример

even.in	even.out
2 4 4 1 3 3 2 2 4 4 1 5 5 1 3 3 2 2 4 4 5 5 1	4 1 3 2 4 -1

Задача S. Обход в глубину

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

Недавно на кружке по программированию Петя узнал об обходе в глубину. Обход в глубину используется во многих алгоритмах на графах. Петя сразу же реализовал обход в глубину на своих любимых языках программирования — паскале и си.

Си

```
int a[maxn + 1][maxn + 1];
int visited[maxn + 1];

void dfs(int v) {
  printf("%d\n", v);
  visited[v] = 1;
  for (int i = 1; i <= n; i++) {
    if ((a[v][i] != 0) && (visited[i] == 0)) {
      dfs(i);
      printf("%d\n", v);
    }
  }
}

void graph_dfs() {
  for (int i = 1; i <= n; i++) {
    if (visited[i] == 0) {
      dfs(i);
    }
  }
}
```

Петина программа хранит граф с использованием матрицы смежности в массиве «a» (вершины графа пронумерованы от 1 до n). В массиве «visited» помечается, в каких вершинах обход в глубину уже побывал.

Петя запустил процедуру «graph_dfs» для некоторого неориентированного графа G с n вершинами и сохранил ее вывод. А вот сам граф потерялся. Теперь Пете интересно, какое максимальное количество ребер могло быть в графе G. Помогите ему выяснить это!

Формат входных данных

Первая строка входного файла содержит два целых числа: n и l — количество вершин в графе и количество чисел в выведенной последовательности ($1 \leqslant n \leqslant 300$, $1 \leqslant l \leqslant 2n - 1$). Следующие l строк по одному числу — вывод Петиной программы. Гарантируется, что существует хотя бы один граф, запуск программы Пети на котором приводит к приведенному во входном файле выводу.

Формат выходных данных

На первой строке выходного файла выведите одно число m — максимальное возможное количество ребер в графе. Следующие m строк должны содержать по два целых числа — номера вершин, соединенных ребрами. В графе не должно быть петель и кратных ребер.

Примеры

dfs.in	dfs.out
6 10 1 2 3 2 4 2 1 5 6 5	6 1 2 1 3 1 4 2 3 2 4 5 6