

Задача А. Получи дерево

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 Мб

Дан связный неориентированный граф без петель и кратных ребер. Разрешается удалять из него ребра. Требуется получить дерево.

Формат входных данных

Сначала вводятся два числа: N (от 1 до 100) и M — количество вершин и ребер графа соответственно. Далее идет M пар чисел, задающих ребра. Гарантируется, что граф связный.

Формат выходных данных

Выведите $N - 1$ пару чисел — ребра, которые войдут в дерево. Ребра можно выводить в любом порядке.

Пример

tree.in	tree.out
4 4	1 2
1 2	2 3
2 3	3 4
3 4	
4 1	

Задача В. Банкет

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 Мб

На банкет были приглашены N Очень Важных Персон (ОВП). Были поставлены 2 стола. Столы достаточно большие, чтобы все посетители банкета могли сесть за любой из них. Проблема заключается в том, что некоторые ОВП не ладят друг с другом и не могут сидеть за одним столом. Вас попросили определить, возможно ли всех ОВП рассадить за двумя столами.

Формат входных данных

В первой строке входных данных содержатся два числа: N и M ($1 \leq N, M \leq 100$), где N — количество ОВП, а M — количество пар ОВП, которые не могут сидеть за одним столом. В следующих M строках записано по 2 числа — пары ОВП, которые не могут сидеть за одним столом.

Формат выходных данных

Если способ рассадить ОВП существует, то выведите YES в первой строке и номера ОВП, которых необходимо посадить за первый стол, во второй строке. В противном случае в первой и единственной строке выведите NO.

Пример

banket.in	banket.out
3 2	YES
1 2	2 3
1 3	

Задача С. Предок

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 Мб

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leq n \leq 100\,000$) — количество вершин в дереве. Во второй строке находится n чисел, i -ое из которых определяет номер непосредственного родителя вершины с номером i . Если это число равно нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leq m \leq 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leq a, b \leq n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число 1, если вершина a является одним из предков вершины b , и 0 в противном случае.

Пример

ancestor.in	ancestor.out
6	0
0 1 1 2 3 3	1
5	1
4 1	0
1 4	0
3 6	
2 6	
6 5	

Задача D. TopSort

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 Мб

Дан ориентированный невзвешенный граф. Необходимо вывести его вершины в таком порядке, что каждое ребро графа ведет из вершины стоящей раньше в вершину, стоящую позже, или сообщить что это невозможно.

Формат входных данных

В первой строке входного файла два натуральных числа N и M ($1 \leq N \leq 10^3, 1 \leq M \leq 10^4$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задается парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходных данных

Вывести любую последовательности номеров вершин, удовлетворяющую условию. Если это невозможно, 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 Е. Обход в глубину

Ограничение по времени: 2 секунды

Ограничение по памяти: 256 Мб

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.

Паскаль	Си
<pre> var a: array [1..maxn, 1..maxn] of boolean; visited: array [1..maxn] of boolean; procedure dfs(v: integer); var i: integer; begin writeln(v); visited[v] := true; for i := 1 to n do begin if a[v][i] and not visited[i] then begin dfs(i); writeln(v); end; end; end; procedure graph_dfs; var i: integer; begin for i := 1 to n do if not visited[i] then dfs(i); end; </pre>	<pre> 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); } } } </pre>

Петина программа хранит граф с использованием матрицы смежности в массиве «а» (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 ($1 \leq n \leq 300$, $1 \leq l \leq 2n - 1$). Следующие l строк по одному числу — вывод Петинной 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	6
1	1 2
2	1 3
3	1 4
2	2 3
4	2 4
2	5 6
1	
5	
6	
5	