Задача A. Обход в глубину

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

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

```
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;
```

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 входного файла

Первая строка входного файла содержит два целых числа: n и l — количество вершин в графе и количество чисел в выведенной последовательности ($1 \le n \le 300$, $1 \le l \le 2n - 1$). 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 выходного файла

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 2 3 1 4 2 4 5 6

Задача B. 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 входного файла

В первой строке входного файла даны два целых числа N и M ($1 \leqslant N \leqslant 100\,000, 0 \leqslant M \leqslant 100\,000$) — количество вершин и рёбер в графе соот-

ветственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходного файла

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

Задача C. Предок

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

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

Формат входного файла

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — 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 ($1 \leqslant m \leqslant 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходного файла

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.

Примеры

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

Задача D. Авиаперелеты

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

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 входного файла

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 выходного файла

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

Примеры

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