Задача A. 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 2 3 3 4 4 1	1 2 2 3 3 4

Задача B. 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 ($1 \leqslant N, M \leqslant 100$), 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 1 2 1 3	YES 2 3

Задача C. 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 ($1 \leqslant n \leqslant 100\,000$) — 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 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 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Задача D. 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

```
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 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