'''

There are N rooms in a row, and all the rooms are locked initially.

You will be provided the master key. You can use the master key only once, and

you can unlock any one room and find a key in it of some other room.

You have to repeat this process until you can find the keys of the locked rooms.

You have to stop this process once you find the key of a room

which is already unlocked.

You are given an array of keys[], where i-th room contains the key of key[i]-th

room and values of keys[] are range from [0..N-1] without duplicates. Your task

is to find the maximum number of rooms can be unlocked including the first room.

Input Format:

-------------

Line-1: An integer N, number of rooms.

Line-2: N space separated integers, keys of rooms.

Output Format:

--------------

Print an integer result.

Sample Input-1:

---------------

8

7 4 6 2 1 0 3 5

Sample Output-1:

----------------

3

Explanation:

-----------

keys[0] = 7, keys[1] = 4, keys[2] = 6, keys[3] = 2, keys[4] = 1, keys[5] = 0,

keys[6] = 3, keys[7]=5.

You can start with Room-0 using the master key, you can open the following rooms:

key[0]=7 => key[7]=5 => key[5]=0

or

key[2]=6 => key[6]=3 => key[3]=2

You can unlock maximum 3 rooms.

Sample Input-2:

---------------

6

3 2 4 0 5 1

Sample Output-2:

----------------

4

Sample Input-3:

---------------

5

0 1 2 3 4

Sample Output-3:

----------------

1

----------------------------

Write your Python code below

----------------------------

'''

n=int(input())

l=list(map(int,input().split()))

d=dict()

ma=[]

# count=0

def fun(key,d,count):

if(d[key]==True or d[d[key]]==True):

return count

d[key]=True

return fun(l[key],d,count+1)

for i in l:

key=i

for j in range(0,n):

d[j]=False

count=fun(key,d,0)

ma.append(count)

print(max(ma))

Mr Suleman is given a checkerboard of size 400\*400, where the indices starts

from (-200,-200) and ends at (199,199). In one step, he can move the box from

position (p,q) to one of the following positions in L shape like as follows:

- (p-2, q-1), (p-2, q+1), (p+2, q-1), (p+2, q+1)

- (p-1, q+2), (p+1, q+2), (p-1, q-2), (p+1, q-2)

Initially the box is at (0,0) position, and need to move the box to position (m,n).

You will be given two integers m and n indicates the position(m,n).

Now your task is to help by Mr Suleman to find the minimum number of steps

required to move the box from (0,0) to (m,n).

Note: It is allowed to move out of the board also.

Input Format:

-----------------

Two space separated integers, m and n, position.

Output Format:

------------------

Print an integer, minimum number of steps to reach (m,n).

Sample Input-1:

---------------

2 4

Sample Output-1:

----------------

2

Explanation:

-------------

Initially, you are at (0,0) position, you can reach (2,4) as follows:

(0,0) -> (1, 2) -> (2, 4)

Sample Input-2:

---------------

4 7

Sample Output-2:

----------------

5

Explanation:

------------

Initially, you are at (0,0) position, you can reach (4,7) as follows:

(0,0) -> (1, 2) -> (2, 4) -> (1, 6) -> (3, 5) -> (4, 7)

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int r=sc.nextInt();

int c=sc.nextInt();

boolean[][] vis=new boolean[401][401];

System.out.println(fun(r,c,vis));

}

static int[][] arrs={{-2,-1},{-2,1},{2,-1},{2,1},{-1,2},{1,2},{-1,-2},{1,-2}};

public static int fun(int r,int c,boolean[][] vis){

vis[200][200]=true;

// visl[0][0]=true;

Queue<int[]> q=new LinkedList<>();

q.add(new int[]{0,0,0});

while(!q.isEmpty()){

int[] arr=q.remove();

int r1=arr[0];

int c1=arr[1];

// vis[r1][c1]=true;

// vis[c1]=true;

int count=arr[2];

if(r1==r && c1==c){

return count;

}

for(int i=0;i<8;i++){

int rr1=r1+arrs[i][0];

int cc1=c1+arrs[i][1];

if(rr1>=-200 && rr1<200 && cc1>=-200 && cc1<200 && !vis[rr1+200][cc1+200] ){

vis[rr1+200][cc1+200]=true;

q.add(new int[]{rr1,cc1,count+1});

}

}

}

return -1;

}

}

There are N cities, and M routes[], each route is a path between two cities.

routes[i] = [city1, city2], there is a travel route between city1 and city2.

Each city is numbered from 0 to N-1.

There are one or more Regions formed among N cities.

A Region is formed in such way that you can travel between any two cities

in the region that are connected directly and indirectly.

Your task is to findout the number of regions formed between N cities.

Input Format:

-------------

Line-1: Two space separated integers N and M, number of cities and routes

Next M lines: Two space separated integers city1, city2.

Output Format:

--------------

Print an integer, number of regions formed.

Sample Input-1:

---------------

5 4

0 1

0 2

1 2

3 4

Sample Output-1:

----------------

2

Sample Input-2:

---------------

5 6

0 1

0 2

2 3

1 2

1 4

2 4

Sample Output-2:

----------------

1

Note: Look HINT for explanation.

// dsu compressed path

// u do union and find and then u have to do set on union array to find the no of regions

import java.util.\*;

public class Main{

static class Edge{

int src;

int dest;

Edge(int src,int dest){

this.src=src;

this.dest=dest;

}

}

public static void create(ArrayList<Edge> graph[],int[][] arr){

for(int i=0;i<graph.length;i++){

graph[i]=new ArrayList<Edge>();

}

for(int i=0;i<arr.length;i++){

graph[arr[i][0]].add(new Edge(arr[i][0],arr[i][1]));

graph[arr[i][1]].add(new Edge(arr[i][1],arr[i][0]));

}

}

public static void dfs(ArrayList<Edge> graph[],boolean[] vis,int src){

vis[src]=true;

for(int i=0;i<graph[src].size();i++){

Edge e=graph[src].get(i);

if(!vis[e.dest]){

dfs(graph,vis,e.dest);

}

}

}

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int m=sc.nextInt();

int[][] arr=new int[m][2];

for(int i=0;i<m;i++){

arr[i][0]=sc.nextInt();

arr[i][1]=sc.nextInt();

}

ArrayList<Edge>[] graph=new ArrayList[n+1];

create(graph,arr);

int count=0;

boolean[] vis=new boolean[n];

for(int i=0;i<n;i++){

if(!vis[i]){

dfs(graph,vis,i);

count+=1;

}

}

System.out.println(count);

}

}

Ramu is a milk supplier to Somu. He has only two aluminium cups

of capacity m and n litres. Using these two cups only,

he can measure the milk. He has to supply exactly P litres of milk to Somu.

To measure the milk, operations allowed are:

- Fill one of the cups completely with milk.

- Empty the other cup.

- Pour milk from one cup into another cup till it fills completely,

or the first cup itself is empty.

Such that, at the end one or both cups contains P litres of the milk.

Please help Ramu, to check whether P litres of milk can be measured

using the two cups or not.

Input Format:

-------------

3 space separated integers, m, n and P.

Output Format:

--------------

Print a boolean value.

Sample Input-1:

---------------

3 5 4

Sample Output-1:

----------------

true

Sample Input-2:

---------------

2 4 5

Sample Output-2:

----------------

False

I stiil didn’t understand the code here

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int m=sc.nextInt();

int n=sc.nextInt();

int p=sc.nextInt();

System.out.println(fun(m,n,p));

}

static Set<String> hs=new HashSet<>();

public static boolean fun(int m,int n,int p){

if((m+n)<p || (m%2==0 && n%2==0 && p%2==1)){

return false;

}

return dfs(m,n,p,0);

}

public static boolean dfs(int m,int n,int p,int cur){

String s=m+","+n+","+cur;

if(hs.contains(s)){

return false;

}

hs.add(s);

if(cur==p){

return true;

}

if(cur>m+n || cur<0){

return false;

}

return (dfs(m,n,p,cur+m)|| dfs(m,n,p,cur+n)||dfs(m,n,p,cur-m)||dfs(m,n,p,cur-n));

}

}