

Незнайка и Коллатц. 11 класс

Незнайка очень полюбил функцию Коллатца $f(n) = \begin{cases} \frac{n}{2}, & n - \text{четное число} \\ 3n + 1, & n - \text{нечетное число} \end{cases}$, про которую

есть недоказанная гипотеза о том, что её применение снова и снова, стартуя с любого натурального ненулевого числа, всегда приводит к числу 1, после чего возникает цикл $1 \rightarrow 4 \rightarrow 2 \rightarrow 1$. Незнайке пришло в голову, что можно применять обратное преобразование, и от какого-то натурального ненулевого числа a переходить к одному из чисел b , такому что $f(b) = a$. Он обозначил такое обратное преобразование f^{-1} и стал искать способы перейти от стартового числа n_1 к финишному числу n_2 за минимально возможное количество шагов, применяя на каждом шаге либо функцию Коллатца f , либо обратное преобразование f^{-1} к текущему результату, полученному на предыдущем шаге (на первом шаге за текущий результат Незнайка брал число n_1). Например, от $n_1 = 10$ к $n_2 = 21$ можно перейти за 5 шагов: $10 \rightarrow f(10) = 5 \rightarrow f(5) = 16 \rightarrow f^{-1}(16) = 32 \rightarrow f^{-1}(32) = 64 \rightarrow f^{-1}(64) = 21$

Полагая, что гипотеза Коллатца верна, помогите Незнайке написать программу, которая определяет минимально возможное количество шагов, за которые из данного стартового числа n_1 можно получить финишное число n_2 . Пусть программа, если количество шагов не менее 2, выводит промежуточные числа, полученные на каждом шаге, кроме последнего.

Формат ввода: В первой строке вводится натуральное число n_1 : $1 \leq n_1 \leq 10000$. Во второй строке содержится натуральное число n_2 : $1 \leq n_2 \leq 10000$.

Формат вывода: В первой строке выводится либо наименьшее из возможных количество последовательных применений функции $f(x)$ или преобразования $f^{-1}(x)$, после которых стартовое число n_1 преобразуется в финишное число n_2 , либо -1, если стартовое число n_1 не может быть преобразовано таким способом в финишное число n_2 . Если $n_1 = n_2$, то выводится 0. Если в первой строке было выведено число, большее 1, то во второй строке выводятся промежуточные результаты – числа, полученные на каждом шаге, разделённые одиночным пробелом. В начале второй строки и в её конце пробел не ставится. Если в первой строке был выведен -1, 0 или 1, то вывод содержит лишь одну строку.

Ввод примера №1:	Ввод примера №2:	Ввод примера №3:	Ввод примера №4:
10	8	10000	2
21	1	10000	1
Вывод примера №1:	Вывод примера №2:	Вывод примера №3:	Вывод примера №4:
5	2	0	1
5 16 32 64	4		

Решение

Ставя каждому натуральному числу в соответствие вершину графа, и соединяя дугами все пары вершин с числами a и b , такими что $f(a) = b$, мы получаем, так называемый, граф Коллатца. Из любой вершины этого графа можно двигаться только в сторону 1, путь до которой существует, если гипотеза Коллатца верна. Известно, что в диапазоне рассматриваемых чисел длина такого пути не превосходит 261. В этом можно убедиться с помощью предварительных расчётов. Длину 261 имеет путь от 6171 к 1. Находим путь к 1, ведущий из n_1 . Находим путь к 1, ведущий из n_2 . Находим вершину, находящуюся дальше всего от 1 и лежащую на обоих путях. Часть пути от n_1 до этой вершины, к которой в приписана задом наперёд часть пути от n_2 до этой вершины, образует ответ. Сама вершина, которую мы нашли, входит в ответ не более чем один раз. Её не следует включать в ответ, если она совпадает с n_1 и/или с n_2 . При поиске пути к 1 следует иметь в виду, что $4 \rightarrow f^{-1}(4) = 1$ и это короче, чем $4 \rightarrow f(4) = 2 \rightarrow f(2) = 1$. Другими словами, если n_1 и/или n_2 находятся в диапазоне от 1 до 2, то такие случаи надо анализировать отдельно. Также следует иметь в виду, что ради эффективности перед поиском путей можно проверить тривиальные случаи, когда n_1 и n_2 совпадают или находятся в одном шаге друг от друга. Составляя программу, стоит иметь в виду, что промежуточные числа могут быть довольно велики. Так, на пути от 9663 к 1 лежит число 27114424.

Код возможного решения

```
program COLLATZ11(input, output);
const   PATH_LENGTH = 262;
type    massiv = array [1..PATH_LENGTH] of cardinal;
```

```

        path = record length : 0..PATH_LENGTH; numbers : massiv end;
function F(X : cardinal) : cardinal;
begin
    if (X mod 2) = 0 then F := X div 2 else F := 3 * X + 1
end;
procedure APPEND (V : cardinal; var P : PATH);
begin with P do begin
    length := length + 1;
    numbers[length] := V
end
end;
procedure PATHTO1 (N : cardinal; var P : path);
begin
    P.length := 0;
    while N > 1 do begin
        APPEND(N, P);
        N := F(N);
    end;
    APPEND(1, P)
end;

var N1, N2, V : cardinal;
    I, J, K, RESULT : cardinal;
    FLAG1, FLAG2 : boolean;
    P1, P2 : path;
begin FLAG1 := false;
    FLAG2 := false;
    readln(N1);
    read(N2);
    if N1=N2 then write(0) else begin
        if N1 > 1 then I := F(N1) else I := N1;
        if N2 > 1 then J := F(N2) else J := N2;
        if (I = N2) or (J = N1) or ((N1 = 4) and (N2 = 1)) or ((N1 = 1) and (N2 = 4)) then
            write(1)
        else begin
            PATHTO1(N1, P1);
            PATHTO1(N2, P2);
            I := P1.length;
            J := P2.length;
            while (I > 0) and (J > 0) and (P1.numbers[I] = P2.numbers[J]) do begin
                I := I - 1;
                J := J - 1;
            end;
            if (I >= 0) and (I < P1.length) then V := P1.numbers[I+1] else
            if (J >= 0) and (J < P2.length) then V := P2.numbers[J+1] else
            V := 1;
            RESULT := I + J;
            if ((N2 = 1) and (N1 > 2)) or ((N1 = 1) and (N2 > 2)) then begin
                RESULT := RESULT - 1;
                FLAG2 := true;
            end;
            writeln(RESULT);
            if I > 1 then begin
                write(P1.numbers[2]);
                for K := 3 to I do if not (FLAG2 and (P1.numbers[K] = 2)) then
                    write(' ', P1.numbers[K]);

```

```

    FLAG1 := true
end;
if (V > 1) and (V <> N1) and (V <> N2) then
    if not (FLAG2 and (V = 2)) then begin
        if FLAG1 then write(' ');
        write(V);
        FLAG1 := true
    end;
if J > 1 then begin
    if FLAG1 then if not (FLAG2 and (P2.numbers[J] = 2)) then write(' ');
    if not (FLAG2 and (P2.numbers[J] = 2)) then write(P2.numbers[J]);
    for K := J-1 downto 2 do begin
        if not (FLAG2 and (P2.numbers[K] <= 4)) then write(' ');
        if not (FLAG2 and (P2.numbers[K] = 2)) then write(P2.numbers[K])
    end;
end
end
end
end.

```