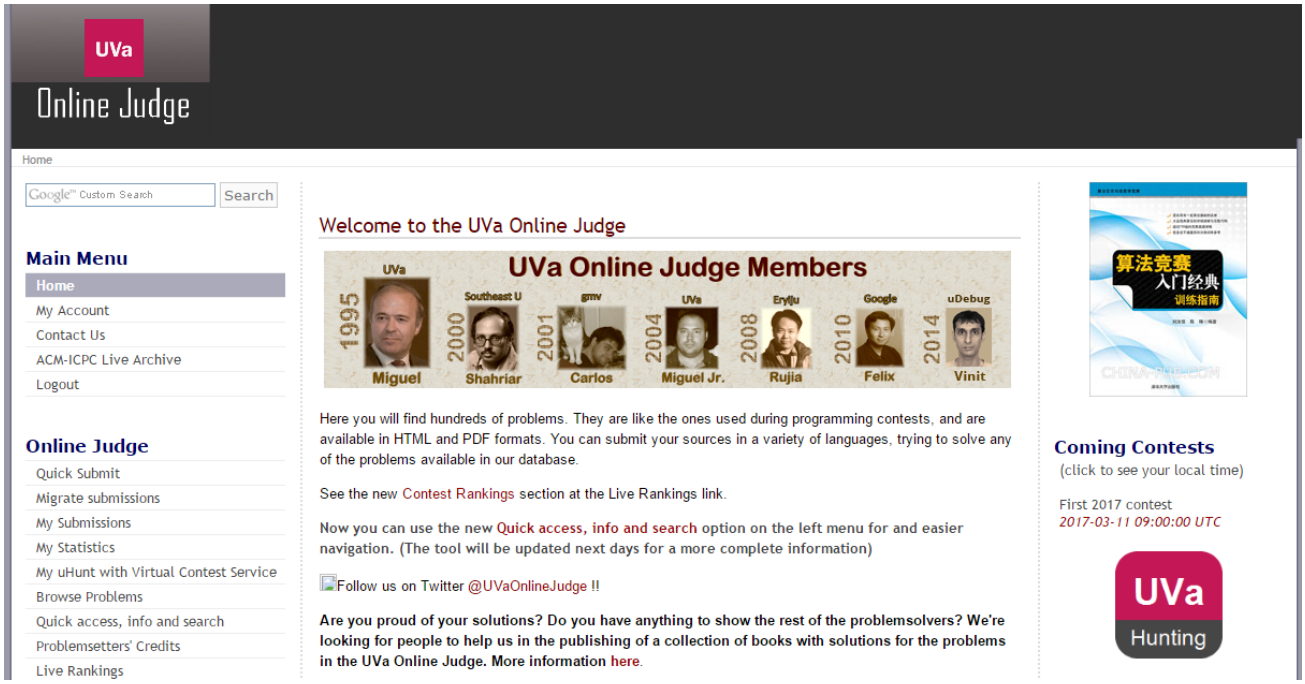
01

O problema do 3n mais 1

Transcrição

Vamos ver o terceiro problema? Ele está no UVa – Online Judge (https://uva.onlinejudge.org). Se cadastre para que façamos o desafio 36 (https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=36)



Seu enunciado começa assim:

> Problems in Computer Science are often classified as belonging to a certain class of problems (e.g., NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.

Ele fala que os problemas de informática costumam ser divididos em algumas classes (como, NP, não resolvíveis, recursivos). E que nesse problema você vai analisar uma propriedade de um algoritmo cuja classificação não é conhecida para todas as entradas possíveis. Ou seja: h'a alguns algoritmos que são de um desses tipos dependendo da entrada. É só um background, que pode ou não ter alguma informação escondida que ele espera que vejamos. E ele segue apresentando o algoritmo, que é o que analisaremos, implementaremos uma solução e a otimizaremos. Mas sua categoria ainda não foi matematicamente comprovada para todas as entradas.

> Consider the following algorithm:

```
1. input n
2. print n
3. if n = 1 then STOP
4. if n is odd then n ← 3n + 1
5. else n ← n/2
6. GOTO 2
```

Vamos ler o número `n` (e estou aqui presumindo que `n` é um número) e imprimi-lo. Se ele for 1, paramos por aí. Se não for e for ímpar (*odd*), multiplicaremos por 3 e somaremos 1. Se for par, dividiremos por 2. E então, basta voltar para o passo `2`. Já sabemos que uma das entradas que testaremos será `1`.

A seguir, ele mostra a sequência que será impressa por esse algoritmo para o número `22`:

> Given the input 22, the following sequence of numbers will be printed
>
> 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Ele imprime o `22`, o divide por 2 por ser par e volta ao passo 2. Imprime o `11`, e como não é um, segue. Ele multiplica por 3 e soma 1 por ser ímpar. Imprime o `34` ... E assim segue, até chegar ao `1`. No total, foram 16 números até chegar nesse `1` que para o algoritmo, contando o primeiro e o último.

> It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that 0 < n < 1, 000, 000 (and, in fact, for many more numbers than this.)

Entretanto, não há prova matemática de que o algoritmo sempre chegará no `1`. Pode ser que haja algum caso em que ele apenas cresça. Ou que gere números infinitos. Por enquanto, não sabemos. Mas, acredita-se que o algoritmo sempre acabará (ou seja: chegará no `1`) se a entrada for um número inteiro. Já se verificou que para todos os números entre `0 < n < 1.000.000` e muitos além desses(*inclusive* ou *exclusive*? lembre-se que sempre precisamos pensar nisso. Nesse caso, *exclusive*), o algoritmo acaba.

> Given an input n, it is possible to determine the number of numbers printed before and including the 1 is printed. For a given n this is called the cycle-length of n. In the example above, the cycle length of 22 is 16.

Então, o enunciado nos diz que é possível determinar quantos números são impressos antes de o algoritmo parar, incluindo o próprio `1`. E ele chama esse número de *cycle-lenght of n*, ou comprimento do ciclo do n. Para o `22`, esse número era `16`. E é esse número que o problema quer que encontremos.

> For any two numbers i and j you are to determine the maximum cycle length over all numbers between and including both i and j.

Para quaisquer dois números `i e j` você deve determinar o maior comprimento de ciclo, de todos os números entre esses dois. Então, ele fala do input.

> **Input** The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 10,000 and greater than 0. You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j. You can assume that no operation overflows a 32-bit integer.

Aqui vemos que o input será composto por vários pares de números inteiros `i e j`, um par por linha. Esse números serão menores que `1.000.000` e maiores que `0`. Devemos processar todos e devolver o maior comprimento de ciclo de todos inteiros entre os números `i e j`, inclusive `i e j`. Ele diz também que podemos assumir que nenhuma operação ultrapassará o número inteiro de 32 bits.

Quando fazemos qualquer soma, subtração, multiplicação ou divisão, temos que nos preocupar com o tamanho dos números. Se estou adicionando um produto de 5 reais ao carrinho, provavelmente não terei problemas. Mas se começamos a somar vários números, uma hora a soma pode estourar o limite definido. E quando esse número, não sabemos exatamente o que acontece. Então, temos que pensar no tipo de variável que usaremos para armazenar esses valores.

> Output For each pair of input integers i and j you should output i, j, and the maximum cycle length for integers between and including i and j. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers i and j must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

Então, o enunciado fala sobre a saída. Ela deve mostrar o par de inteiros, na mesma ordem da entrada, e o número do comprimento máximo de ciclo. Os números devem ser separados com pelo menos um espaço, e deve haver uma linha de saída para cada uma de entrada.

E ele nos dá exemplos de entrada e de saída.

```
Sample Input
1 10
100 200
201 210
900 1000

Sample Output
1 10 20
100 200 125
201 210 89
900 1000 174
```

Assim, de todos os números entre `1` e `10`, o que tem o maior comprimento de ciclo passa `20` vezes pelo algoritmo. Entre `100` e `200`, são `125` vezes. E assim por diante.

Esse é um problema com vários detalhes. Dê uma pensada nesse problema. O que você faria primeiro? Não precisa começar a implementar, mas mantenha o problema em mente. Em breve a gente começa a lidar com ele. Até lá!