# Juízes Eletrônicos

## Problemas resolvidos

Prof. Edson Alves

2018

Faculdade UnB Gama

# Sumário

# UVA 100 – The $3n + 1$ problem

# Problema

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.

Consider the following algorithm:

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

Given the input 22, the following sequence of numbers will be printed

$$22 \quad 11 \quad 34 \quad 17 \quad 52 \quad 26 \quad 13 \quad 40 \quad 20 \quad 10 \quad 5 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

# Problema

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.)

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.

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

# Entrada e saída

## 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.

## 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).

# Exemplo de entradas e saídas

### Exemplo de Entrada

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

### Exemplo de Saída

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

# Solução

- A solução para este problema consiste em quatro etapas
- A primeira é ler a entrada do problema
- A segunda é codificar uma rotina que computa o *cycle-length* de um inteiro positivo $n$
- A terceira é computar o *cycle-length* de todos os números entre $i$ e $j$
- Por fim, produzir a saída correta
- A segunda etapa é o cerne do problema, e efetivamente a implementação é a réplica exata do algoritmo dado no problema

# Solução WA

```cpp
#include <iostream>

using namespace std;

int cycle_length(int n)
{
    int ans = 1;

    while (n > 1)
    {
        if (n % 2)
            n = 3*n + 1;
        else
            n /= 2;

        ans++;
    }

    return ans;
}

```

# Solução WA

```
22 int main()
23 {
24     int i, j;
25 
26     while (cin >> i >> j)
27     {
28         int ans = 0;
29 
30         for (int n = i; n <= j; n++)
31             ans = max(ans, cycle_length(n));
32 
33         cout << i << " " << j << " " << ans << '\n';
34     }
35 
36     return 0;
37 }
```

# Solução correta

- A solução anterior, embora aparentemente correta, tem veredito WA!
- Importante notar que o problema não está na rotina que computa do *cycle-length*
- O erro advém de uma leitura sem o rigor devido da entrada: não há, na descrição da entrada, a garantia de que $i$ e $j$ estejam em ordem crescente!
- Por exemplo, a entrada `10 1 20` produziria uma saída errada!
- Para corrigir este problema é preciso garantir a ordem crescente dos limites no laço do intervalo
- Porém, é preciso preservar os valores de $i$ e $j$ para produzir a saída na ordem correta

# Solução AC

```cpp
#include <iostream>

using namespace std;

int cycle_length(int n)
{
    int ans = 1;

    while (n > 1)
    {
        if (n % 2)
            n = 3*n + 1;
        else
            n /= 2;

        ans++;
    }

    return ans;
}

```

# Solução AC

```
22 int main()
23 {
24     int i, j;
25 
26     while (cin >> i >> j)
27     {
28         int ans = 0, a = min(i, j), b = max(i, j);
29 
30         for (int n = a; n <= b; n++)
31             ans = max(ans, cycle_length(n));
32 
33         cout << i << " " << j << " " << ans << '\n';
34     }
35 
36     return 0;
37 }
```

# AtCoder Beginner Contest 101 – Problem C: Minimization

# Problema

There is a sequence of length $N$: $A_1, A_2, \ldots, A_N$. Initially, this sequence is a permutation of $1, 2, \ldots, N$.

On this sequence, Snuke can perform the following operation:

- Choose $K$ consecutive elements in the sequence. Then, replace the value of each chosen element with the minimum value among the chosen elements.

Snuke would like to make all the elements in this sequence equal by repeating the operation above some number of times. Find the minimum number of operations required. It can be proved that, under the constraints of this problem, this objective is always achievable.

# Entrada e saída

### Constraints

- $2 \leq K \leq N \leq 100000$
- $A_1, A_2, \ldots, A_N$ is a permutation of $1, 2, \ldots, N$

### Input

Input is given from Standard Input in the following format:

$$N \quad K$$
$$A_1 \quad A_2 \quad \ldots \quad A_N$$

### Output

Print the minimum number of operations required.

# Exemplo de entradas e saídas

| Exemplo de Entrada | Exemplo de Saída |
| --- | --- |
| 4 3<br>2 3 1 4 | 2 |
| 3 3<br>1 2 3 | 1 |
| 8 3<br>7 3 1 8 4 6 2 5 | 4 |

# Solução

- O primeiro fato a ser observado é que é possível modificar, a cada operação, no máximo $K - 1$ elementos
- O segundo fato importante é que o elemento mínimo da sequência é sempre o número um
- Aqui de ficar claro que o objetivo não é realizar as operações, mas sim contar o mínimo necessário para completar a transformação
- Do primeiro fato apresentado segue que são necessárias pelo menos

$$\left\lfloor \frac{N-1}{K-1} \right\rfloor$$

operações, pois é preciso modificar os $N - 1$ elementos restantes, um por vez

# Solução

- Observe também que a sequência de intervalos $I_0, I_1, \ldots, I_{\lfloor \frac{N-1}{K-1} - 1 \rfloor}$ cobre toda a sequência, onde

$$I_i = i(K-1)+1, i(K-1)+2, \ldots, i(K-1)+K$$

- Com esta divisão da sequência, localize o intervalo $I_k$ que contém o elemento 1
- Então aplique $k$ operações nos intervalos $I_k, I_{k-1}, \ldots, I_0$, nesta ordem
- Agora aplique a operação nos intervalos $I_{k+1}, I_{k+1}, \ldots, I_{\lfloor \frac{N-1}{K-1} - 1 \rfloor}$
- Logo, é possível finalizar o processo com $\left\lfloor \frac{N-1}{K-1} \right\rfloor$ operações

# Solução AC

```cpp
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(false);

    int N, K;
    cin >> N >> K;

    auto ans = (N - 1 + K - 2)/(K - 1);

    cout << ans << '\n';

    return 0;
}
```

# Referências

1. UVA 100 – The $3n + 1$ problem
2. AtCoder Beginner Contest 101 – Problem C: Minimization