Máximo comprimento das sequências de números granizo

Requested files: user.c, input.txt (Download)
Type of work: Individual work
Grade settings: Maximum grade: 10 **Hidden**
Run: Yes **Evaluate**: Yes **Evaluate just on submission**: Yes
Automatic grade: Yes

A sequência de números granizo é a sequência que começa com um número inteiro *n*. Se *n* é par, o seguinte número na sequência é resultado de dividir *n* por 2. Se n é ímpar, o seguinte número na sequência é resultado de multiplicar *n* por 3 e adicionar 1. O processo é repetido com o novo valor de *n*, terminando no momento em que *n*=1. Por exemplo, a seguinte sequência de números granizo serão gerados para *n*=22:

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

Para uma sequência de números granizo iniciado em n, o ciclo de comprimento de n é a quantidade de números gerados até 1 (incluindo o 1). No exemplo acima, o comprimento da sequência de número granizo iniciada em 22 é 16.

Escreva um algoritmo que, dados dois números *i* e *j*, determine o máximo ciclo de comprimento das sequências de números granizo iniciadas nos números pertencentes ao intervalo *i* e *j* (incluindo os números *i* e *j*). Por exemplo, dados os valores *i*=10 e *j*=13, o máximo ciclo de comprimento das sequências de números granizo iniciados em 10, 11, 12 e 13 será 15 como é mostrado abaixo.

- Para *n*=10, a sequência de números granizo é 10, 5, 16, 8, 4, 2, 1 com comprimento 8.
- **Para *n*=11, a sequência de números granizo é 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 com comprimento 15.**
- Para *n*=12, a sequência de números granizo é 12, 6, 3, 10, 5, 16, 8, 4, 2, 1 com comprimento 10
- Para *n*=13, a sequência de números granizo é 13, 40, 20, 10, 5, 16, 8, 4, 2 com comprimento 10.

Entrada e Saída:

A entrada será constituída por pares de números inteiros *i* e *j* separados por um espaço em branco. Todos os inteiros serão números menores que 10000 e maiores do que 0, assim como sempre *i* será menor que *j*. Cada linha no arquivo "input.txt" representará uma entrada para o programa. Como saída você deve imprimir três números: os valor de *i*, o valor de *j* e o máximo ciclo para todas as sequências de números granizos iniciadas com valores no intervalo *i* e *j*.

Exemplos de entrada	Saída para os exemplos de entrada
1 10	1 10 20
100 200	100 200 125
201 210	201 210 89
900 1000	900 1000 174

Requested files

user.c

```c
#include <stdio.h>

int main() {
          int i, j;
          scanf("%d %d",&i, &j);
          // escreva seu código aqui

          return 0;
}
```

input.txt

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

Execution files

vpl_run.sh

```
#! /bin/bash

cat > vpl_execution <<EEOOFF
#! /bin/bash
prog1="user"
prog2="test"
gcc \${prog1}.c -o \${prog1} -lm | grep -v Note > grepLines.out
gcc \${prog2}.c -o \${prog2} -lm | grep -v Note > grepLines.out
if [ -s grepLines.out ] ; then
     echo "Some compiler ERRORS reported"
     cat grepLines.out
     exit
fi

while IFS='' read -r line || [[ -n "\${line}" ]]; do
    if [[ !  -z  "\${line}"  ]]; then
        echo "\${line}" > in.txt
        echo "----------Your program----------"
        ./\${prog1} < in.txt
        echo "--------------------"
        echo "----------Test program----------"
        ./\${prog2} < in.txt
        echo "--------------------"
    fi
done < input.txt
EEOOFF

chmod +x vpl_execution
```

vpl_debug.sh

```

```

vpl_evaluate.sh

```
#!/bin/bash
cat >vpl_execution << 'EOF'
#!/bin/bash

    user="user"
    test="test"
    params_file="params.in"
    input_tests="tests.in"

    # > Compile the executable
    gcc $user.c -o $user -lm
    gcc $test.c -o $test -lm

    # > The params file format:
    # .. First line: number of tests;
    # .. Second line: number of inputs from each test
    typeset -i num_tests=$(head -n 1 $params_file)
    typeset -i num_input=$(tail -n 1 $params_file)
    declare -i successes=0

    # > Read every test ...
    for num in $(seq 0 $(($num_tests-1)));
    do
        > "in.txt"
        # ... param by param, composing an input file
        for input in $(seq 0 $(($num_input-1)));
        do
            typeset -i selected_line=$(((num*num_input)+input+1))
            cmd="$selected_line!d"
            sed $cmd $input_tests >> "in.txt" # get the selected input from file
        done

        # > Execute both user and test programs with the same input
        echo `./$user < in.txt` > ${user}_out
        echo `./$test < in.txt` > ${test}_out

        diff -y -w -B --ignore-all-space ${user}_out ${test}_out > diff.out
            # > Wrong answer
        if (($? > 0)); then
          echo "Comment :=>> Incorrect output found on test $num"
          echo "Comment :=>>- Your output"
          echo "<|--"
          cat ${user}_out
          echo "--|>"
          echo ""
          echo "Comment :=>>- Expected output "
          echo "<|--"
          cat ${test}_out
          echo "--|>"

            # > Right answer
        else
            successes=$((successes+1))
            #echo "Comment :=>> Correct output."
        fi
    done
    echo "----------------------------"
    echo "Comment :=>>- Your success rate is ${successes}/${num_tests}."
    echo "Grade :=>>$(((10*successes)/num_tests))"
    echo "----------------------------"

EOF

chmod +x vpl_execution
```

vpl_evaluate.cases

params.in

```
1 10
2 2
```

test.c

```c
#include <stdio.h>

int main() {
    int i, j;
    scanf("%d %d", &i, &j);
    int valorInicial, maxComprimento = 0;
    for (valorInicial = i ; valorInicial <= j ; valorInicial++ ) {
        int n = valorInicial;
        int comprimento = 1;
        while (n != 1) {
            if (n % 2 == 0) {
                n = n/2;
            } else {
                n = (n*3) + 1;
            }
            comprimento = comprimento + 1;
        }
        if (comprimento > maxComprimento) {
            maxComprimento = comprimento;
        }
    }
    printf("%d %d %d\n", i, j, maxComprimento);
    return 0;
}
```

tests.in

```
1 1
2 100
3 101
4 200
5 201
6 301
7 302
8 400
9 400
10 500
11 600
12 700
13 800
14 900
15 900
16 1000
17 1000
18 2000
19 2000
20 3000
```

Moodle Docs for this page

You are logged in as Admin User (Log out)
Introdução à Ciência de Computação - 2016