

Implementando o ciclo

Transcrição

Já sabemos que o cálculo funciona para números individuais, ou ao menos parece funcionar para os testes que passamos. Precisamos calcular tudo entre `i` e `j` e imprimir o maior deles. Para isso, faremos um `for()`. Ele irá do `int i = i` até... Fica meio feio ficar usando `i`, não? É um nome feio para a variável. Colocaremos `atual`.

```
public static void main(String[] args){

    calculaPara(1);
    calculaPara(22);
    calculaPara(999999);

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
        int i = scanner.nextInt();
        int j = scanner.nextInt();
        for(int atual = i; atual < j; atual++){

        }
        System.out.println(i + " " + j + " xxxxxxxx");
    }

}

private static void calculaPara(int i){
    int impressos = 1;
    //System.out.println(n);
    while (n != 1) {
        if(n % 2 == 1) n = n * 3 + 1;
        else n = n / 2;
        impressos++;
        //System.out.println(n);
    }
    System.out.println("Total " + impressos);
}
```

Estamos começando o `i` e indo até o `j`. No enunciado ele fala:

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`.

Os número `i` e `j` devem estar inclusos. Lembra que sempre que fazemos uma condição, precisamos verificar se devemos ou não incluir os pontos das condições iniciais e finais. Assim, devemos usar `<=`.

```
public static void main(String[] args){

    calculaPara(1);
```

```

calculaPara(22);
calculaPara(999999);

Scanner scanner = new Scanner(System.in);
while(scanner.hasNextLine()){
    int i = scanner.nextInt();
    int j = scanner.nextInt();
    for(int atual = i; atual <= j; atual++){

    }
    System.out.println(i + " " + j + " xxxxxx");
}

...
}

```

Vamos dar nomes de verdade para as variáveis? Podemos chamar `i` de `menor` e `j` de `maior`. Note que são nomes tendenciosos, que podem nos induzir a erro. O que queremos agora é calcular para o `atual`, ou seja:

`calculaPara(atual)`.

```

public static void main(String[] args){

    //calculaPara(1);
    //calculaPara(22);
    //calculaPara(999999);

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
        int menor = scanner.nextInt();
        int maior = scanner.nextInt();
        for(int atual = menor; atual <= maior; atual++){
            int resultado = calculaPara(atual);

        }
        System.out.println(i + " " + j + " xxxxxx");
    }

    private static void calculaPara(int i){
        int impressos = 1;
        //System.out.println(n);
        while (n != 1) {
            if(n % 2 == 1) n = n * 3 + 1;
            else n = n / 2;
            impressos++;
            //System.out.println(n);
        }
        System.out.println("Total " + impressos);
    }
}

```

O método `calculaPara` precisa retornar a quantidade de números impressos. Daremos um `Ctrl + 1` para que o retorno seja um `int`, não `void`.

```

public static void main(String[] args){

    //calculaPara(1);
    //calculaPara(22);
    //calculaPara(999999);

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
        int menor = scanner.nextInt();
        int maior = scanner.nextInt();
        for(int atual = menor; atual <= maior; atual++){
            int resultado = calculaPara(atual);

        }
        System.out.println(i + " " + j + " xxxxxxxx");
    }

    private static int calculaPara(int i){
        int impressos = 1;
        //System.out.println(n);
        while (n != 1) {
            if(n % 2 == 1) n = n * 3 + 1;
            else n = n / 2;
            impressos++
            //System.out.println(n);
        }
        return impressos;
    }
}

```

Agora ele pode nos devolver o resultado direitinho. Mas ainda estamos calculando todos os resultados individualmente. E não queremos cada um deles, queremos apenas o maior. Podemos selecionar esse número de diversas maneiras. Vamos armazenar um maior resultado `maiorCicloAteAgora`, que por enquanto tem tamanho `0`. Se o resultado foi maior (`>`) que o maior ciclo, ele se torna o `maiorCicloAteAgora`.

```

public static void main(String[] args){

    //calculaPara(1);
    //calculaPara(22);
    //calculaPara(999999);

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
        int menor = scanner.nextInt();
        int maior = scanner.nextInt();

        int maiorCicloAteAgora = 0;
        for(int atual = menor; atual <= maior; atual++){
            int resultado = calculaPara(atual);
            if(resultado > maiorCicloAteAgora){
                maiorCicloAteAgora = resultado;
            }
        }
    }
}

```

```

        System.out.println(i + " " + j + " xxxxxxxx");
    }

    private static int calculaPara(int i){
        int impressos = 1;
        //System.out.println(n);
        while (n != 1) {
            if(n % 2 == 1) n = n * 3 + 1;
            else n = n / 2;
            impressos++;
            //System.out.println(n);
        }
        return impressos;
    }
}

```

Mas qual é o menor ciclo possível? É quando ele imprime o `n` e para. Ou seja, imprime ao menos uma vez. Assim, o menor ciclo possível é `1`. Repare que, ao decidir o menor e o maior número, temos que decidir as condições iniciais. E para isso, é preciso pensar na condição inicial que realmente nos serve.

No nosso `sysout`, já podemos colocar o terceiro número: `maiorCicloAteAgora`.

```

public static void main(String[] args){

    //calculaPara(1);
    //calculaPara(22);
    //calculaPara(999999);

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
        int menor = scanner.nextInt();
        int maior = scanner.nextInt();

        int maiorCicloAteAgora = 1;
        for(int atual = menor; atual <= maior; atual++){
            int resultado = calculaPara(atual);
            if(resultado > maiorCicloAteAgora){
                maiorCicloAteAgora = resultado;
            }
        }

        System.out.println(i + " " + j + maiorCicloAteAgora);
    }

    private static int calculaPara(int i){
        int impressos = 1;
        //System.out.println(n);
        while (n != 1) {
            if(n % 2 == 1) n = n * 3 + 1;
            else n = n / 2;
            impressos++;
            //System.out.println(n);
        }
        return impressos;
    }
}

```

Vamos testar no terminal?

```
Alura-Azul:bin alura$ java Main < entrada2.txt
1 10 20
100 200 125
201 210 89
900 1000 89
```

Os valores estão batendo com os apresentados no enunciado, mas a entrada 2 tem mais valores. O que está acontecendo? O programa ainda não terminou de rodar! Ele está demorando para fazer a conta entre 999000 e 999999. Se tentarmos submeter esse código para o juiz, será que ele vai passar? Provavelmente não, porque o algoritmo está lento demais.

Mas vamos tentar. Lá no site do UVa, há o botão `Submit`. É preciso estar logado para fazer a submissão.

The screenshot shows the Online Judge website. At the top, there's a navigation bar with 'Home' and 'Browse Problems'. Below this, there's a search bar and a 'Main Menu' with links like 'Home', 'My Account', 'Contact Us', 'ACM-ICPC Live Archive', and 'Logout'. The 'Online Judge' section has links for 'Quick Submit', 'Migrate submissions', 'My Submissions', 'My Statistics', 'My uHunt with Virtual Contest Service', 'Browse Problems', 'Quick access, info and search', 'Problemsetters' Credits', 'Live Rankings', 'Site Statistics', 'Contests', 'Electronic Board', 'Additional Information', and 'Other Links'. The main content area displays the problem '100 - The 3n + 1 problem' with a time limit of 3.000 seconds. The problem description states: '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 ← 3n + 1'. On the right side, there are buttons for 'Submit', 'Statistics', 'Debug', and 'PDF'. The 'Submit' button is highlighted with a red circle.

Ao clicar nele, chegaremos na página a seguir. Basta selecionar a linguagem Java dentre as disponíveis e colar nosso código no campo.

This screenshot shows the same Online Judge interface, but with the 'Submit' button clicked. The 'Language' dropdown is set to 'JAVA 1.8.0 - OpenJDK Java'. The code editor contains the following Java code:

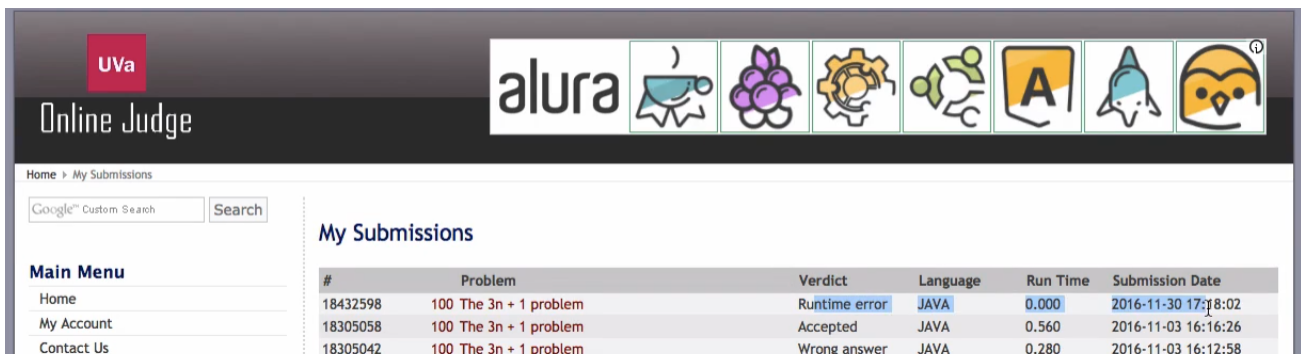
```
public static void main(String[] args){
    //calculaPara(1);
    //calculaPara(22);
    //calculaPara(999999);

    Scanner scanner = new Scanner(System.in);
    while(scanner.hasNextLine()){
        int menor = scanner.nextInt();
        int maior = scanner.nextInt();

        int maiorCicloAteAgora = 1;
        for(int atual = menor; atual <= maior; atual++){
            int resultado = calculaPara(atual);
            if(resultado > maiorCicloAteAgora){
                maiorCicloAteAgora = resultado;
            }
        }
    }
}
```

Below the code editor, there's a button 'Escolher arquivo' and a text 'Nenhum arquivo selecionado'. At the bottom, there are 'Submit' and 'Reset form' buttons.

Para ver o resultado, devemos clicar em `My Submissions`, no menu lateral.



The screenshot shows the 'Online Judge' interface for Alura. At the top, there's a navigation bar with the Alura logo and several icons. Below the navigation bar, the page title is 'Online Judge'. On the left, there's a 'Main Menu' with links to 'Home', 'My Account', and 'Contact Us'. The main content area is titled 'My Submissions' and contains a table with submission details.

#	Problem	Verdict	Language	Run Time	Submission Date
18432598	100 The 3n + 1 problem	Runtime error	JAVA	0.000	2016-11-30 17:18:02
18305058	100 The 3n + 1 problem	Accepted	JAVA	0.560	2016-11-03 16:16:26
18305042	100 The 3n + 1 problem	Wrong answer	JAVA	0.280	2016-11-03 16:12:58

Ele aponta um erro: `Runtime error`. Então não está apenas muito lerdo para nós, como também dá erro. É pior ainda, porque não estamos sequer conseguindo rodar o nosso programa para um caso válido. Não é um caso que o cliente passou, mas que nós pudemos criar a partir das condições que ele passou.

Nós paramos para pensar as entradas que eram possíveis. E é esse o nosso trabalho, pensar a mais e entender os negócios dos cliente para poder trabalhar com eles. Nosso trabalho não é apenas mecânico; ele é criativo também. Assim como o do cliente, que é criativo ao inventar soluções de processo. Nós temos que entender os processos, modelar, compreender as dificuldades que teremos (como entradas complicadas), para trabalhar junto ao cliente.

Para o nosso exemplo, está demorando demais. Para algum dos testes que o cliente está fazendo, está dando erro. Vamos tentar resolver os dois casos? Dê uma olhadinha no nosso código e tente imaginar onde está o erro e o porquê de ele estar tão lento. Em breve resolveremos os dois problemas. Até lá!