04

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 + " xxxxxxx");
    }

}

  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 + " xxxxxxx");
      }

...
}
```

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 + " xxxxxxx");
      }

    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 + " xxxxxxx");
    }

  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 + " xxxxxxx");
    }

  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`.

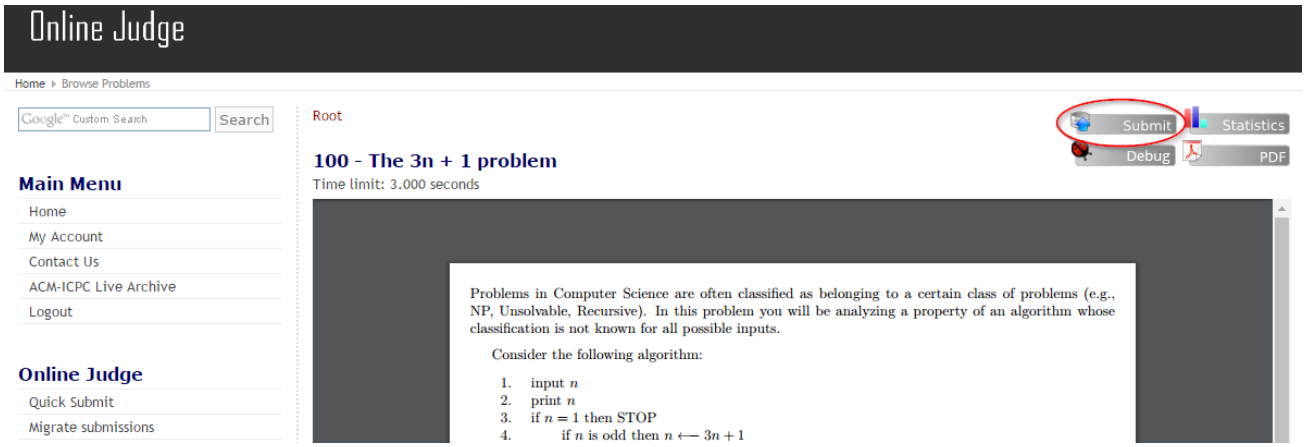
```
  public static void main(String[] args){

    //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?

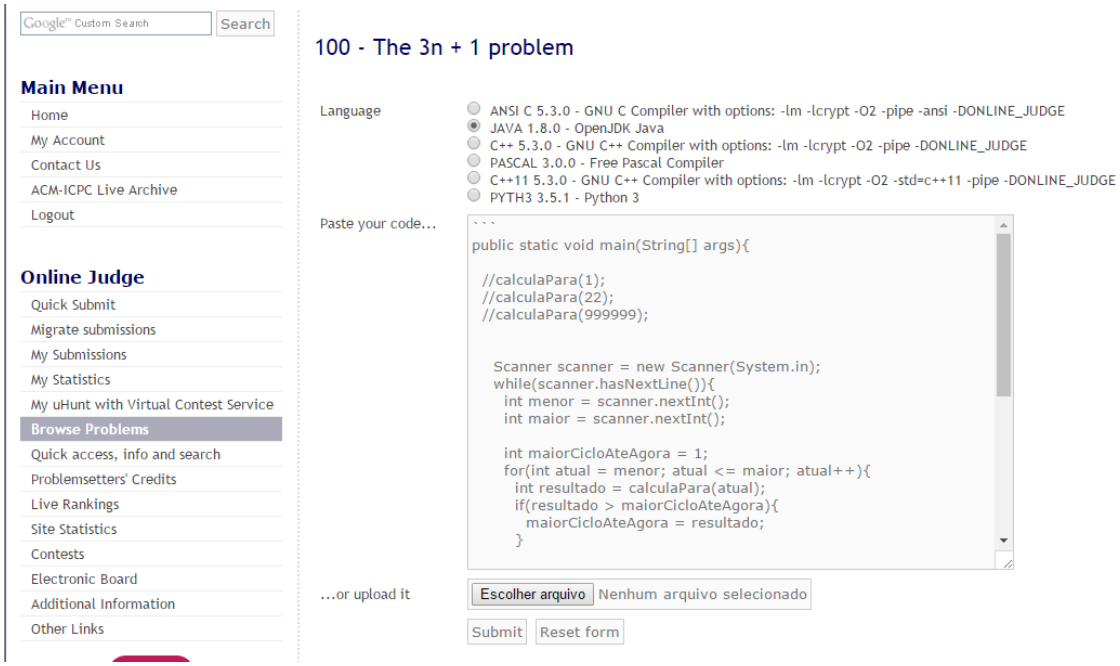
```
Alura-Azul:bin alura$ java Main < entrada2.txt
1 10 20
100 200 125
201 210 89
900 1000 89
```

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.

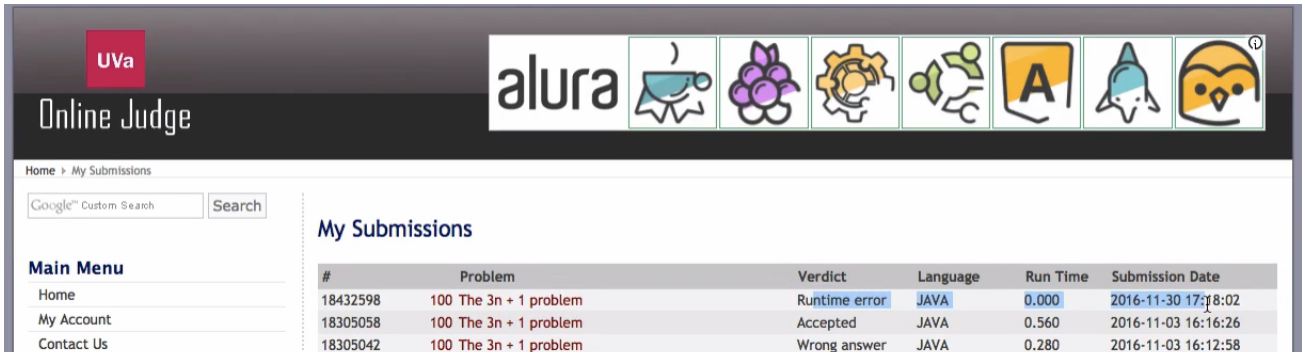
Mas vamos tentar. Lá no site do UVa, há o botão `Submit`. É preciso estar logado para fazer a submissão.



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



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



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!