03

# Resolvendo palíndromos

### Transcrição

O próximo problema é o PALINCOD, de palíndromos, que sabemos não ser possível de resolver no limite exigido de 250 caracteres. Vamos resolver como eu faria a despeito dessa exigência, começando pela leitura do enunciado.

> A palindrome is a word, phrase, number, or other sequence of units that can be read the same way in either direction, with general allowances for adjustments to punctuatiom and word dividers for eg., HELLOLLEH is a Palindrome string, and 123456789 is not a Palindrome number.

Ele explica o que é palíndromo, e diz para tomarmos cuidado com pontuações e divisores de palavra. Então nos dá um exemplo de palíndromo e um exemplo que não é palíndromo.

> Input t - number of testcases [ t < 1000 ]. On each of the next t lines given string.
>
> Output For each next t lines output the dataset number as a decimal integer (start counting at one), a space and "YES" if the given input is palindrome, or "NO" if the input is not.

Teremos que ler `t` número de casos, e cada uma das linhas seguintes terá uma palavra. E teremos que imprimir, para cada uma delas, o número do caso e a palavra `"YES"` ou `"NO"`, entre aspas.

Lembro novamente que não vamos nos preocupar com o número de caracteres, pois em uma maratona normal de programação isso não importa, muito menos no mundo real. Isso só existe nesse site, que está pensando que você pode tornar as pessoas genias ao forçá-las a escrever poucos caracteres. O que não tem nada a ver. Tanto que não existe lugar no mundo que paga para você digitar 10 caracteres a menos. Você é pago para resolver um problema e manter esse problema resolvido por muito tempo. Esse é o lugar em que a gente está.

Vamos para o Eclipse. Criaremos um projeto chamado `maratona-palincod`, e a classe `Main`. Como sempre, começamos com o método `main`.

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

  }
}
```

Podemos ir criando a `entrada.txt` com o exemplo que o enunciado nos dá.

```
2
HElloolLEH
ILOVEyou
```

Já de sacanagem vou acrescentar mais um palíndromo no final, lembrando que o programa só pode ler os dois primeiros casos pois o número que começa a entrada é `2`.

```
2
HELloolLEH
ILOVEyou
HELloolLEH
```

Começaremos por um `Scanner`.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

  }
}
```

A seguir, temos que ler um número, depois linhas. Quando dizemos "linhas", estamos dizendo que pode haver espaços nelas. Para garantir que funcionará, vamos já acrescentar casos complicadinhos. Colocaremos um caso com número ímpar de caracteres ( `HELlolLEH` ), um em que só o *case* é diferente ( `HELlOolLEH` ), um em que os `o` s serão substituídos por espaços ( `HEL l lLEH` ), que deve funcionar. Outro caso interessante é um que tenha apenas um espaço( `HEL l lLEH` ), tornando o número de caracteres ímpar. Note que são várias combinações das condições, e exceto pela linha com o *case*, todos devem dar `true`.

```
2
HELloolLEH
HELlolLEH
HELlOolLEH
HEL l  lLEH
HEL l lLEH
ILOVEyou
HELloolLEH
```

Como temos que ler linhas, já começaremos pedindo a leitura de linhas. É um pouco diferente do que fizemos antes.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    while(sc.hasNextLine()){
      String line = nextLine();
    }

  }
}
```

Opa! Estamos esquecendo que a primeira coisa a ser lida é o número. Ele é uma string, que transformaremos em `integer`.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    while(sc.hasNextLine()){
```

```
      String line = nextLine();
    }

  }
}
```

O que acabamos escrever é o número de casos. Vamos fazer um `for()`, que crie o número que imprimiremos (portanto, um `sysout` também é necessário) antes do diagnóstico do palíndromo, e apagaremos o `while()`.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    for(int i=0;i<numero;i++) {
      System.out.println(i + " ");
    }
    }

  }
}
```

Vamos testar no terminal, porque parece que há algo errado nas condições do `for()`. Depois de entrar na pasta correta:

```
Alura-Azul:bin alura$ java Main <entrada.txt
0
1
```

Faz sentido essa saída? Não, era para ser `1` e `2`. Assim, `i=1`, e `i<=numero`.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    for(int i=1;i<=numero;i++) {
      System.out.println(i + " ");
    }
    }

  }
}
```

Testando novamente:

```
Alura-Azul:bin alura$ java Main <entrada.txt
0
1
Alura-Azul:bin alura$ java Main <entrada.txt
1
2
```

Ótimo! Podemos prosseguir, agora sim lendo as linhas. As chamaremos de `frase`, e pediremos um `sysout` delas, para ter certeza de que estamos lendo corretamente.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    for(int i=1;i<=numero;i++) {
      System.out.println(i + " ");
      String frase = sc.nextLine;
      System.out.prinln(frase);
    }
    }

  }
}
```

No terminal, temos:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
HELloolLEH
2
HELlolLEH
```

Ele leu apenas os dois primeiros, o que está certo. Vamos apenas mexer na ordem da entrada e deixar o segundo exemplo que o cliente nos deu no começo, e depois testamos os outros.

```
2
HELloolLEH
ILOVEyou
HELlolLEH
HELlOolLEH
HELl  lLEH
HELl lLEH
HELloolLEH
```

Testando esse arquivo atualizado:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
HELloolLEH
2
ILOVEyou
```

A leitura está certa, e agora só precisamos fazer o algoritmo. Como eu conheço o Java, sei que `StringBuilder` ajuda a trabalhar com strings, e que tem um método que inverte, o `reverse`.

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

```
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    for(int i=1;i<=numero;i++) {
      System.out.println(i + " ");

      String frase = sc.nextLine;
      String invertido = new StringBuilder(frase).reverse().toString();


      System.out.prinln(frase);
    }
    }

  }
}
```

Note que mesmo em uma maratona eu estou extraindo, para manter tudo muito claro. Sabemos que se uma `frase` for igual a `invertido`, é palíndromo e devemos imprimir `"\"YES\""`. Caso contrário (`else`), devemos imprimir `"\"NO\""`. Vou deixar como comentário a impressão da frase, agora que sabemos que a leitura está correta.

```
public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    for(int i=1;i<=numero;i++) {
      System.out.println(i + " ");

      String frase = sc.nextLine;
      String invertido = new StringBuilder(frase).reverse().toString();
      if(frase.equals(invertido)){
        System.out.println("\"YES\"");
      } else {
        System.out.println("\"NO\"");
      }

//      System.out.prinln(frase);
    }
    }
  }
}
```

Vamos testar no terminal?

```
Alura-Azul:bin alura$ java Main <entrada.txt
1
"YES"
2
"NO"
```

Tem uma quebra de linha que não deveria haver. Usamos `println`, basta trocar por `print`.

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

```
    Scanner sc = new Scanner(System.in);
    int numero = Integer.parseInt(sc.nextLine());
    for(int i=1;i<=numero;i++) {
      System.out.println(i + " ");

      String frase = sc.nextLine;
      String invertido = new StringBuilder(frase).reverse().toString();
      if(frase.equals(invertido)){
        System.out.print("\"YES\"");
      } else {
        System.out.print("\"NO\"");
      }

//      System.out.prinln(frase);
    }
    }
  }
}
```

Testando no terminal, temos:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1 "YES"
2 "NO"
```

Está certinho, mas será que tem espaço a mais? Parece que não, mas podemos conferir, criando um arquivo com a saída exata que esperamos. Chamaremos esse arquivo de `saida.txt`, e nele copiaremos a saída que o enunciado nos dá, acrescentando um último `enter`.

```
1 "YES"
2 "NO"
```

Agora rodaremos no terminal comparando ( `diff` ) com o arquivo `saida.txt`.

```
Alura-Azul:bin alura$ java Main <entrada.txt | diff - saida.txt
Alura-Azul:bin alura$
```

Ele não encontrou diferenças. Agora podemos rodar com as outras palavras que criamos. Para isso, mudaremos o número no começo da `entrada.txt` para `7`, abarcando todos os casos.

```
7
HELloolLEH
ILOVEyou
HELlolLEH
HELlOolLEH
HELl  lLEH
HELl lLEH
HELloolLEH
```

No terminal:

```
Alura-Azul:bin alura$ java Main <entrada.txt
1 "YES"
2 "NO"
3 "YES"
4 "NO"
5 "YES"
6 "YES"
7 "YES"
```

Analisando caso a caso: sabemos que o primeiro e o segundo estão corretos. O terceiro ( `HEllolLEH` ) é palíndromo com número ímpar de letras, o quarto ( `HEllOolLEH` ) não é palíndromo, por causa do *case* diferente. O quinto( `HEll lLEH` ), sexto ( `HEll lLEH` ) e sétimo ( `HElloolLEH` ) também são. Inclusive, o sétimo é igual ao primeiro. Podemos aproveitar para colocar um caso com espaço no começo e no final.

```
8
HElloolLEH
ILOVEyou
HEllolLEH
HEllOolLEH
HEll  lLEH
HEll lLEH
HElloolLEH
 HElloolLEH
```

No terminal:

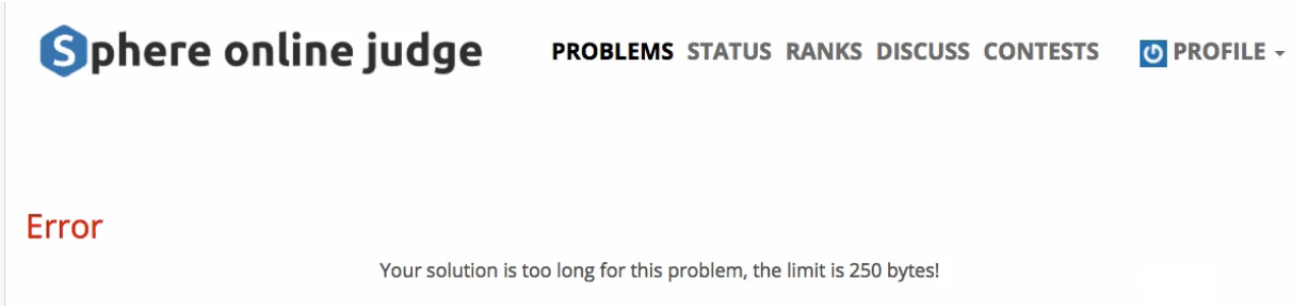
```
Alura-Azul:bin alura$ java Main <entrada.txt
1 "YES"
2 "NO"
3 "YES"
4 "NO"
5 "YES"
6 "YES"
7 "YES"
8 "YES"
```

Também é! Estamos acertando então. Não temos como confirmar com o `diff` porque não temos um parâmetro de referência para os casos que inventamos. Mas podemos repetir para a `saida.txt` .

```
Alura-Azul:bin alura$ java Main <entrada.txt | diff - saida.txt
< 3 "YES"
< 4 "NO"
< 5 "YES"
< 6 "YES"
< 7 "YES"
< 8 "YES"
```

Saiu o resultado esperado. Já comparamos as demais entradas no olho, e embora haja um risco em fazer dessa maneira, vamos mandar para o cliente. Na maratona trabalhamos um pouco no desespero.

Ao submeter, temos o seguinte resultado:




Ele me lembra que `250 bytes` é o máximo para esse problema. E já sabemos que não é possível cumprir essa condição em Java. Se você conseguir, mande para nós que ficaremos felizes.

Lembro novamente que problema nenhum da vida real se importa com o número de caracteres que você digita. Evite digitar caracteres com critérios doidos, como o desse problema. Mesmo nas maratonas e campeonatos mundiais não se pede esse tipo de coisa. Eles estão interessados em saber se você é bom em entender e resolver um problema, bem como em entender a cabeça de um cliente e resolver problemas que ele nem sabia que tinha, ou que não explicitou. Eles querem ver se somos capazes de resolvê-los de maneira útil, não se conseguimos resolver digitando pouco.

As competições não são de digitar menos. São de resolver problemas. Não somos digitadores, somos programadores. Esse está resolvido, apesar da condição. Assim, podemos ir para o próximo. Até lá!