

Programando em Java

(Strings e Vectores de Caracteres)

Mestrado Integrado em Engenharia Informática FCT UNL

<http://ctp.di.fct.unl.pt/miei/ip/>

Corpo Docente 2020/2021

António Ravara, Artur Miguel Dias, Bernardo Toninho,
Ema Vieira, Inês Fernandes, Margarida Mamede,
Miguel Monteiro, Rui Nóbrega

Rega Gota-a-Gota (adaptado do ToPAS)



Para cada cultura, conhece-se o **valor mínimo** e o **valor máximo** da quantidade de água que deve ser fornecida diariamente a cada planta. A quantidade de água é medida em número de gotas.

Uma planta é **feliz** num dia se receber gotas em número suficiente ($\geq$ valor mínimo), mas não excessivo ($\leq$ valor máximo).

O sistema de rega gota-a-gota regista diariamente uma sequência não vazia de zeros e uns, que indica se não houve ou se houve a rega de uma gota num dado minuto. O comprimento dessas sequências varia entre 1 e 1440 (24 horas vezes 60 minutos).

Por exemplo, a sequência `110111101111101111000000001` indica que a quantidade de água regada nesse dia foi 16 gotas (porque há 16 uns).

Rega Gota-a-Gota

Um agricultor quer trocar algumas plantas de uma cultura por plantas de outra cultura, sem alterar a programação do sistema de rega, cujo controlador se avariou. Ajude-o a tomar a melhor opção.



Escreva um programa que, dados:

- o nome, o valor mínimo e o valor máximo de duas culturas e
- uma sequência de registos diários da rega numa localização,

indique:

- em quantos dias as plantas de qualquer uma das duas culturas seriam felizes (nesses dias, a quantidade de água foi adequada para ambas as culturas);
- qual a cultura que seria mais feliz (aquela que seria feliz em mais dias) ou se as culturas seriam igualmente felizes (o número de dias felizes seria igual).

Rega Gota-a-Gota

Forma do Input

<i>nomeCultura_1</i>	Cadeia de caracteres de comprimento em [1, 20]
<i>minCultura_1</i>	Inteiro em [1, 1000]
<i>maxCultura_1</i>	Inteiro em [<i>minCultura_1</i> , 1000]
<i>nomeCultura_2</i>	Cadeia de caracteres de comprimento em [1, 20]
<i>minCultura_2</i>	Inteiro em [1, 1000]
<i>maxCultura_2</i>	Inteiro em [<i>minCultura_2</i> , 1000]
<i>numRegistos</i>	Inteiro em [1, 300]
<i>registo_1</i>	Cada registo é uma sequência de zeros e uns de comprimento em [1, 1440]
<i>registo_2</i>	
.....	
<i>registo_numRegistos</i>	



Rega Gota-a-Gota



Forma do Output

- A primeira linha tem:
“**D** dias adequados para ambas”,
onde **D** é o número de dias em que a quantidade de água foi adequada para ambas as culturas.
- A segunda linha tem uma das duas formas seguintes:
 - Se alguma cultura seria feliz em mais dias do que a outra, a 2ª linha tem:
“Opte por **N**”,
onde **N** é o nome da cultura que seria mais feliz.
 - Se as culturas seriam felizes no mesmo número de dias, a 2ª linha tem:
“Opcoes equivalentes”.

Rega Gota-a-Gota

Input 1

```
tomate
5
50
salsa
3
6
6
1110010101
1100001
11111111111
010111
0110
011111
```

Output 1

```
2 dias adequados para ambas
Opte por salsa
```



Número de gotas em cada dia: 6, 3, 11, 4, 2, 5

Dias adequados para tomate: 1º, 3º, 6º

Dias adequados para salsa: 1º, 2º, 4º, 6º

Rega Gota-a-Gota

Input 2

tomate
5
50
salsa
3
6
6
1110010101
1100001
11111111111
010
0000
011111

Output 2

2 dias adequados para ambas
Opcoes equivalentes



Número de gotas em cada dia: 6, 3, 11, 1, 0, 5

Dias adequados para tomate: 1º, 3º, 6º

Dias adequados para salsa: 1º, 2º, 6º

Strings e Vetores de Caracteres

Uma String tem uma variável de instância que é um vector de caracteres. Podem-se consultar, mas não se podem alterar os caracteres de uma string.

Alguns métodos muito usados da classe String:

- **int** length() devolve o comprimento da string
- **char** charAt(**int** pos) devolve o carácter na posição `pos` da string
// Pre: 0 <= pos && pos < this.length()
- **char**[] toCharArray() ←
devolve um **novo** vector com os caracteres da string

Nos dois casos,
alterações posteriores
no vector
não alteram a string

Para criar uma **nova** string com (todos) os caracteres de um vector de caracteres, há um construtor na classe String:

- String(**char**[] vec) ←
// Pre: vec != null

Strings e Vetores de Caracteres

Exemplos:

```
String str = "007, o agente secreto";
```

O valor da expressão `str.length()` é 21.

O valor da expressão `str.charAt(2)` é o carácter '7' (não é o número 7).

```
char[] vec = str.toCharArray();
```

O valor da expressão `vec.length` é 21.

O valor da expressão `vec[3]` é o carácter ','.

```
char[] letras = new char[3];
```

```
letras[0] = 'a';  letras[1] = 'b';  letras[2] = 'c';
```

```
String strLetras = new String(letras);
```

```
letras[0] = 'x';  letras[1] = 'y';  letras[2] = 'z';
```

```
System.out.println(strLetras);  escreve: abc
```



Entidades (Classes)

- Cultura (Crop)
 - nome
 - valor mínimo
 - valor máximo
- Sistema de Rega (WaterSystem)
 - cultura 1
 - cultura 2
 - nº de dias adequados para ambas
 - nº de dias adequados apenas para a cultura 1
 - nº de dias adequados apenas para a cultura 2
- Main (encarregue da entrada e da saída)
 - sistema de rega (é uma variável auxiliar do método main)

Classe Crop

```
public class Crop {  
  
    private String name;  
    private int minValue;  
    private int maxValue;  
  
    // Pre: name != null && min <= max  
    public Crop( String name, int min, int max ) {...}  
  
    public String getName( ) {...}  
  
    // Pre: day != null  
    public boolean isSuitable( String day ) {...}  
    public boolean isSuitable( int numDrops ) {...}  
  
}
```



2 alternativas possíveis (a classe tem um destes métodos, mas não ambos)

Classe WaterSystem (página 1/2)

```
public class WaterSystem {  
  
    private Crop crop1;  
    private Crop crop2;  
    private int daysForBoth;  
    private int daysOnlyFor1;  
    private int daysOnlyFor2;  
  
    // Pre: name1 != null && min1 <= max1 &&  
    //       name2 != null && min2 <= max2  
    public WaterSystem( String name1, int min1, int max1,  
        String name2, int min2, int max2 ) {...}
```

Classe WaterSystem (página 2/2)

```
// Pre: day != null
public void registerDay( String day ) {...}

public int getNumDaysForBoth( ) {...}

public boolean areCropsEquivalent( ) {...}

// Pre: !this.areCropsEquivalent()
public String getBestCropName( ) {...}

}
```