Exercício 3

Raízes de polinómios

Unidade Curricular de
Laboratórios de Programação

2021/2022

Raízes de Polinómios

Objetivos

- Resolução de problemas utilizando programação

Descrição do problema

Seja

$$P(n) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

um polinómio de grau n, com coeficientes $a_n, a_{n-1}, \dots, a_1, a_0$.

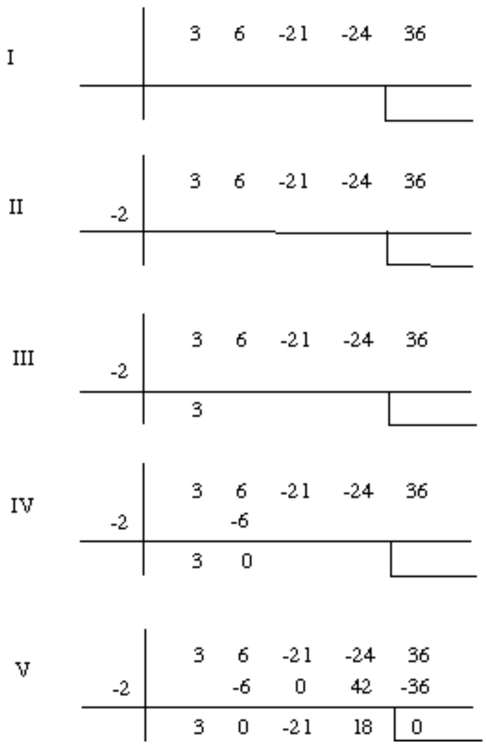
Se `z` é uma raíz de `P`, ou seja, `P(z)=0`, então o polinómio de grau um `(x-z)` divide `P`, ou seja, `P(x)=(x-z)Q(x)`, onde `Q` é um polinómio de grau menor que `P`.

Da mesma forma, se `w` é uma raíz de `Q`, então `Q(x)=(x-w)R(x)`, e, obviamente, `P(x)=(x-z)(x-w)R(x)`, significando que `w` também é raíz de `P`.

Quanto mais raízes de um polinómio conhecemos, mais fácil se torna descobrir as que ainda não conhecemos, pois vamos obtendo polinómios de menor grau. Quando, finalmente, obtemos um polinómio de grau dois – $ax^2 + bx + c$ – temos uma forma direta de calcular as suas raízes usando a formula quadrática:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Como podemos então obter `Q`, tal que `P(x)=(x-z)Q(x)`, conhecendo `P` e uma das suas raízes, `z`? Ou seja, como podemos dividir `P` por `(x-z)`?



Descrevemos agora, com a ajuda de um exemplo, a regra de **Ruffini**, um processo simples para dividir um dado polinómio por um polinómio de grau um da forma `(x-z)`.

Sabendo que uma das raízes do polinómio $P(x) = 3x^4 + 6x^3 - 21x^2 - 24x + 36$ é -2, o exemplo ao lado ilustra a aplicação da regra de Ruffini para calcular a divisão de `P(x)` por `(x - z)`, para o caso em que `z` é `-2`:

- Numa primeira linha escrevemos os coeficientes de `P(x)` (ver figura I).
- Numa segunda linha, à esquerda, escrevemos a raíz do polinómio `(x-z)`, que é `z` (neste caso `z` é `-2`; ver figura II).
- Na terceira linha começamos por escrever o primeiro coeficiente de `P(x)` que é `3` (ver figura III).
- De seguida escrevemos na segunda linha, exatamente abaixo do segundo coeficiente de `P(x)`, o valor que resulta do produto de `z` (que é `-2`) pelo valor que está

à esquerda na terceira linha (que é `3`); neste caso obtemos `-6`. De seguida escrevemos na terceira linha, exatamente abaixo desse valor `-6`, a sua soma com o segundo coeficiente de `P` (que é `6`), o que resulta em zero (ver figura IV).

- Repetimos o passo anterior para os restantes coeficientes de `P` (ver figura V).

No fim do processo, obtemos na terceira linha os coeficientes do polinómio resultante da divisão – `Q` – assim como o resto da divisão (neste caso o resto é zero pois estamos a dividir `P` por uma das suas raízes). Na figura V vemos os coeficientes de `Q` na terceira linha, por isso, $Q(x) = 3x^3 + 0x^2 - 21x + 18$.

Antes de Começar

- Descarregar o arquivo `studentsExercise3.zip` disponível na página de LabP e descompactá-lo. Esse arquivo contém:
  1. O ficheiro `RunPolyRoots.java`
  2. O ficheiro `TestsPolyRoots.java`, contendo vários testes *JUnit*
  3. Seis ficheiros de *input* com extensão `.poly` (ver mais adiante)
  4. O ficheiro de texto `expectedOutput.txt` contendo o resultado esperado da execução de `RunPolyRoots`
- No Eclipse, criar um novo projeto Java e, em seguida:
  a) copiar para dentro da pasta *src* desse projeto os dois ficheiros dados com código Java;
  b) copiar os ficheiros de extensão `.poly` para a raiz do projeto;
  c) Configurar o *Build Path* desse projeto Java para incluir, na *classPath*, a biblioteca *JUnit5* (veja como o fazer na página 14 do tutorial sobre o Eclipse IDE).

O que é para fazer, então?

Implementar um método que, dada a informação sobre os coeficientes de um polinómio de grau `n` e `n-2` raízes desse polinómio, encontre as outras duas raízes. Assume-se que todas as raízes são reais.

Ficheiro de dados de entrada

Os dados de entrada necessários estão contidos num ficheiro de extensão `.poly` com o seguinte formato:

n

$a_n \ a_{n-1} \ \ldots \ a_1 \ a_0$

$r_1 \ r_2 \ \ldots \ r_{n-2}$

onde n é o grau do polinómio, a_n a_{n-1} … a_1 a_0 são os coeficientes do polinómio e r_1 r_2 … r_{n-2} são n-2 raízes do polinómio. Os valores das raízes podem ser repetidos.

Por exemplo, um ficheiro *input1.poly* contendo

```
6
1 -3 -5 15 4 -12 0
1 -2 0 2
```

representa o polinómio de grau seis $x^6 - 3x^5 - 5x^4 + 15x^3 + 4x^2 - 12x$ do qual são conhecidas 4 raízes: 1, -2, 0 e 2.

O método `find2Roots`

Deve criar a classe `PolyRoots` contendo um único método público (quaisquer outros métodos criados deverão ser privados):

- `double[] find2Roots(String inFile)` que, dado o nome de um ficheiro que se assume seguir o formato `.poly` com as informações sobre um polinómio, calcule e devolva as duas raízes que falta conhecer (a raíz de menor valor deve estar na primeira posição do *array* de resultado).

NOTA: deve partir do princípio que os ficheiros de *input* contêm informação correta, ou seja, não precisa de fazer verificações.

Entrega

Deve entregar um *zip* de nome `E3fcxxxxx.zip`, onde `xxxxx` é o seu número de aluno, contendo o ficheiro `PolyRoots.java`.