Polinomios II

Taller de Álgebra I

Verano 2018

División y algoritmo de Euclides

Ejercicios

1. Implementar una función
   `division :: Polinomio -> Polinomio -> (Polinomio, Polinomio)`
   que dados polinomios $P(X)$ y $Q(X)$ calcule el cociente $C(X)$ y el resto $R(X)$ de la división (se debe verificar $P(X) = C(X)Q(X) + R(X)$ y $\deg R(X) < \deg Q(X)$ o $R(X) = 0$).

```
Ejemplo> division [1, 0, 2, 0, (-1), (-8)] [1, (-2), 1]
([1, 2, 5, 8], [10, (-16)])
```

2. Implementar el algortimo de Euclides para calcular el MCD de dos polinomios.
   `mcdP :: Polinomio -> Polinomio -> Polinomio`

```
Ejemplo> mcdP [1, 0, 2, 0, 1, 0] [1, 0, 0, 0, (-1)]
[1, 0, 1]
```

   Como MCD suele tomarse el polinomio mónico, es decir con coeficiente principal 1, pero si está multiplicado por otro número real, también es correcto. Por ejemplo, en vez de `[1, 0, 1]` puede devolver `[-1, 0, -1]`.

Raíces múltiples

Ejercicios

1. Implementar una función
`multiplicidad :: Float -> Polinomio -> Integer`
que dados un real x y un polinomio $P(X)$ determine la multiplicidad de x como raíz de P.

```
Ejemplo> multiplicidad 3 [1, (-6), 9]
2
Ejemplo> multiplicidad 3 [1, (-6), 10]
0
Ejemplo> multiplicidad (-2) [1, 9, 30, 44, 24]
3
```

2. Implementar una función para determinar si un polinomio tiene raíces múltiples.
`raicesMultiples :: Polinomio -> Bool`

```
Ejemplo> raicesMultiples [1, (-6), 10]
False
```

```
Ejemplo> raicesMultiples [1, 9, 30, 44, 24]
True
```

Polinomios en $\mathbb{Z}[X]$

type

Para trabajar con polinomios en $\mathbb{Z}[X]$, definimos renombres de tipos.

```
type Escalar = Integer
type Monomio = (Escalar, Integer)
type Polinomio = [Escalar]
```

(el tipo `Escalar` indica el anillo de coeficientes de los polinomios)

Ejercicio

Adaptar las funciones de la clase anterior a los nuevos tipos.
La función `evaluar` debe permitir evaluar un polinomio con coeficientes enteros en un `Float`. Pueden utilizar la función `realToFrac` que transforma tanto `Integer` como `Float` a `Float`.

Raíces racionales

Ejercicios

1. Implementar una función
`esRaizRacional :: Polinomio -> (Integer, Integer) -> Bool`
que indique si un racional a/b (ingresado como el par (a, b)) es raíz de un polinomio dado.

```
Ejemplo> esRaizRacional [4, (-3), (-25), (-6)] ((-1), 4))
True
```

2. Implementar una función que dado un polinomio y una lista de números racionales devuelva los números de la lista que son raíces del polinomio.
`raicesEnConjunto :: Polinomio -> [(Integer, Integer)] -> [(Integer, Integer)]`

```
Ejemplo> raicesEnConjunto [4, (-3), (-25), (-6)] [((-1), 4), (3, 2), (3,
 1)]
[((-1),4),(3,1)]
```

Teorema de Gauss

Ejercicios

1. Implementar una función
`candidatosRaices :: Polinomio -> [(Integer, Integer)]`
que dado un polinomio devuelva una lista todos los candidatos a raíces según el teorema de Gauss (es decir, los racionales p/q tales que p divide a a_0 y q divide a a_n).

```
Ejemplo> candidatosRaices [2, 5, (-3)]
[(3,2),(3,1),(-3,2),(-3,1),(1,2),(1,1),(-1,2),(-1,1)]
```

2. Implementar una función que dado un polinomio devuelva una lista de todas las raíces racionales, utilizando el teorema de Gauss para encontrarlas.
`raicesRacionales :: Polinomio -> [(Integer, Integer)]`

```
Ejemplo> raicesRacionales [4, (-3), (-25), (-6)]
[(3,1),(-2,1),(-1,4)]
```