Taller de Álgebra I

Clase 12 - Polinomios

Primer cuatrimestre 2020

Polinomios

Esta clase tiene dos partes:

- En la primera parte vamos a trabajar con polinomios con coeficientes en $\mathbb{R}$ (o $\mathbb{Q}$), pero todo lo que haremos se puede llevar al contexto de polinomios con coeficientes en otros cuerpos como $\mathbb{C}$ o $\mathbb{Z}/p\mathbb{Z}$.

- En la segunda parte vamos a trabajar específicamente con polinomios con coeficientes en $\mathbb{Z}$.

Primera parte

Nuevo tipo

Definimos un renombre de tipos para polinomios en $\mathbb{R}[X]$:

```haskell
type Polinomio = [Float]
```

donde el elemento i de la lista contando desde cero y empezando desde la cola corresponde al coeficiente a_i del polinomio $P(X) = \sum_{i=0}^{n} a_i X^i$.

Por ejemplo el polinomio $4X^3 + 3X - 2$ se representa por `[4, 0, 3, (-2)]`.

- Invariante del tipo: la lista de coeficientes no empieza con cero.
  - Se puede suponer que el invariante se cumple en los polinomios que reciben nuestras funciones.
  - Se debe garantizar que el invariante se cumple en los polinomios que devuelven nuestras funciones.

Ejercicio

Implementar las siguientes funciones:

1. `limpiar :: [Float] -> Polinomio`, que dada una lista cualquiera de números reales, le borra los 0s iniciales si los hubiera (luego el resultado cumple el invariante del tipo `Polinomio`).
2. `grado :: Polinomio -> Int` (en general se usa la convención $\text{grado}(0) = -\infty$, en la implementación se puede dejar `grado []` indefinido).
3. `evaluar :: Polinomio -> Float -> Float`, que dados $P \in \mathbb{R}[X]$ y $a \in \mathbb{R}$ calcula $P(a)$.

Suma de polinomios

Para sumar polinomios, queremos implementar la función
`suma :: Polinomio -> Polinomio -> Polinomio`.

No es difícil, pero hay que hacerlo con cuidado...

```
suma [2, 1, 3] [1, 4, 0, 1] = [1, 6, 1, 4]

suma [2, 1, 3] [-2, 5, 0] = [6, 3]
```

Ejercicio

Implementar la función `suma :: Polinomio -> Polinomio -> Polinomio`.

Sugerencia: Tener en cuenta las funciones `init` y `last` del `Prelude`, para hacer la recursión de atrás hacia adelante.

Nuevo tipo

Definimos también un renombre de tipos para monomios en $\mathbb{R}[X]$:

```
type Monomio = (Float, Int)
```

donde el monomio aX^n se representa con `(a, n)`.

El invariante de este tipo es que $a \neq 0$.

Producto de polinomios

Ejercicios

1. `productoPorEscalar :: Float -> Polinomio -> Polinomio`, que calcula el producto de un escalar por un polinomio.

```
Ejemplo> productoPorEscalar 5 [2, 3, (-1)]
[10.0,15.0,-5.0]
```

2. `resta :: Polinomio -> Polinomio -> Polinomio`,
que calcula la resta de polinomios.

```
Ejemplo> resta [8, -4, 2, 7] [8, 1, 1, 0]
[-5.0, 1.0, 7.0]
```

3. `productoPorMonomio :: (Float, Int) -> Polinomio -> Polinomio`, que calcula el producto de un monomio por un polinomio.

```
Ejemplo> productoPorMonomio (5, 2) [2, 3, (-1)]
[10,15.0,-5.0,0.0,0.0]
```

4. `producto :: Polinomio -> Polinomio -> Polinomio`, que calcula el producto de dos polinomios.

```
Ejemplo> producto [2, 3, (-1)] [1, (-3)]
[2.0,-3.0,-10.0,3.0]
```

Derivadas

Ejercicios

Implementar las siguientes funciones:

- `hacerPolinomio :: Monomio -> Polinomio`, que “convierte” un monomio en un polinomio (considerando sus respectivas codificaciones).
- `derivadaMonomio :: Monomio -> Monomio`, que deriva un monomio.
- `derivada :: Polinomio -> Polinomio`
- `derivadaNEsima :: Polinomio -> Int -> Polinomio`, que dados $P \in \mathbb{R}[X]$ y $n \in \mathbb{N}$ calcula la derivada n-ésima de P.

División de polinomios

$$
\begin{array}{rrrrl}
 & 6X^3 + 5X^2 - 7X + 3 & & \multicolumn{1}{|l}{2X^2 + 3X - 1} \\
- & 6X^3 + 9X^2 - 3X & & \multicolumn{1}{|l}{\overline{3X - 2}} \\
\hline
 & -4X^2 - 4X + 3 & & \\
- & -4X^2 - 6X + 2 & & \\
\hline
 & 2X + 1 & &
\end{array}
$$

Ejercicios

Implementar las siguientes funciones:

1. `primerCociente :: Polinomio -> Polinomio -> Monomio`, que calcula el primer monomio del cociente de la división de dos polinomios.

```
Ejemplo> primerCociente [6, 5, -7, 3] [2, 3, -1]
(3.0, 1)
```

2. `primerResto :: Polinomio -> Polinomio -> Polinomio`, que calcula la resta entre el dividendo y el producto del divisor por el primer cociente.

```
Ejemplo> primerResto [6, 5, -7, 3] [2, 3, -1]
[-4.0, -4.0, 3]
```

División de polinomios

$$
\begin{array}{rrrrrrrr|l}
 & 6X^3 & + & 5X^2 & - & 7X & + & 3 & 2X^2 + 3X - 1 \\
- & 6X^3 & + & 9X^2 & - & 3X & & & \hline 3X - 2 \\
\hline
 & & - & 4X^2 & - & 4X & + & 3 & \\
 & - & - & 4X^2 & - & 6X & + & 2 & \\
\hline
 & & & & & 2X & + & 1 &
\end{array}
$$

Ejercicios

3 `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 [6, 5, -7, 3] [2, 3, -1]
([3.0, -2.0], [2.0, 1.0])
```

Ejercicio

1. 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, 0.0, 1.0]
```

Raíces múltiples

Ejercicios

Implementar las siguientes funciones:

1. `multiplicidad :: Float -> Polinomio -> Int`, 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. `raicesMultiples :: Polinomio -> Bool`, que determina si un polinomio tiene raíces múltiples.

   Sugerencia: Pensar qué condición tiene que cumplir el resultado del MCD entre el polinomio pasado como parámetro y su derivada.

```
Ejemplo> raicesMultiples [1, -6, 10]
False
Ejemplo> raicesMultiples [1, 9, 30, 44, 24]
True
```

Segunda parte

Nuevo Tipo

Definimos un renombre de tipos para representar a los números racionales, dado que queremos poner el énfasis en la expresión de estos números como cociente entre dos enteros:

```
type Racional = (Int, Int)
```

El racional p/q se representa con `(p, q)`. El invariante de este tipo es que $q \geq 1$ (el signo del número racional lo "lleva" siempre el numerador) y además p y q son coprimos. De esta manera, la representación es única.

Por ejemplo:

- el racional 0 está representado por `(0, 1)`,
- el racional 1 está representado por `(1, 1)`,
- el racional $-3/4$ está representado por `(-3, 4)`.

Ejercicios

Implementar las siguientes funciones:

1. `sumaR :: Racional -> Racional -> Racional`
2. `multiplicaR :: Racional -> Racional -> Racional`
3. `potenciaR :: Racional -> Int -> Racional`

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

Nuevo Tipo

Para trabajar con polinomios en $\mathbb{Z}[X]$, definimos un renombre de tipos:

```
type PolinomioZ = [Int]
```

Nuevamente, el invariante del tipo `PolinomioZ` es que la lista de coeficientes no tiene ceros a la izquierda.

Ejercicios

Implementar las siguientes funciones:

1. `evaluarZ :: PolinomioZ -> Racional -> Racional`
2. `esRaizRacional :: PolinomioZ -> Racional -> Bool`

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

3. `raicesRacEnConjunto :: PolinomioZ -> Set Racional -> Set Racional`

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

Teorema de Gauss

Implementar las siguientes funciones:

Ejercicios

1. `candidatosRaices :: PolinomioZ -> Set Racional`, que dado un polinomio devuelve un conjunto con 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. `raicesRacionales :: PolinomioZ -> Set Racional`, que dado un polinomio devuelve una lista de todas las raíces racionales, utilizando el teorema de Gauss para encontrarlas.

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