Taller de Álgebra I

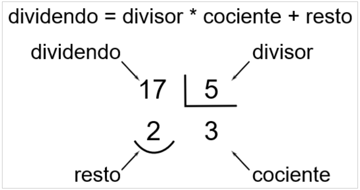
Clase 9 - Algoritmo de división y Algoritmo de Euclides

Primer cuatrimestre 2022

Cociente y resto – Algoritmo de división

Dados a (dividendo), d (divisor) $\in \mathbb{Z}$, $d \neq 0$, existen únicos q (cociente), r (resto) $\in \mathbb{Z}$ tales que

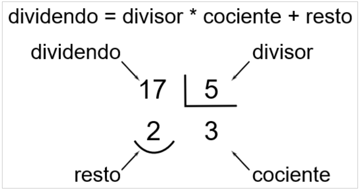
- $a = dq + r$,
- $0 \leq r < |d|$.



Cociente y resto – Algoritmo de división

Dados a (dividendo), d (divisor) $\in \mathbb{Z}$, $d \neq 0$, existen únicos q (cociente), r (resto) $\in \mathbb{Z}$ tales que

- $a = dq + r$,
- $0 \leq r < |d|$.



Implementar la siguiente función

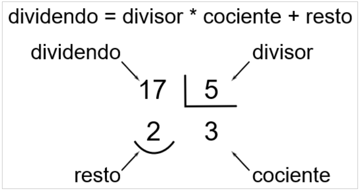
```
division :: Int -> Int -> (Int, Int)
```

Debe funcionar para $a \geq 0$, $d > 0$ y no se pueden usar `div`, `mod` ni `/`.

Cociente y resto – Algoritmo de división

Dados a (dividendo), d (divisor) $\in \mathbb{Z}$, $d \neq 0$, existen únicos q (cociente), r (resto) $\in \mathbb{Z}$ tales que

- $a = dq + r$,
- $0 \leq r < |d|$.



Implementar la siguiente función

```
division :: Int -> Int -> (Int, Int)
```

Debe funcionar para $a \geq 0$, $d > 0$ y no se pueden usar `div`, `mod` ni `/`.

Ideas

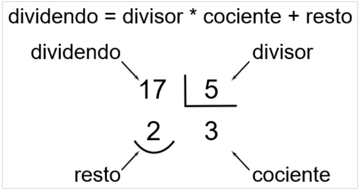
Usaremos recursión (idea sacada de la demostración del teorema de la división)

- `division a d = ??`

Cociente y resto – Algoritmo de división

Dados a (dividendo), d (divisor) $\in \mathbb{Z}$, $d \neq 0$, existen únicos q (cociente), r (resto) $\in \mathbb{Z}$ tales que

- $a = dq + r$,
- $0 \leq r < |d|$.



Implementar la siguiente función

```
division :: Int -> Int -> (Int, Int)
```

Debe funcionar para $a \geq 0$, $d > 0$ y no se pueden usar `div`, `mod` ni `/`.

Ideas

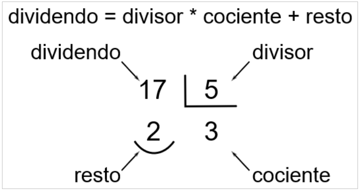
Usaremos recursión (idea sacada de la demostración del teorema de la división)

- `division a d = ??`
- Hacemos recursión... sobre d? Para `division 15 4`, me sirve `division 15 3`?

Cociente y resto – Algoritmo de división

Dados a (dividendo), d (divisor) $\in \mathbb{Z}$, $d \neq 0$, existen únicos q (cociente), r (resto) $\in \mathbb{Z}$ tales que

- $a = dq + r$,
- $0 \leq r < |d|$.



Implementar la siguiente función

```
division :: Int -> Int -> (Int, Int)
```

Debe funcionar para $a \geq 0$, $d > 0$ y no se pueden usar `div`, `mod` ni `/`.

Ideas

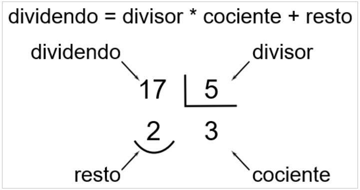
Usaremos recursión (idea sacada de la demostración del teorema de la división)

- `division a d = ??`
- Hacemos recursión... sobre d? Para `division 15 4`, me sirve `division 15 3`?
- Hacemos recursión... sobre a? Para `division 15 4`, me sirve `division 14 4`?

Cociente y resto – Algoritmo de división

Dados a (dividendo), d (divisor) $\in \mathbb{Z}$, $d \neq 0$, existen únicos q (cociente), r (resto) $\in \mathbb{Z}$ tales que

- $a = dq + r$,
- $0 \leq r < |d|$.



Implementar la siguiente función

```
division :: Int -> Int -> (Int, Int)
```

Debe funcionar para $a \geq 0$, $d > 0$ y no se pueden usar `div`, `mod` ni `/`.

Ideas

Usaremos recursión (idea sacada de la demostración del teorema de la división)

- `division a d = ??`
- Hacemos recursión... sobre d? Para `division 15 4`, me sirve `division 15 3`?
- Hacemos recursión... sobre a? Para `division 15 4`, me sirve `division 14 4`?
- Para `division 15 4`, me sirve `division k 4` para algún k?
- Para determinar lo anterior, pensar qué quiere decir dividir un número por otro.

Algoritmo de división

```
division :: Int -> Int -> (Int, Int)
division a d | a < d = (0, a)
             | otherwise = (fst (division (a-d) d) + 1,
                            snd (division (a-d) d))
```

Algoritmo de división

```
division :: Int -> Int -> (Int, Int)
division a d | a < d = (0, a)
             | otherwise = (fst (division (a-d) d) + 1,
                            snd (division (a-d) d))
```

¿Se puede no poner dos veces `division (a-d) d`? Sí:

Algoritmo de división

```
division :: Int -> Int -> (Int, Int)
division a d | a < d = (0, a)
             | otherwise = (fst (division (a-d) d) + 1,
                            snd (division (a-d) d))
```

¿Se puede no poner dos veces `division (a-d) d`? Sí:

```
division :: Int -> Int -> (Int, Int)
division a d | a < d = (0, a)
             | otherwise = (fst qr' + 1, snd qr')
              where qr' = division (a-d) d
```

Ejercicio

Extender la función `division :: Int -> Int -> (Int, Int)` para que funcione para $a \in \mathbb{Z}$, $d > 0$.

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a \ : \ b) = (a + kb \ : \ b)$$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a \ : \ b) = (a + kb \ : \ b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a \ : \ b) = (a - qb \ : \ b) = (r \ : \ b) = (b \ : \ r)$$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a : b) = (a + kb : b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a : b) = (a - qb : b) = (r : b) = (b : r)$$

Por ejemplo, para calcular $(30 : 48)$:

1. $(30 : 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a : b) = (a + kb : b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a : b) = (a - qb : b) = (r : b) = (b : r)$$

Por ejemplo, para calcular $(30 : 48)$:

1. $(30 : 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$
2. $= (48 : 30)$ — Dividimos 48 por 30, $q = 1$, $r = 18$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a : b) = (a + kb : b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a : b) = (a - qb : b) = (r : b) = (b : r)$$

Por ejemplo, para calcular $(30 : 48)$:

1. $(30 : 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$
2. $= (48 : 30)$ — Dividimos 48 por 30, $q = 1$, $r = 18$
3. $= (30 : 18)$ — $q = 1$, $r = 12$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a : b) = (a + kb : b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a : b) = (a - qb : b) = (r : b) = (b : r)$$

Por ejemplo, para calcular $(30 : 48)$:

1. $(30 : 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$
2. $= (48 : 30)$ — Dividimos 48 por 30, $q = 1$, $r = 18$
3. $= (30 : 18)$ — $q = 1$, $r = 12$
4. $= (18 : 12)$ — $q = 1$, $r = 6$

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a \ : \ b) = (a + kb \ : \ b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a \ : \ b) = (a - qb \ : \ b) = (r \ : \ b) = (b \ : \ r)$$

Por ejemplo, para calcular $(30 \ : \ 48)$:

1. $(30 \ : \ 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$
2. $= (48 \ : \ 30)$ — Dividimos 48 por 30, $q = 1$, $r = 18$
3. $= (30 \ : \ 18)$ — $q = 1$, $r = 12$
4. $= (18 \ : \ 12)$ — $q = 1$, $r = 6$
5. $= (12 \ : \ 6)$ — $q = 2$, $r = 0$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a : b) = (a + kb : b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a : b) = (a - qb : b) = (r : b) = (b : r)$$

Por ejemplo, para calcular $(30 : 48)$:

1. $(30 : 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$
2. $= (48 : 30)$ — Dividimos 48 por 30, $q = 1$, $r = 18$
3. $= (30 : 18)$ — $q = 1$, $r = 12$
4. $= (18 : 12)$ — $q = 1$, $r = 6$
5. $= (12 : 6)$ — $q = 2$, $r = 0$
6. $= (6 : 0)$

Algoritmo de Euclides

El **algoritmo de Euclides** calcula el máximo común divisor entre dos números $a, b \in \mathbb{Z}$.

Se basa en que si a, $b \in \mathbb{Z}$ y $k \in \mathbb{Z}$ es un número cualquiera, entonces

$$(a : b) = (a + kb : b)$$

Si q y r son el cociente y el resto de la división de a por b, tenemos $a = qb + r$, entonces $a - qb = r$. Por lo tanto,

$$(a : b) = (a - qb : b) = (r : b) = (b : r)$$

Por ejemplo, para calcular $(30 : 48)$:

1. $(30 : 48)$ — Dividimos 30 por 48, $q = 0$, $r = 30$
2. $= (48 : 30)$ — Dividimos 48 por 30, $q = 1$, $r = 18$
3. $= (30 : 18)$ — $q = 1$, $r = 12$
4. $= (18 : 12)$ — $q = 1$, $r = 6$
5. $= (12 : 6)$ — $q = 2$, $r = 0$
6. $= (6 : 0)$
7. $= 6$

Algoritmo de Euclides: Ejercicios

Ejercicios

1. Programar la función
   `mcd :: Int -> Int -> Int`
   que calcule el máximo común divisor entre dos números utilizando el algoritmo de Euclides.
   `mcd a b` debe funcionar siempre que `a` $>$ `0`, `b` $\geq$ `0`.

2. Pensar otro algoritmo para calcular el máximo común divisor. Ideas:
   - usar la función `menorDivisor` programada en clases anteriores.
   - extender la función `menorDivisor` para que calcule el `mayorDivisorComun`

3. Comparar el tiempo que tardan ambos programas para números pequeños y números grandes (por ejemplo, números de 10 dígitos). En GHCI se puede saber el tiempo que tarda una función utilizado el comando: `set +s`

Algoritmo de Euclides extendido

Dados números $a, b \in \mathbb{Z}$, existen enteros s, t tales que

$$sa + tb = (a : b).$$

Los valores de s, t se pueden obtener con la versión extendida del algoritmo de Euclides.

Ejemplos

- $(8 : 5) = 1 \quad \text{y} \quad 2 \cdot 8 - 3 \cdot 5 = 1$
- $(9 : 15) = 3 \quad \text{y} \quad 2 \cdot 9 - 1 \cdot 15 = 3$

Algoritmo de Euclides extendido

Dados números $a, b \in \mathbb{Z}$, existen enteros s, t tales que

$$sa + tb = (a : b).$$

Los valores de s, t se pueden obtener con la versión extendida del algoritmo de Euclides.

Ejemplos

- $(8 : 5) = 1$ y $2 \cdot 8 - 3 \cdot 5 = 1$
- $(9 : 15) = 3$ y $2 \cdot 9 - 1 \cdot 15 = 3$

Un ejemplo fácil

Para el caso de $(n : 0)$ es trivial.

Algoritmo de Euclides extendido

Dados números $a, b \in \mathbb{Z}$, existen enteros s, t tales que

$$sa + tb = (a \ : \ b).$$

Los valores de s, t se pueden obtener con la versión extendida del algoritmo de Euclides.

Ejemplos

- $(8 \ : \ 5) = 1 \quad$ y $\quad 2 \cdot 8 - 3 \cdot 5 = 1$
- $(9 \ : \ 15) = 3 \quad$ y $\quad 2 \cdot 9 - 1 \cdot 15 = 3$

Un ejemplo fácil

Para el caso de $(n \ : \ 0)$ es trivial.
$(n \ : \ 0) = n = n * \mathbf{1} + 0 * \mathbf{0}$.
Es decir, $s = 1$ y $t = 0$

Algoritmo de Euclides extendido

Por algoritmo de división:

$$a = bq + r \tag{1}$$

Y por el algoritmo de Euclides clásico, sabemos que:

$$(a \ : \ b) = (b \ : \ r) = g \tag{2}$$

Algoritmo de Euclides extendido

Por algoritmo de división:

$$a = bq + r \tag{1}$$

Y por el algoritmo de Euclides clásico, sabemos que:

$$(a : b) = (b : r) = g \tag{2}$$

Si asumimos que tenemos s', t' tales que:

$$s'b + t'r = g \tag{3}$$

¿Cómo obtenemos $sa + tb = g$?

Algoritmo de Euclides extendido

Por algoritmo de división:

$$a = bq + r \tag{1}$$

Y por el algoritmo de Euclides clásico, sabemos que:

$$(a : b) = (b : r) = g \tag{2}$$

Si asumimos que tenemos s', t' tales que:

$$s'b + t'r = g \tag{3}$$

¿Cómo obtenemos $sa + tb = g$?
Multiplicamos (1) por $\boldsymbol{t'}$:

$$\boldsymbol{t'}a = \boldsymbol{t'}bq + \boldsymbol{t'}r$$

Algoritmo de Euclides extendido

Por algoritmo de división:

$$a = bq + r \tag{1}$$

Y por el algoritmo de Euclides clásico, sabemos que:

$$(a : b) = (b : r) = g \tag{2}$$

Si asumimos que tenemos s', t' tales que:

$$s'b + t'r = g \tag{3}$$

¿Cómo obtenemos $sa + tb = g$?
Multiplicamos (1) por $\boldsymbol{t'}$:

$$\boldsymbol{t'}a = \boldsymbol{t'}bq + \boldsymbol{t'}r$$

Reemplazamos la expresión de $t'r$ según (3):

$$t'a = t'bq + (\boldsymbol{g - s'b})$$

Algoritmo de Euclides extendido

Por algoritmo de división:

$$a = bq + r \tag{1}$$

Y por el algoritmo de Euclides clásico, sabemos que:

$$(a : b) = (b : r) = g \tag{2}$$

Si asumimos que tenemos s', t' tales que:

$$s'b + t'r = g \tag{3}$$

¿Cómo obtenemos $sa + tb = g$?
Multiplicamos (1) por $\boldsymbol{t'}$:

$$\boldsymbol{t'}a = \boldsymbol{t'}bq + \boldsymbol{t'}r$$

Reemplazamos la expresión de $t'r$ según (3):

$$t'a = t'bq + (\boldsymbol{g - s'b})$$

Despejamos g

$$t'a - t'bq + s'b = g$$

Algoritmo de Euclides extendido

Por algoritmo de división:

$$a = bq + r \tag{1}$$

Y por el algoritmo de Euclides clásico, sabemos que:

$$(a : b) = (b : r) = g \tag{2}$$

Si asumimos que tenemos s', t' tales que:

$$s'b + t'r = g \tag{3}$$

¿Cómo obtenemos $sa + tb = g$?
Multiplicamos (1) por $\boldsymbol{t'}$:

$$\boldsymbol{t'}a = \boldsymbol{t'}bq + \boldsymbol{t'}r$$

Reemplazamos la expresión de $t'r$ según (3):

$$t'a = t'bq + (\boldsymbol{g - s'b})$$

Despejamos g

$$t'a - t'bq + s'b = g$$

Sacamos factor común

$$\boldsymbol{t'}a + (\boldsymbol{s' - t'q})b = g$$

Es decir, $s = t'$ y $t = (s' - t'q)$.

Algoritmo extendido de Euclides

1. Programar la función
```
emcd :: Int -> Int -> (Int, Int, Int)
```
que utilice el algoritmo de Euclides extendido para obtener una 3-upla (g, s, t) tal que $g = (a : b) = sa + tb$.
Recordar que $s = t'$ y $t = (s' - t'q)$ (ver diapositiva anterior).

Sugerencia: para acceder a los elementos de la 3-upla, podemos definir las funciones

```
fst3 (x, _, _) = x
snd3 (_, y, _) = y
trd3 (_, _, z) = z
```