PARTE I

Algoritmo de división (lento y conceptual)

Teorema de la división

Teorema

Dados $a \in \mathbb{Z}$, $d \in \mathbb{Z}$, $d \neq 0$, existen únicos $q, r \in \mathbb{Z}$ tales que $a = qd + r$ y $0 \leq r < |d|$.

Terminología y notación

- a es el dividendo, d el divisor, q el cociente y r el resto
- q se escribe a/d, mientras que r se escribe $r_d(a)$

Problemas: dados $a \in \mathbb{Z}$ y $d \in \mathbb{Z} \setminus \{0\}$

- división: determinar a/d
- resto: determinar $r_d(a)$

División y resto en naturales

Demostración inductiva de existencia para $a \geq 0, d > 0$

Casos base: $a < d$. Entonces $a = 0d + a \Rightarrow q = 0$ y $r = a$.
Paso inductivo $a \geq d$. Por inducción, como $a - d \geq 0$, existen q' y $0 \leq r < d$ tales que $(a - d) = q'd + r$. En consecuencia, $a = qd + r$ con $q = q' + 1$.

División y resto en naturales

Demostración inductiva de existencia para $a \geq 0, d > 0$

Casos base: $a < d$. Entonces $a = 0d + a \Rightarrow q = 0$ y $r = a$.
Paso inductivo $a \geq d$. Por inducción, como $a - d \geq 0$, existen q' y $0 \leq r < d$ tales que $(a - d) = q'd + r$. En consecuencia, $a = qd + r$ con $q = q' + 1$.

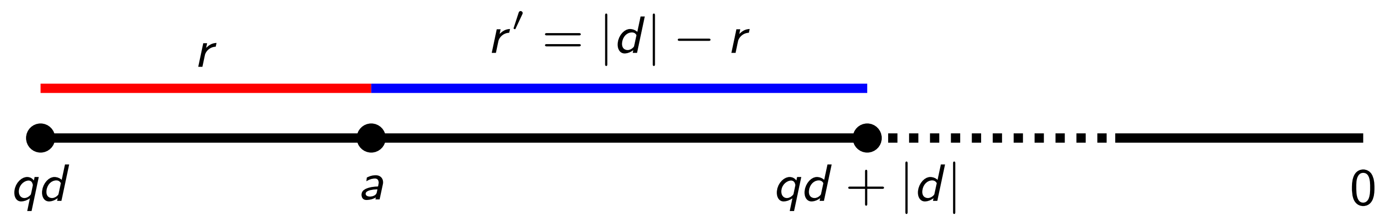
```
-- |Division de numeros naturales_0: a `divNat` d = a `div` d
divNat :: Int -> Int -> Int
divNat a d | a < d = 0
           | otherwise = (a-d) `divNat` d + 1

-- |Resto de numeros naturales_0: a `modNat` d = a `mod` d
modNat :: Int -> Int -> Int
modNat a d = a - d*(a `divNat` d)
```

Algoritmo general para el resto

Observación

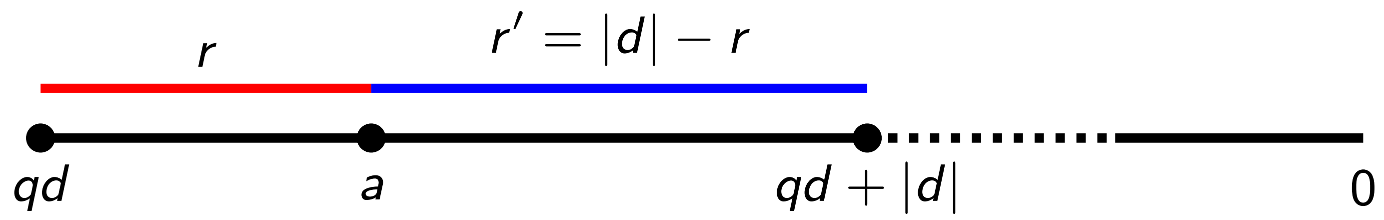
Supongamos que $a = qd + r$ y sea $r' = r_{|d|}(|a|)$. Si $a \geq 0$ o $r' = 0$, entonces $r = r'$; caso contrario, $r = |d| - r'$.



Algoritmo general para el resto

Observación

Supongamos que $a = qd + r$ y sea $r' = r_{|d|}(|a|)$. Si $a \geq 0$ o $r' = 0$, entonces $r = r'$; caso contrario, $r = |d| - r'$.

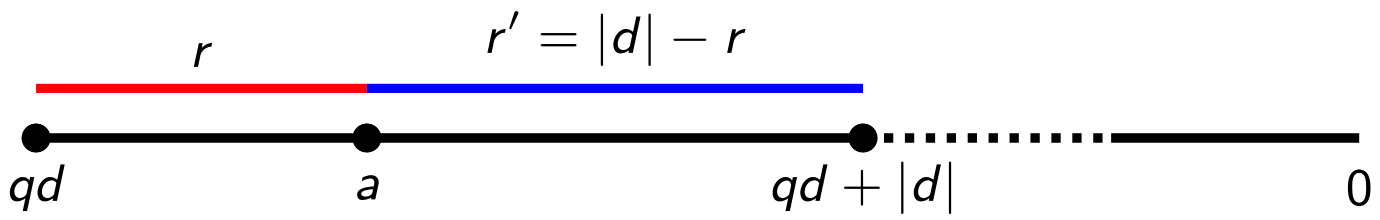


1. Si $a \geq 0$, entonces $|a| = a = qd + r = |qd + r| = |q||d| + r$. Luego $r' = r$
2. Si $a < 0$, entonces $|a| = -a = (-qd - |d|) + (|d| - r) = (|q| - 1)|d| + (|d| - r)$.
   Luego, $|d| - r \equiv r' (\mathrm{mod}\, |d|)$ y, como $r < |d| \Rightarrow r = 0$ si $r' = 0$ y $|d| - r = r'$ si $r' \neq 0$.

Algoritmo general para el resto

Observación

Supongamos que $a = qd + r$ y sea $r' = r_{|d|}(|a|)$. Si $a \geq 0$ o $r' = 0$, entonces $r = r'$; caso contrario, $r = |d| - r'$.



1. Si $a \geq 0$, entonces $|a| = a = qd + r = |qd + r| = |q||d| + r$. Luego $r' = r$
2. Si $a < 0$, entonces $|a| = -a = (-qd - |d|) + (|d| - r) = (|q| - 1)|d| + (|d| - r)$.
   Luego, $|d| - r \equiv r'(\mathrm{mod}|d|)$ y, como $r < |d| \Rightarrow r = 0$ si $r' = 0$ y $|d| - r = r'$ si $r' \neq 0$.

```
-- |Modulo de numeros enteros a `modulo` d = a `mod` d
modulo :: Int -> Int -> Int
modulo a d | a >= 0 || r' == 0 = r'
           | otherwise = abs d - r'
    where r' = abs a `modNat` abs d
```

Algoritmo general para la división

Observaciones: para calcular la división

1. Para todo n, $n = \mathrm{sgn}(n) \cdot |n|$ y, si $n = 0$, entonces $n = p|n|$ para todo p
2. Si $a = qd + r$, entonces $|a - r| = |qd| = |q| \cdot |d|$, i.e., $|q| = |a - r|/|d|$
3. Si $a = qd + r$, entonces $q = 0$ o $\mathrm{sgn}(q) = \mathrm{sgn}(a) \cdot \mathrm{sgn}(d)$

Algoritmo general para la división

Observaciones: para calcular la división

1. Para todo n, $n = \text{sgn}(n) \cdot |n|$ y, si $n = 0$, entonces $n = p|n|$ para todo p
2. Si $a = qd + r$, entonces $|a - r| = |qd| = |q| \cdot |d|$, i.e., $|q| = |a - r|/|d|$
3. Si $a = qd + r$, entonces $q = 0$ o $\text{sgn}(q) = \text{sgn}(a) \cdot \text{sgn}(d)$

```haskell
-- |Division de numeros enteros: n `dividido` m = n `div` m
dividido :: Int -> Int -> Int
dividido a d = sgq * absq                        --obs 1
  where absq = abs (a-r) `divNat` (abs d)        --obs 2
        sgq = (signum a) * (signum d)            --obs 3
        r = a `modulo` d
```

A partir de ahora, usamos `div` **y** `mod` **que son más eficientes**

PARTE II

Sistemas de numeración

Escritura de números en una base

Propiedad

Si $b > 1$ y $n > 0$, entonces existe una única secuencia $d_k, \ldots, d_0$ tal que

- $d_k > 0$ y $0 \leq d_i < b$ para todo $0 \leq i \leq k$, y
- $n = d_k b^k + \ldots + d_0 b^0$

Terminología: visto en teórica

- $(d_k \ldots d_0)_b$ es la representación de n en base b.
- $(0)_b$ es la representación de 0 en base b.

Representación por listas

- Escribimos $[d_0, \ldots, d_k]_b$ para todo $n > 0$ y $[]_b = 0$
- Notar que la lista $[d_0, \ldots, d_k]_b$ "se escribe" al revés de $(d_k \ldots d_0)_b \ldots$
- $\ldots$, lo que *conviene para programar* porque el i-ésimo corresponde al digito b^i.

Ejemplos:

- $1537 = (1537)_{10} = [7, 3, 5, 1]_{10}$, $29 = (11101)_2 = [1, 0, 1, 1, 1]_2$
- $1024 = (400)_{16} = [0, 0, 4]_{16}$, $255 = (FF)_{16} = [15, 15]_{16}$

Observaciones: si $n = [d_0, \ldots, d_k]_b$ y $n > 0$, entonces

- $d_0 = n \bmod b$ (¿por qué?)
- $n/b = [d_1, \ldots, d_k]_b$ (¿por qué?)

Ejercicios

1. Definir la función `digitos :: Integer -> Integer -> [Integer]` que, dados $n \geq 0$ y $b > 1$, retorne su representación por listas en base b.
2. Definir la función `numero :: [Integer] -> Integer -> Integer` que, dada la representación por listas de $n \geq 0$ en base b y la base $b > 1$, retorne n.

PARTE III

Algoritmo de Euclides

Máximo común divisor

Problema del máximo comun divisor (mcd):

- Dados $a, b \in \mathbb{Z}$ $(|a| + |b| > 0)$, encontrar $(a : b) = \max\{d \text{ t. q. } d \mid a \text{ y } d \mid b\}$
- $(a : b)$ es el máximo común divisor (mcd) de a y b
- Obs: $(a : b)$ siempre existe y es positivo (¿por qué?)

Posible algoritmo: por definición

```
mcdDef :: Int -> Int -> Int
mcdDef a 0 = abs a
mcdDef 0 b = abs b
mcdDef a b = maximo (interseccion (divisores a) (divisores b))
```

Ejercicios

3. Escribir la función `divisores :: Int -> Set Int` que dado un valor $n \neq 0$ retorna el conjunto de sus divisores positivos
4. Completar la función `mcdDef`, definiendo las funciones restantes
5. Medir el tiempo que tarda `mcdDef` para un par de valores en $10^{10} \leq a, b \leq 2 \cdot 10^{10}$

Algoritmo de Euclides

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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. $= (18 : 12)$ — $q = 1$, $r = 6$

Algoritmo de Euclides

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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. $= (12 : 6)$ — $q = 2$, $r = 0$

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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. $= (6 : 0)$

Observación fundamental

$(a : b) = (a + kb : b)$ para todo $k \in \mathbb{Z}$

Si $a = qb + r$, entonces

- $(a : b) = (a - qb : b) = (r : b) = (b : r)$
- Luego, $(a : b) = (b : r_b(a))$ define un algoritmo que termina porque $0 \leq r_b(a) < |b|$
- Como caso base, $(a : 0) = |a|$

Ejemplo:

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. $= 6$

Mini-análisis de eficiencia

Observación

Si $b \geq c > 0$ y $r = r_c(b)$, entonces $r < b/2$

1. Sea q el cociente de dividir b por c, i.e., $b = qc + r$
2. Si $q > 1$, entonces $r < c \leq b/q \leq b/2$
3. Si $q = 1$, entonces $c > b/2$ y, por lo tanto, $r = b - c < b/2$

Mini-análisis de eficiencia

Observación

Si $b \geq c > 0$ y $r = r_c(b)$, entonces $r < b/2$

1. Sea q el cociente de dividir b por c, i.e., $b = qc + r$
2. Si $q > 1$, entonces $r < c \leq b/q \leq b/2$
3. Si $q = 1$, entonces $c > b/2$ y, por lo tanto, $r = b - c < b/2$

Corolario:

- En dos pasos de Euclides (suponiendo $a \geq b > 0$) tenemos $(a \ : \ b) = (c \ : \ r)$ para $c = r_b(a)$ y $r = r_c(b) < b/2$
- Luego, si $b < 2^k$, entonces se requieren no mas de $2k$ pasos

Mini-análisis de eficiencia

Observación

Si $b \geq c > 0$ y $r = r_c(b)$, entonces $r < b/2$

1. Sea q el cociente de dividir b por c, i.e., $b = qc + r$
2. Si $q > 1$, entonces $r < c \leq b/q \leq b/2$
3. Si $q = 1$, entonces $c > b/2$ y, por lo tanto, $r = b - c < b/2$

Corolario:

- En dos pasos de Euclides (suponiendo $a \geq b > 0$) tenemos $(a : b) = (c : r)$ para $c = r_b(a)$ y $r = r_c(b) < b/2$
- Luego, si $b < 2^k$, entonces se requieren no mas de $2k$ pasos

Ejemplo: $(a : b)$ para $b \approx 2^{1000}$

- Si pudieramos calcular 10^{12} restos por segundo, y
- el algoritmo básico para divisores recorre los valores en $[1, \sqrt{b}]$
- Tiempo Euclides $\lesssim 2000/10^{12}\text{s} = 2\text{ns}$
- Tiempo divisores $\approx \sqrt{2^{1000}}/10^{12}\text{s} > 10^{131}$ años

Algoritmo de Euclides: Ejercicios

Ejercicios

6. Definir la función `mcd :: Int -> Int -> Int` que dados $a, b \in \mathbb{Z}$, $b \neq 0$, calcule $(a : b)$ usando el algoritmo de Euclides.
7. Medir el tiempo de esta función y compararlo con `mcdDef`.
8. Definir un función `mcm :: Int -> Int -> Int` que dados $a \geq 0$ y $b \geq 0$ calcule el mínimo $d \geq 0$ que sea múltiplo tanto de a como de b. ¿Cuánto vale `mcm 0 0`?

PARTE IV

Algoritmo de Euclides extendido

Corolario del algoritmo de Euclides (observación fundamental)

Para todo $a, b \in \mathbb{Z}$ existen (infinitos pares) $s, t \in \mathbb{Z}$ tales que $sa + tb = (a : b)$

Ejemplos

- $(8 : 5) = 1$ y $2 \cdot 8 - 3 \cdot 5 = 1$ (o $7 \cdot 8 - 11 \cdot 5 = 1$)
- $(9 : 15) = 3$ y $2 \cdot 9 - 1 \cdot 15 = 3$

Algoritmo de Euclides extendido

- Dados a y b, computa $(a : b)$ junto con algún par s y t.

Algoritmo de Euclides extendido (contd)

Corolario del algoritmo de Euclides (observación fundamental)

Para todo $a, b \in \mathbb{Z}$ existen (infinitos pares) $s, t \in \mathbb{Z}$ tales que $sa + tb = (a : b)$

Caso base: $b = 0$. Entonces $(a : 0) = |a| = (\operatorname{sgn} a)a + tb$ para todo $t \in \mathbb{Z}$

Algoritmo de Euclides extendido (contd)

Corolario del algoritmo de Euclides (observación fundamental)

Para todo $a, b \in \mathbb{Z}$ existen (infinitos pares) $s, t \in \mathbb{Z}$ tales que $sa + tb = (a \ : \ b)$

Caso base: $b = 0$. Entonces $(a \ : \ 0) = |a| = (\operatorname{sgn} a)a + tb$ para todo $t \in \mathbb{Z}$

Paso inductivo: $|b| > 0$

- Sea $r = r_b(a)$, i.e., $r = a - qb$ donde $q = a/b$

Algoritmo de Euclides extendido (contd)

Corolario del algoritmo de Euclides (observación fundamental)

Para todo $a, b \in \mathbb{Z}$ existen (infinitos pares) $s, t \in \mathbb{Z}$ tales que $sa + tb = (a : b)$

Caso base: $b = 0$. Entonces $(a : 0) = |a| = (\operatorname{sgn} a)a + tb$ para todo $t \in \mathbb{Z}$

Paso inductivo: $|b| > 0$

- Sea $r = r_b(a)$, i.e., $r = a - qb$ donde $q = a/b$
- Por inducción, existen σ e infinitos τ con $\sigma b + \tau r = (b : r)$
- Por Euclides, $(a : b) = (b : r)$

Corolario del algoritmo de Euclides (observación fundamental)

Para todo $a, b \in \mathbb{Z}$ existen (infinitos pares) $s, t \in \mathbb{Z}$ tales que $sa + tb = (a : b)$

Caso base: $b = 0$. Entonces $(a : 0) = |a| = (\operatorname{sgn} a)a + tb$ para todo $t \in \mathbb{Z}$

Paso inductivo: $|b| > 0$

- Sea $r = r_b(a)$, i.e., $r = a - qb$ donde $q = a/b$
- Por inducción, existen σ e infinitos τ con $\sigma b + \tau r = (b : r)$
- Por Euclides, $(a : b) = (b : r)$
- Luego,
$$(a : b) = (b : r) = \sigma b + \tau r = \sigma b + \tau(a - qb) = \tau a + (\sigma - q\tau)b$$

Algoritmo de Euclides extendido (contd)

Corolario del algoritmo de Euclides (observación fundamental)

Para todo $a, b \in \mathbb{Z}$ existen (infinitos pares) $s, t \in \mathbb{Z}$ tales que $sa + tb = (a : b)$

Caso base: $b = 0$. Entonces $(a : 0) = |a| = (\operatorname{sgn} a)a + tb$ para todo $t \in \mathbb{Z}$

Paso inductivo: $|b| > 0$

- Sea $r = r_b(a)$, i.e., $r = a - qb$ donde $q = a/b$
- Por inducción, existen σ e infinitos τ con $\sigma b + \tau r = (b : r)$
- Por Euclides, $(a : b) = (b : r)$
- Luego,
$$(a : b) = (b : r) = \sigma b + \tau r = \sigma b + \tau(a - qb) = \tau a + (\sigma - q\tau)b$$
- Por lo tanto, alcanza con tomar $s = \tau$ y $t = \sigma - (a/b)\tau$
- Ciertamente, s y t son enteros y $\sigma - (a/b)\tau \neq \sigma - (a/b)\tau'$ para $\tau \neq \tau'$

Algoritmo de Euclides extendido (contd.)

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$
6. ¿$(6 : 0), s_5, t_5$?

Algoritmo de Euclides extendido (contd.)

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$
6. ¿$(6 : 0), s_5, t_5$? $\rightarrow (6 : 0) = 6,\ s_5 = 1,\ t_5 = 2.$

Algoritmo de Euclides extendido (contd.)

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$ $\quad \rightarrow (12 : 6) = 6,\ s_4 = 2,\ t_4 = 1 - 2 \cdot 2 = -3$
6. ¿$(6 : 0), s_5, t_5$? $\quad \rightarrow (6 : 0) = 6,\ s_5 = 1,\ t_5 = 2.$

Algoritmo de Euclides extendido (contd.)

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$ $\quad \rightarrow (18 : 12) = 6,\ s_3 = -3,\ t_3 = 2 - 1 \cdot (-3) = 5.$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$ $\quad \rightarrow (12 : 6) = 6,\ s_4 = 2,\ t_4 = 1 - 2 \cdot 2 = -3$
6. ¿$(6 : 0), s_5, t_5$? $\quad \rightarrow (6 : 0) = 6,\ s_5 = 1,\ t_5 = 2.$

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$ $\quad \to (30 : 18) = 6,\ s_2 = 5,\ t_2 = -3 - 1 \cdot 5 = -8$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$ $\quad \to (18 : 12) = 6,\ s_3 = -3,\ t_3 = 2 - 1 \cdot (-3) = 5.$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$ $\quad \to (12 : 6) = 6,\ s_4 = 2,\ t_4 = 1 - 2 \cdot 2 = -3$
6. ¿$(6 : 0), s_5, t_5$? $\quad \to (6 : 0) = 6,\ s_5 = 1,\ t_5 = 2.$

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$ $\rightarrow (48 : 30) = 6,\ s_1 = -8,\ t_1 = 5 - 1 \cdot (-8) = 13$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$ $\rightarrow (30 : 18) = 6,\ s_2 = 5,\ t_2 = -3 - 1 \cdot 5 = -8$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$ $\rightarrow (18 : 12) = 6,\ s_3 = -3,\ t_3 = 2 - 1 \cdot (-3) = 5.$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$ $\rightarrow (12 : 6) = 6,\ s_4 = 2,\ t_4 = 1 - 2 \cdot 2 = -3$
6. ¿$(6 : 0), s_5, t_5$? $\rightarrow (6 : 0) = 6,\ s_5 = 1,\ t_5 = 2.$

Algoritmo de Euclides extendido (contd.)

Queremos encontrar $(a : b), s, t$ conociendo $(b : r), \sigma, \tau$.

- Tomar $s = \tau$ y $t = \sigma - (a/b)\tau$.
- El caso base es $b = 0$

Ejemplo:

1. ¿$(30 : 48), s_0, t_0$? $q_0 = 0$ $\to (30 : 48) = 6,\ s_0 = 13,\ t_0 = -8 - 0 \cdot 13 = -8$
2. ¿$(48 : 30), s_1, t_1$? $q_1 = 1$ $\to (48 : 30) = 6,\ s_1 = -8,\ t_1 = 5 - 1 \cdot (-8) = 13$
3. ¿$(30 : 18), s_2, t_2$? $q_2 = 1$ $\to (30 : 18) = 6,\ s_2 = 5,\ t_2 = -3 - 1 \cdot 5 = -8$
4. ¿$(18 : 12), s_3, t_3$? $q_3 = 1$ $\to (18 : 12) = 6,\ s_3 = -3,\ t_3 = 2 - 1 \cdot (-3) = 5.$
5. ¿$(12 : 6), s_4, t_4$? $q_4 = 2$ $\to (12 : 6) = 6,\ s_4 = 2,\ t_4 = 1 - 2 \cdot 2 = -3$
6. ¿$(6 : 0), s_5, t_5$? $\to (6 : 0) = 6,\ s_5 = 1,\ t_5 = 2.$

Entonces,

$$sa + tb = 13 \cdot 30 - 8 \cdot 48 = 6 = (a : b)$$

Ejercicios

9. Programar la función `emcd :: Int -> Int -> (Int, Int, Int)` que, dados a y b, utilice el algoritmo de Euclides extendido para obtener una tripla $((a : b), s, t)$ tal que $sa + tb = (a : b)$
10. Definir una función que dados $a \neq 0$ y $b \neq 0$ encuentre el par $s, t \in \mathbb{Z}$ tal que $sa + tb = (a : b)$ donde $s \geq 0$ sea lo mínimo posible. Repasar la teórica para este ejercicio.