Algoritmos para números

Clase 26

IIC 1253

Prof. Cristian Riveros

Recordatorio: Representación de números

Teorema

Sea $b > 1$. Si $n \in \mathbb{N}$, entonces se puede escribir de forma única como:

$$n \;=\; a_{k-1}b^{k-1} + a_{k-2}b^{k-2} + \ldots + a_1 b + a_0 \;=\; \sum_{i=0}^{k-1} a_i b^i$$

- $k \geq 1$,
- $a_0, \ldots, a_{k-1}$ menor que $b \quad (a_i < b) \quad$ y
- $a_{k-1} \neq 0$.

Desde ahora, decimos que la representación de ***n* en base *b*** es la secuencia:

$$(n)_b \;=\; a_{k-1} \ldots a_1 a_0$$

Ejemplo

$$(123)_{10} = 123 \qquad (123)_2 = 1111011 \qquad (123)_8 = 173$$

Recordatorio: Algoritmo para conversión de base

Algoritmo

input : Número $n \in \mathbb{N} - \{0\}$, base $b \geq 2$

output: Una secuencia $(n)_b = a_{k-1} \dots a_1 a_0$

```
Function ConversiónBase (n, b)
    q := n
    k := 0
    while q ≠ 0 do
        a_k := q mod b
        q := q div b
        k := k + 1
    return a_{k-1} ... a_1 a_0
```

¿cuál es el tiempo del algoritmo en términos de n?

Rec: ¿cuál es el tamaño de $(n)_b$ con respecto a n?

Suponga que $|(n)_b| = k$.

- Como n tiene k dígitos en base b, entonces:

$$b^{k-1} \quad \leq \quad n \quad \leq \quad b^k - 1$$

- Despejando k, tenemos que:

$$\log_b(n+1) \quad \leq \quad k \quad \leq \quad \log_b(n) + 1$$

- Como k es un valor entero:

$$\lceil \log_b(n+1) \rceil \quad \leq \quad k \quad \leq \quad \lfloor \log_b(n) + 1 \rfloor$$

Teorema

Para todo $n \in \mathbb{N}$ y $b \geq 2$, se cumple que $|(n)_b| = \lceil \log_b(n+1) \rceil$.

Por lo tanto, $|(n)_b| \in \mathcal{O}(\log(n))$.

Outline

Algoritmos clásicos

Máximo común divisor

¿cómo sumamos dos números en base b?

Considere dos números en base 2 ($b = 2$):

$$\begin{aligned} n &= n_{k-1}2^{k-1} + \ldots + n_1 \cdot 2 + n_0 \\ m &= m_{k-1}2^{k-1} + \ldots + m_1 \cdot 2 + m_0 \end{aligned}$$

Sumando ambos números tenemos que:

$$n + m = \left(n_{k-1} + m_{k-1}\right) \cdot 2^{k-1} + \ldots + \left(n_1 + m_1\right) \cdot 2 + \left(n_0 + m_0\right)$$

¿estamos listos?

¿cómo sumamos dos números en base b?

Sumando ambos números tenemos que:

$$n + m \quad = \quad (n_{k-1} + m_{k-1}) \cdot 2^{k-1} + \ldots + (n_1 + m_1) \cdot 2 + (n_0 + m_0)$$

Para $(n_0 + m_0)$ sabemos que $(n_0 + m_0) = c_0 \cdot 2 + s_0$.

$$n + m \quad = \quad (n_{k-1} + m_{k-1}) \cdot 2^{k-1} + \ldots + (n_1 + m_1 + c_0) \cdot 2 + s_0$$

Para $(n_1 + m_1 + c_0)$ sabemos que $(n_1 + m_1 + c_0) = c_1 \cdot 2 + s_1$.

$$n + m \quad = \quad (n_{k-1} + m_{k-1}) \cdot 2^{k-1} + \ldots (n_2 + m_2 + c_1) \cdot 2^2 + s_1 \cdot 2 + s_0$$

$\ldots$

¿a qué corresponden los valores c_0, c_1, $\ldots$?

¿cómo sumamos dos números en base b?

Por lo tanto, se define recursivamente:

$$\begin{aligned}
n_0 + m_0 &= c_0 \cdot 2 + s_0 \\
n_1 + m_1 + c_0 &= c_1 \cdot 2 + s_1 \\
n_2 + m_2 + c_1 &= c_2 \cdot 2 + s_2 \\
&\dots \\
n_{k-1} + m_{k-1} + c_{k-2} &= c_{k-1} \cdot 2 + s_{k-1}
\end{aligned}$$

Para lo cuál se obtiene:

$$n + m = c_{k-1} \cdot 2^k + s_{k-1} \cdot 2^{k-1} + \ldots + s_1 \cdot 2 + s_0$$

Demuestre que $c_i \leq 1$ (sin importar la base).

$\ldots$ por lo tanto, $|(n+m)_b| \leq \max\{|(n)_b|, |(m)_b|\} + 1$

¿cómo sumamos dos números en base b?

Ejemplo

Considere la suma de $(11)_2 = 1011$ y $(14)_2 = 1110$.

$$\begin{aligned}
1 + 0 &= 0 \cdot 2 + 1 \\
1 + 1 + 0 &= 1 \cdot 2 + 0 \\
0 + 1 + 1 &= 1 \cdot 2 + 0 \\
1 + 1 + 1 &= 1 \cdot 2 + 1 \\
0 + 0 + 1 &= 0 \cdot 2 + 1
\end{aligned}$$

Por lo tanto, $(11 + 14)_2 = 1\,1001$.

Algoritmo de suma de números en base b

Algoritmo

input : Números n y m con $(n)_b = n_{k-1} \dots n_1 n_0$,
$(m)_b = m_{k-1} \dots m_1 m_0$

output: Una secuencia $(n+m)_b = s_k s_{k-1} \dots s_1 s_0$

Function `SumaEnBaseB` *(n, m)*

- $c := 0$
- **for** $j = 0$ **to** $k-1$ **do**
  - $s_j := (n_j + m_j + c) \bmod b$
  - $c := (n_j + m_j + c) \operatorname{div} b$
- $s_k := c$
- **return** $s_k s_{k-1} \dots s_1 s_0$

¿cuál es el **tiempo** del algoritmo en términos de k?

¿cómo multiplicamos dos números en base b?

Considere dos números en base 2 ($b = 2$):

$$\begin{aligned} n &= n_{k-1}2^{k-1} + \ldots + n_1 \cdot 2 + n_0 \\ m &= m_{k-1}2^{k-1} + \ldots + m_1 \cdot 2 + m_0 \end{aligned}$$

Multiplicando ambos números tenemos que:

$$\begin{aligned} n \cdot m &= n \cdot (m_{k-1}2^{k-1} + \ldots + m_1 \cdot 2 + m_0) \\ &= n \cdot (m_{k-1}2^{k-1}) + \ldots + n \cdot (m_1 \cdot 2) + n \cdot (m_0) \end{aligned}$$

¿cuál es el valor de $p_i := n \cdot (m_i \cdot 2^i)$?

¿cómo multiplicamos dos números en base b?

Multiplicando ambos números tenemos que:

$$n \cdot m \quad = \quad n \cdot (m_{k-1}2^{k-1}) + \ldots + n \cdot (m_1 \cdot 2) + n \cdot (m_0)$$

¿cuánto vale $p_i := n \cdot (m_i \cdot 2^i)$?

- Si $m_i = 0$, entonces $p_i := n \cdot (m_i \cdot 2^i) = 0$.
- Si $m_i = 1$, entonces $p_i := n \cdot (m_i \cdot 2^i) = n_{k-1}2^{i+k-1} + \ldots + n_1 \cdot 2^{i+1} + n_0 \cdot 2^i$.

$$(p_i)_2 = \begin{cases} 0 & \text{si } m_i = 0 \\ n_{k-1} \ldots n_1 n_0 \underbrace{0 \ldots 0}_{i-\text{veces}} & \text{si } m_i = 1 \end{cases}$$

Es posible calcular p_i haciendo **shift** i-veces de n.

¿cómo multiplicamos dos números en base b?

Ejemplo

Considere la multiplicación de $(14)_2 = 1110$ y $(11)_2 = 1011$.

$$
\begin{array}{ccccccc}
 & p_0 & = & 1110\cdot(1\cdot 2^0) & = & 1110 \\
 & p_1 & = & 1110\cdot(1\cdot 2^1) & = & 1110\underline{0} \\
 & p_2 & = & 1110\cdot(0\cdot 2^2) & = & 0000\underline{00} \\
+ & p_3 & = & 1110\cdot(1\cdot 2^3) & = & 1110\underline{000} \\
\hline
 & & & p_1+p_2+p_3+p_4 & = & 10011010
\end{array}
$$

Por lo tanto, $(14\cdot 11)_2 = 1001\,1010$.

Algoritmo de multiplicación de números en base b

Algoritmo

input : Números n y m con $(n)_b = n_{k-1} \dots n_1 n_0$,
$(m)_b = m_{k-1} \dots m_1 m_0$
output: Una secuencia $(n \cdot m)_b = p_{2k} \dots p_1 p_0$

Function `MultiplicaciónEnBaseB` (n, m)
  for $i = 0$ **to** $k-1$ **do**
    if $m_i > 0$ **then**
      $p_i := (n \cdot m_i)_b \underbrace{0 \dots 0}_{i-\text{veces}}$
    else
      $p_i := 0$
  $p := 0$
  for $i = 0$ **to** $k-1$ **do**
    $p := p + p_i$
  return $(p)_b$

¿cuál es el **tiempo** del algoritmo en términos de k?

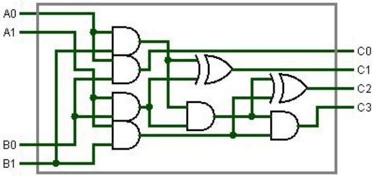
Resumen de tiempos de algoritmos

Teorema

Para números con k-dígitos en base b:

1. Suma: $\mathcal{O}(k)$.
2. Multiplicación: $\mathcal{O}(k^2)$.

¿cómo suma y multiplica el computador?



Outline

Algoritmos clásicos

Máximo común divisor

Máximo común divisor

Definición

Sea $a, b \in \mathbb{Z} - \{0\}$.
Se define el **máximo común divisor** $\gcd(a, b)$ de a, b como el mayor número d tal que $d \mid a$ y $d \mid b$.

Ejemplos

$$\gcd(8, 12) = 4 \qquad \gcd(24, 36) = 12 \qquad \gcd(54, 24) = 6$$

En otras palabras, $\gcd(a, b)$ es el $\leq$-máximo del conjunto:

$$D_{a,b} = \{c \in \mathbb{Z} \mid c \mid a \ \wedge \ c \mid b\}$$

Para $a, b \in \mathbb{Z} - \{0\}$, ¿siempre existe $\gcd(a, b)$?

¿cómo calculamos $\gcd(a, b)$ para a y b?

Supongamos que queremos calcular $\gcd(287, 91)$.

Si dividimos 287 por 91:

$$287 = 91 \cdot 3 + 14$$

¿cuál es la relación entre 287, 91 y 14?

- Si $d \mid 91$ y $d \mid 14$, entonces $d \mid 287$. (¿por qué?)

$$\{\, d \in \mathbb{Z} \mid d \mid 91 \wedge d \mid 14 \,\} \subseteq \{\, d \in \mathbb{Z} \mid d \mid 287 \wedge d \mid 91 \,\}$$

- Si $d \mid 287$ y $d \mid 91$, entonces $d \mid 14$. (¿por qué?)

$$\{\, d \in \mathbb{Z} \mid d \mid 287 \wedge d \mid 91 \,\} \subseteq \{\, d \in \mathbb{Z} \mid d \mid 91 \wedge d \mid 14 \,\}$$

Por lo tanto, $\gcd(287, 91) = \gcd(91, 14)$

¿cómo calculamos $\gcd(a, b)$ para a y b?

Teorema

Para todo $a, b \in \mathbb{Z} - \{0\}$, $\gcd(a, b) \;=\; \gcd(b, (a \bmod b))$.

Demostración: ejercicio.

... ¿para qué nos sirve este resultado?

Ejemplo

$$
\begin{array}{rcl@{\qquad}rcl}
287 & = & 91 \cdot 3 + 14 & \gcd(287, 91) & = & \gcd(91, 14) \\
91 & = & 14 \cdot 6 + 7 & \gcd(91, 14) & = & \gcd(14, 7) \\
14 & = & 7 \cdot 2 & \gcd(14, 7) & = & 7
\end{array}
$$

$$\gcd(287, 91) = \gcd(91, 14) = \gcd(14, 7) = 7$$

¿cómo calculamos $\gcd(a, b)$ para a y b?

Si $r_0 = a$ y $r_1 = b$ con $a \geq b$, se tiene que:

$$\begin{array}{rcll} r_0 & = & r_1 \cdot q_1 + r_2 & 0 \leq r_2 < r_1 \\ r_1 & = & r_2 \cdot q_2 + r_3 & 0 \leq r_3 < r_2 \\ & \vdots & & \vdots \\ r_{n-2} & = & r_{n-1} \cdot q_{n-1} + r_n & 0 \leq r_n < r_{n-1} \\ r_{n-1} & = & r_n \cdot q_n & \end{array}$$

Por el teorema anterior, tenemos que:

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \gcd(r_2, r_3) = \cdots = \gcd(r_{n-1}, r_n) = r_n$$

Este es el **algoritmo de Euclides**.

Algoritmo de máximo común divisor

Algoritmo de Euclides

input : Números a y b con $a \geq b \geq 0$
output: Máximo común divisor entre a y b

Function `MaximoComúnDivisor` (a, b)
  $x := a$
  $y := b$
  while $y \neq 0$ **do**
    $r := x \bmod y$
    $x := y$
    $y := r$
  return x

¿cuál es el **tiempo** del algoritmo de Euclides?

¿cuál es el tiempo del algoritmo de Euclides?

Para $a = r_0$ y $b = r_1$ sabemos que la cantidad de pasos n cumple que:

$$\begin{array}{rcll}
r_0 & = & r_1 \cdot q_1 + r_2 & 0 \leq r_2 < r_1 \\
r_1 & = & r_2 \cdot q_2 + r_3 & 0 \leq r_3 < r_2 \\
& \vdots & & \vdots \\
r_{n-2} & = & r_{n-1} \cdot q_{n-1} + r_n & 0 \leq r_n < r_{n-1} \\
r_{n-1} & = & r_n \cdot q_n &
\end{array}$$

Entonces tenemos que:

$$\begin{array}{rclclcl}
r_n & \geq & 1 & & & = & f_2 \\
r_{n-1} & \geq & 2r_n & \geq & 2f_2 & = & f_3 \\
r_{n-2} & \geq & r_{n-1} + r_n & \geq & f_2 + f_3 & = & f_4 \\
& \vdots & & \vdots & & \vdots & \\
r_2 & \geq & r_3 + r_4 & \geq & f_{n-1} + f_{n-2} & = & f_n \\
r_1 & \geq & r_2 + r_3 & \geq & f_n + f_{n-1} & = & f_{n+1}
\end{array}$$

¿qué relación cumplen los valores f_i?

¿cuál es el tiempo del algoritmo de Euclides?

Lema (Fibonnaci)

Para $n \geq 3$, se cumple que:

$$f_n > \left(\frac{1+\sqrt{5}}{2}\right)^{n-2}$$

Demuestre el lema usando inducción fuerte.

Usando el lema anterior, tenemos que:

$$b \;>\; f_{n+1} > \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}$$

Despejando, obtenemos que $n < \dfrac{\log(b)}{\log(\alpha)} + 1$ con $\alpha = \dfrac{1+\sqrt{5}}{2}$.

Por lo tanto, el **tiempo** del algoritmo de Euclides esta en $\mathcal{O}(\log(b))$.