Operaciones Aritméticas

José Ortiz Bejar

Facultad de Ingeniería Eléctrica
Universidad Michoacana de San Nicolás de Hidalgo

Febrero 21, 2022

Contenido

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : 0, 1, 2, . . . , 9 (¿Porqué diez?)

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : $0, 1, 2, \ldots, 9$ (¿Porqué diez?)
  - *polinomial* con base $b = 10$

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : 0, 1, 2, . . . , 9 (¿Porqué diez?)
  - *polinomial* con base $b = 10$
- Por ejemplo

$n =$ | 5 | 3 | 7 | 2 | 0 |

$$n = 5 \times 10000 + 3 \times 1000 + 7 \times 1002 \times 10 + 0 \times 1$$

- ¿Cómo representan los números las computadoras?

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : 0, 1, 2, . . . , 9 (¿Porqué diez?)
  - *polinomial* con base $b = 10$
- Por ejemplo

	d_4	d_3	d_2	d_1	d_0
$n =$	5	3	7	2	0

$$n = 5 \times 10000 + 3 \times 1000 + 7 \times 1002 \times 10 + 0 \times 1$$

- ¿Cómo representan los números las computadoras?

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : 0, 1, 2, . . . , 9 (¿Porqué diez?)
  - *polinomial* con base $b = 10$
- Por ejemplo

$$0 \leq d_i < b$$

	d_4	d_3	d_2	d_1	d_0
$n =$	5	3	7	2	0

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : 0, 1, 2, . . . , 9 (¿Porqué diez?)
  - *polinomial* con base $b = 10$
- Por ejemplo

$$\begin{array}{rc} & \begin{array}{ccccc} d_4 & d_3 & d_2 & d_1 & d_0 \end{array} \\ n = & \begin{array}{|c|c|c|c|c|} \hline 5 & 3 & 7 & 2 & 0 \\ \hline \end{array} \end{array} \qquad 0 \leq d_i < b$$

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

$$n = 5 \times 10000 + 3 \times 1000 + 7 \times 1002 \times 10 + 0 \times 1$$

Como representamos los números

- ¿Qué representación utilizamos los seres humanos?
- Utilizamos ***notación decimal***
  - diez símbolos : 0, 1, 2, . . . , 9 (¿Porqué diez?)
  - *polinomial* con base $b = 10$
- Por ejemplo

$$\begin{array}{rc} & \begin{array}{ccccc} d_4 & d_3 & d_2 & d_1 & d_0 \end{array} \\ n = & \begin{array}{|c|c|c|c|c|} \hline 5 & 3 & 7 & 2 & 0 \\ \hline \end{array} \end{array} \qquad 0 \leq d_i < b$$

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

$$n = 5 \times 10000 + 3 \times 1000 + 7 \times 1002 \times 10 + 0 \times 1$$

- ¿Cómo representan los números las computadoras?

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$
- Por ejemplo

$n =$ | 1 | 1 | 0 | 0 | 1 |

$$n = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$n = 1 \times 16_{base_{10}} + 1 \times 8_{base_{10}} + 0 \times 4_{base_{10}} + 0 \times 2_{base_{10}} + 1 \times 1_{base_{10}}$$

- Las preguntas que deberíamos hacernos
  - ¿Es una representación adecuada?
  - ¿Cuál es su costo? (en términos de espacio)
  - ¿Existe una representación mejor?

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$
- Por ejemplo

$n =$

d_4	d_3	d_2	d_1	d_0
1	1	0	0	1

$$n = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$n = 1 \times 16_{base_{10}} + 1 \times 8_{base_{10}} + 0 \times 4_{base_{10}} + 0 \times 2_{base_{10}} + 1 \times 1_{base_{10}}$$

- Las preguntas que deberíamos hacernos
  - ¿Es una representación adecuada?
  - ¿Cuál es su costo? (en términos de espacio)
  - ¿Existe una representación mejor?

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$
- Por ejemplo

$$n = \begin{array}{|c|c|c|c|c|} \hline d_4 & d_3 & d_2 & d_1 & d_0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline \end{array} \qquad 0 \leq d_i < b$$

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$
- Por ejemplo

$$n = \begin{array}{|c|c|c|c|c|} \hline d_4 & d_3 & d_2 & d_1 & d_0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline \end{array} \qquad 0 \leq d_i < b$$

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

$$n = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$
- Por ejemplo

$$n = \begin{array}{|c|c|c|c|c|} \hline d_4 & d_3 & d_2 & d_1 & d_0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline \end{array} \qquad 0 \leq d_i < b$$

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

$$n = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$n = 1 \times 16_{base_{10}} + 1 \times 8_{base_{10}} + 0 \times 4_{base_{10}} + 0 \times 2_{base_{10}} + 1 \times 1_{base_{10}}$$

Representado números en una computadora

- Las computadoras utilizan una ***representación binaria***
  - los símbolos : 0, 1 (¿Porqué dos?)
  - un *polinomial* con base $b = 2$
- Por ejemplo

$$n = \begin{array}{|c|c|c|c|c|} \hline d_4 & d_3 & d_2 & d_1 & d_0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline \end{array} \qquad 0 \leq d_i < b$$

$$d_4 b^4 + d_3 b^3 + d_2 b^2 + d_1 b^1 + d_0 b^0$$

$$n = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

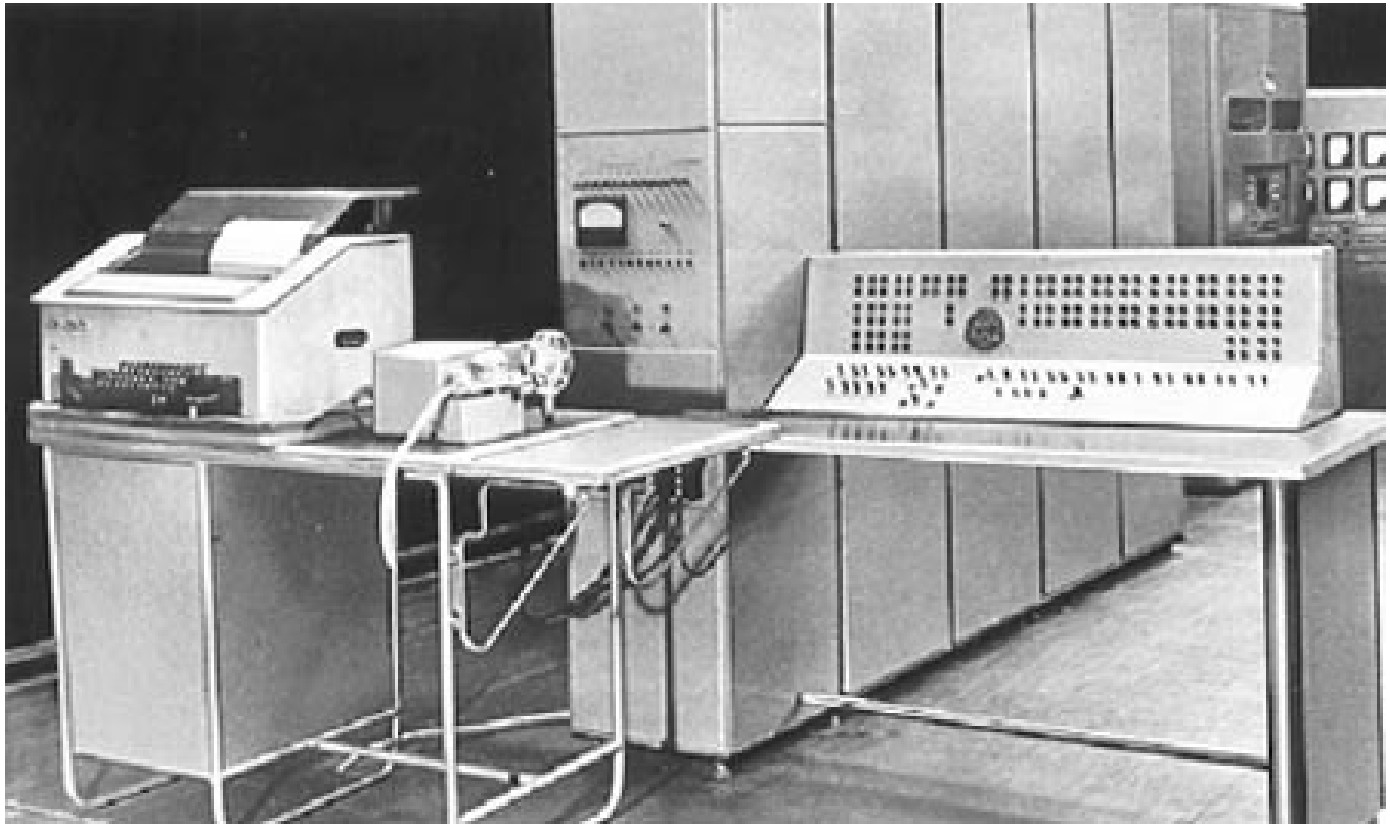
$$n = 1 \times 16_{base_{10}} + 1 \times 8_{base_{10}} + 0 \times 4_{base_{10}} + 0 \times 2_{base_{10}} + 1 \times 1_{base_{10}}$$

- Las preguntas que deberíamos hacernos
  - ¿Es una representación adecuada?
  - ¿Cuál es su costo? (en términos de espacio)
  - ¿Existe una representación mejor?

¿Una computadora *Ternaria*?

¿Una computadora *Ternaria*?

- Sí se construyo una computadora *ternaria*



La computadora denominada ***Setun*** fue una computadora *ternaria* desarrollada por la Universidad Estatal de Moscú en 1958.

Corrección

- ¿Qué significa?

Corrección

- ¿Qué significa?
- Existe *al menos una* representación para cada valor.

Corrección

- ¿Qué significa?
- Existe *al menos una* representación para cada valor.
- Un representación no debe ser ambigua.
  - e.i., existe *a lo más* un valor para cada representación.

Corrección

- ¿Qué significa?
- Existe *al menos una* representación para cada valor.
- Un representación no debe ser ambigua.
  - e.i., existe *a lo más* un valor para cada representación.
- No profundizaremos más al respecto de este punto.

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?
- Con ℓ dígitos se tiene
  $N \leq d_{\ell-1} b^{\ell-1} + d_{\ell-2} b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?
- Con ℓ dígitos se tiene
  $N \leq d_{\ell-1} b^{\ell-1} + d_{\ell-2} b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

$$\begin{aligned} N &\leq (b-1)b^{\ell-1} + (b-1)b^{\ell-2} + \cdots + (b-1)b^1 + b - 1 \\ &= b^\ell - b^{\ell-1} + b^{\ell-1} - b^{\ell-2} + b^{\ell-2} - \cdots - b + b - 1 \\ &= b^\ell - 1 \end{aligned}$$

- Con ℓ dígitos, el valor más grande que podemos representar es $N = b^\ell - 1$
- Entonces, $\ell = \lceil \log_b (N+1) \rceil \quad \Rightarrow \quad \ell = \Theta(\log N)$

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?
- Con ℓ dígitos se tiene
  $N \leq d_{\ell-1} b^{\ell-1} + d_{\ell-2} b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

$$
\begin{aligned}
N &\leq (b-1)b^{\ell-1} + (b-1)b^{\ell-2} + \cdots + (b-1)b^1 + b - 1 \\
&= b^\ell - b^{\ell-1} + b^{\ell-1} - b^{\ell-2} + b^{\ell-2} - \cdots - b + b - 1 \\
&= b^\ell - 1
\end{aligned}
$$

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?
- Con ℓ dígitos se tiene
  $N \leq d_{\ell-1} b^{\ell-1} + d_{\ell-2} b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

$$
\begin{aligned}
N &\leq (b-1) b^{\ell-1} + (b-1) b^{\ell-2} + \cdots + (b-1) b^1 + b - 1 \\
&= b^\ell - b^{\ell-1} + b^{\ell-1} - b^{\ell-2} + b^{\ell-2} - \cdots - b + b - 1 \\
&= b^\ell - 1
\end{aligned}
$$

- Con ℓ dígitos, el valor más grande que podemos representar es $N = b^\ell - 1$

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?
- Con ℓ dígitos se tiene
  $N \leq d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

$$\begin{aligned} N &\leq (b-1)b^{\ell-1} + (b-1)b^{\ell-2} + \cdots + (b-1)b^1 + b - 1 \\ &= b^\ell - b^{\ell-1} + b^{\ell-1} - b^{\ell-2} + b^{\ell-2} - \cdots - b + b - 1 \\ &= b^\ell - 1 \end{aligned}$$

- Con ℓ dígitos, el valor más grande que podemos representar es $N = b^\ell - 1$
- Entonces, $\ell = \lceil \log_b (N+1) \rceil$

Complejidad en espacio

- ¿Cuantos dígitos requerimos para representar un número N?
- ¿Cuál es número N más grande que podemos representar utilizando ℓ dígitos?
- Con ℓ dígitos se tiene
  $N \leq d_{\ell-1} b^{\ell-1} + d_{\ell-2} b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

$$\begin{aligned} N &\leq (b-1)b^{\ell-1} + (b-1)b^{\ell-2} + \cdots + (b-1)b^1 + b - 1 \\ &= b^\ell - b^{\ell-1} + b^{\ell-1} - b^{\ell-2} + b^{\ell-2} - \cdots - b + b - 1 \\ &= b^\ell - 1 \end{aligned}$$

- Con ℓ dígitos, el valor más grande que podemos representar es $N = b^\ell - 1$
- Entonces, $\ell = \lceil \log_b (N+1) \rceil \quad \Rightarrow \quad \ell = \Theta(\log N)$

Complejidad en espacio (2)

- La ***complejidad en espacio*** para N es $\ell = \Theta(\log N)$

Complejidad en espacio (2)

- La ***complejidad en espacio*** para N es $\ell = \Theta(\log N)$
- ¿Es posible hacerlo mejor?

Complejidad en espacio (2)

- La ***complejidad en espacio*** para N es $\ell = \Theta(\log N)$
- ¿Es posible hacerlo mejor?
- No!

Complejidad en espacio (2)

- La ***complejidad en espacio*** para N es $\ell = \Theta(\log N)$
- ¿Es posible hacerlo mejor?
- No!
  - existente exactamente b^ℓ combinaciones de ℓ dígitos tomados de un alfabeto Σ con cardinalidad $|\Sigma| = b$
  - e.i., no se pueden *expresar* más de b^ℓ valores con ℓ dígitos

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

- **Teorema:** la suma de tres dígitos base-b (cuyos valores son $x, y, z \leq b - 1$) puede ser representada con be *dos* dígitos base-b.

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

- **Teorema:** la suma de tres dígitos base-b (cuyos valores son $x, y, z \leq b - 1$) puede ser representada con be *dos* dígitos base-b.

  Prueba:

  - *caso* $b = 2$: $\ell = 2$ dado que $1 + 1 + 1 = 3_{\text{base}_{10}} = 11_{\text{base}_2}$

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

- **Teorema:** la suma de tres dígitos base-b (cuyos valores son $x, y, z \leq b - 1$) puede ser representada con be *dos* dígitos base-b.

  Prueba:

  - *caso* $b = 2$*:* $\ell = 2$ dado que $1 + 1 + 1 = 3_{\text{base}_{10}} = 11_{\text{base}_2}$
  - *caso* $b \geq 3$*:*

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

- **Teorema:** la suma de tres dígitos base-b (cuyos valores son $x, y, z \leq b - 1$) puede ser representada con be *dos* dígitos base-b.

  Prueba:

  - *caso* $b = 2$*:* $\ell = 2$ dado que $1 + 1 + 1 = 3_{\text{base}_{10}} = 11_{\text{base}_2}$
  - *caso* $b \geq 3$*:*

    1. podemos representar $x + y + z$ con $\ell = \lceil \log_b (x + y + z + 1) \rceil$

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

- **Teorema:** la suma de tres dígitos base-b (cuyos valores son $x, y, z \leq b - 1$) puede ser representada con be *dos* dígitos base-b.

  Prueba:

  - *caso* $b = 2$: $\ell = 2$ dado que $1 + 1 + 1 = 3_{\text{base}_{10}} = 11_{\text{base}_2}$
  - *caso* $b \geq 3$:

    1. podemos representar $x + y + z$ con $\ell = \lceil \log_b (x + y + z + 1) \rceil$
    2. $x + y + z + 1 \leq 3b$, debido a que x, y, z son dígitos base-b, entonces $\ell = \lceil \log_b (x + y + z + 1) \rceil \leq \log_b 3b = \log_b 3 + 1$

Suma de números

- Se pueden representar dos números n_1 y n_2 co $\Theta(\log n_1)$ y $\Theta(\log n_2)$ bits, respectivamente.

- ¿Cómo *sumamos* dos números representados en una notación base-b?

- **Teorema:** la suma de tres dígitos base-b (cuyos valores son $x, y, z \leq b - 1$) puede ser representada con be *dos* dígitos base-b.

  Prueba:

  - *caso* $b = 2$: $\ell = 2$ dado que $1 + 1 + 1 = 3_{\text{base}_{10}} = 11_{\text{base}_2}$
  - *caso* $b \geq 3$:

    1. podemos representar $x + y + z$ con $\ell = \lceil \log_b (x + y + z + 1) \rceil$
    2. $x + y + z + 1 \leq 3b$, debido a que x, y, z son dígitos base-b, entonces $\ell = \lceil \log_b (x + y + z + 1) \rceil \leq \log_b 3b = \log_b 3 + 1$
    3. $b \geq 3$, implica que $\log_b 3 \leq 1$, por tanto $\ell \leq 2$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-b dígitos será un número de la los más *dos* dígitos base-b, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados x y y

$$\begin{aligned} x &= & x_{\ell-1} & \quad \cdots & x_1 & \quad x_0 \\ y &= & y_{\ell-1} & \quad \cdots & y_1 & \quad y_0 \\ x+y &= & & & & \end{aligned}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-b dígitos será un número de la los más *dos* dígitos base-b, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados x y y

$$\begin{array}{rccccc} & & & & & 0 \\ x = & & x_{\ell-1} & \cdots & x_1 & x_0 \\ y = & & y_{\ell-1} & \cdots & y_1 & y_0 \\ x+y = & & & & & \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-*b* dígitos será un número de la los más *dos* dígitos base-*b*, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados *x* y *y*

$$\begin{array}{rccccc} & & & & c_0 & 0 \\ x = & x_{\ell-1} & \cdots & x_1 & x_0 \\ y = & y_{\ell-1} & \cdots & y_1 & y_0 \\ x+y = & & & & & s_0 \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-*b* dígitos será un número de la los más *dos* dígitos base-*b*, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados *x* y *y*

$$\begin{array}{rccccc} & & & & c_0 & 0 \\ x = & & x_{\ell-1} & \cdots & x_1 & x_0 \\ y = & & y_{\ell-1} & \cdots & y_1 & y_0 \\ x+y = & & & & & s_0 \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-*b* dígitos será un número de la los más *dos* dígitos base-*b*, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados *x* y *y*

$$\begin{array}{rccccc} & & & & c_1 & c_0 & 0 \\ x = & x_{\ell-1} & \cdots & & & x_1 & x_0 \\ y = & y_{\ell-1} & \cdots & & & y_1 & y_0 \\ x+y = & & & & & s_1 & s_0 \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-b dígitos será un número de la los más *dos* dígitos base-b, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados x y y

$$\begin{array}{rccccc} & & & & c_1 & c_0 & 0 \\ x = & & x_{\ell-1} & \cdots & x_1 & x_0 \\ y = & & y_{\ell-1} & \cdots & y_1 & y_0 \\ x+y = & & & & s_1 & s_0 \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-b dígitos será un número de la los más *dos* dígitos base-b, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados x y y

$$\begin{array}{rccccccc} & c_{\ell-1} & c_{\ell-2} & & & c_1 & c_0 & 0 \\ x = & & x_{\ell-1} & \cdots & & x_1 & x_0 & \\ y = & & y_{\ell-1} & \cdots & & y_1 & y_0 & \\ x+y = & & s_{\ell-1} & \cdots & & s_1 & s_0 & \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-b dígitos será un número de la los más *dos* dígitos base-b, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados x y y

$$\begin{array}{rccccccc} & & & & c_1 & c_0 & 0 \\ x = & & x_{\ell-1} & \cdots & & x_1 & x_0 \\ y = & & y_{\ell-1} & \cdots & & y_1 & y_0 \\ x+y = & c_{\ell-1} & s_{\ell-1} & \cdots & & s_1 & s_0 \end{array}$$

Suma de números (2)

- Sabemos que la suma de tres dígitos base-b dígitos será un número de la los más *dos* dígitos base-b, esto lo usaremos como un bloque básico para definir.

$$(x_i, y_i, z_i) \longrightarrow (c, s_i)$$

- dados x y y

$$\begin{array}{rccccc} & & & c_1 & c_0 & 0 \\ x = & x_{\ell-1} & \cdots & x_1 & x_0 & \\ y = & y_{\ell-1} & \cdots & y_1 & y_0 & \\ x+y = & c_{\ell-1} & s_{\ell-1} & \cdots & s_1 & s_0 \end{array}$$

$$\ell + 1$$

Suma de números (3)

- Dados dos *arreglos* de ℓ base-b dígitos, A y B

Suma de números (3)

- Dados dos *arreglos* de ℓ base-b dígitos, A y B

```
Add(A, B) 1  R = 0 // [0, ..., 0] of size ℓ + 1
          2  c = 0
          3  for i = 1 to ℓ
          4        (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5  R[ℓ + 1] = c
          6  return R
```

Suma de números (3)

- Dados dos *arreglos* de ℓ base-b dígitos, A y B

```
Add(A, B) 1   R = 0 // [0, ..., 0] of size ℓ + 1
          2   c = 0
          3   for i = 1 to ℓ
          4         (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5   R[ℓ + 1] = c
          6   return R
```

- ¿El algoritmo es correcto?

Suma de números (3)

- Dados dos *arreglos* de ℓ base-b dígitos, A y B

```
Add(A, B) 1   R = 0 // [0, ..., 0] of size ℓ + 1
          2   c = 0
          3   for i = 1 to ℓ
          4         (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5   R[ℓ + 1] = c
          6   return R
```

- ¿El algoritmo es correcto? Sí

Suma de números (3)

- Dados dos *arreglos* de ℓ base-b dígitos, A y B

```
Add(A, B) 1   R = 0 // [0, ..., 0] of size ℓ + 1
          2   c = 0
          3   for i = 1 to ℓ
          4         (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5   R[ℓ + 1] = c
          6   return R
```

- ¿El algoritmo es correcto? Sí
- ¿Cuál es el costo computacional?
- ¿Es posible hacerlo mejor?

Complejidad

- Estamos interesados en determinar $T(\ell)$ (recuerda que $\ell = \Theta(\log N)$)

Complejidad

- Estamos interesados en determinar $T(\ell)$ (recuerda que $\ell = \Theta(\log N)$)

```
Add(A, B) 1  R = 0 // [0, ..., 0] of size ℓ + 1
          2  c = 0
          3  for i = 1 to ℓ
          4        (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5  R[ℓ + 1] = c
          6  return R
```

- Estamos interesados en determinar $T(\ell)$ (recuerda que $\ell = \Theta(\log N)$)

```
Add(A, B) 1  R = 0 // [0, ..., 0] of size ℓ + 1
          2  c = 0
          3  for i = 1 to ℓ
          4        (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5  R[ℓ + 1] = c
          6  return R
```

$$T(\ell) = \Theta(\ell)$$

Complejidad

- Estamos interesados en determinar $T(\ell)$ (recuerda que $\ell = \Theta(\log N)$)

```
Add(A, B) 1  R = 0 // [0, ..., 0] of size ℓ + 1
          2  c = 0
          3  for i = 1 to ℓ
          4        (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5  R[ℓ + 1] = c
          6  return R
```

$$T(\ell) = \Theta(\ell)$$

- ¿Es posible hacerlo mejor?

Complejidad

- Estamos interesados en determinar $T(\ell)$ (recuerda que $\ell = \Theta(\log N)$)

```
Add(A, B) 1   R = 0 // [0, ..., 0] of size ℓ + 1
          2   c = 0
          3   for i = 1 to ℓ
          4         (c, R[i]) = AddThreeDigits(A[i], B[i], c)
          5   R[ℓ + 1] = c
          6   return R
```

$$T(\ell) = \Theta(\ell)$$

- ¿Es posible hacerlo mejor? No!
  - debemos notar que tenemos ℓ dígitos de entrada
  - por lo que debemos asignar al menos $\ell + 1$ para almacenar el resultado.

Multiplicando Números

- Ya sabemos como sumar dos números.
- ¿ Como podemos realizar una *multiplicación* de dos números?

Multiplicando Números

- Ya sabemos como sumar dos números.
- ¿ Como podemos realizar una *multiplicación* de dos números?
- Recuerden que la representación que estamos utilizando es polinomial.

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

Multiplicando Números

- Ya sabemos como sumar dos números.
- ¿ Como podemos realizar una *multiplicación* de dos números?
- Recuerden que la representación que estamos utilizando es polinomial.

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

Si multiplicamos x por otro polinomio en b, por ejemplo $y = y_i b^i$, obtenemos

$$x \times y = y_i \left(x_{\ell-1} b^{\ell-1+i} + x_{\ell-2} b^{\ell-2+i} + \cdots + x_1 b^{i+1} + x_0 b^i \right)$$

Multiplicando Números

- Ya sabemos como sumar dos números.
- ¿ Como podemos realizar una *multiplicación* de dos números?
- Recuerden que la representación que estamos utilizando es polinomial.

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

Si multiplicamos x por otro polinomio en b, por ejemplo $y = y_i b^i$, obtenemos

$$x \times y = y_i (x_{\ell-1} b^{\ell-1+i} + x_{\ell-2} b^{\ell-2+i} + \cdots + x_1 b^{i+1} + x_0 b^i)$$

- Multiplicar por b^i es equivalente a *desplazar* nuestra representación i posiciones a la izquierda
  - la *izquierda* está en dirección del dígito más significativo bits

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

- Por ejemplo, sea $x = 1001_{\text{base}_2}$ y $y = 1011_{\text{base}_2}$

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

- Por ejemplo, sea $x = 1001_{\text{base}_2}$ y $y = 1011_{\text{base}_2}$

$x \times y =$

$$1 \quad 0 \quad 0 \quad 1 \quad (1001 \times 1)$$

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

- Por ejemplo, sea $x = 1001_{\text{base}_2}$ y $y = 1011_{\text{base}_2}$

$x \times y =$

			1	0	0	1	(1001×1)
+		1	0	0	1		$(1001 \times 1$ desplazado 1)

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

- Por ejemplo, sea $x = 1001_{\text{base}_2}$ y $y = 1011_{\text{base}_2}$

$x \times y =$

			1	0	0	1	(1001 × 1)
+		1	0	0	1		(1001 × 1 desplazado 1)
+	0	0	0	0			(1001 × 0 desplazado 2)

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

- Por ejemplo, sea $x = 1001_{\text{base}_2}$ y $y = 1011_{\text{base}_2}$

$x \times y =$

				1	0	0	1	(1001×1)
+			1	0	0	1		$(1001 \times 1$ desplazado 1)
+		0	0	0	0			$(1001 \times 0$ desplazado 2)
+	1	0	0	1				$(1001 \times 1$ desplazado 3)

Multiplicando Números Binarios

- Ahora revisemos que ocurre en números binarios (e.i., base $b = 2$)

$$x = x_{\ell-1} b^{\ell-1} + x_{\ell-2} b^{\ell-2} + \cdots + x_1 b + x_0$$

donde cada x_i es 0 o 1

- Por ejemplo, sea $x = 1001_{\text{base}_2}$ y $y = 1011_{\text{base}_2}$

$x \times y =$

					1	0	0	1	(1001 × 1)
+				1	0	0	1		(1001 × 1 desplazado 1)
+			0	0	0	0			(1001 × 0 desplazado 2)
+		1	0	0	1				(1001 × 1 desplazado 3)
=	1	1	1	0	0	0	1	1	(1001 × 1011)

Multiplicando Números (2)

- Dados dos *arreglos* de ℓ dígitos binarios, A y B

Multiplicando Números (2)

- Dados dos *arreglos* de ℓ dígitos binarios, A y B

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 1 to ℓ
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

Multiplicando Números (2)

- Dados dos *arreglos* de ℓ dígitos binarios, A y B

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 1 to ℓ
4        if B[i] == 1
5              R = Add(R, T)
6        T = ShiftLeft(T)
7  return R
```

- ¿Es correcto?

Multiplicando Números (2)

- Dados dos *arreglos* de ℓ dígitos binarios, A y B

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 1 to ℓ
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

- ¿Es correcto? Sí

Multiplicando Números (2)

- Dados dos *arreglos* de ℓ dígitos binarios, A y B

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 1 to ℓ
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

- ¿Es correcto? Sí
- ¿Cuál es su costo?
- ¿Podemos hacerlo mejor?

Complejidad

- Nuevamente estamos interesados en $T(\ell)$

- Nuevamente estamos interesados en $T(\ell)$

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 0 to ℓ − 1
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

- Nuevamente estamos interesados en $T(\ell)$

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 0 to ℓ − 1
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

$$T(\ell) = \Theta(\ell^2)$$

- Nuevamente estamos interesados en $T(\ell)$

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 0 to ℓ − 1
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

$$T(\ell) = \Theta(\ell^2)$$

- ¿Podemos hacermos mejor?

- Nuevamente estamos interesados en $T(\ell)$

```
Multiply(A, B)
1  R = 0
2  T = A
3  for i = 0 to ℓ − 1
4       if B[i] == 1
5            R = Add(R, T)
6       T = ShiftLeft(T)
7  return R
```

$$T(\ell) = \Theta(\ell^2)$$

- ¿Podemos hacermos mejor? Sí

Una mejor forma de Multiplicar

Una mejor forma de Multiplicar

- Intuición: dividimos la representación de los números a la mitad

Una mejor forma de Multiplicar

- Intuición: dividimos la representación de los números a la mitad

$x =$ $\boxed{X_L}$ $\boxed{X_R}$ y $y =$ $\boxed{Y_L}$ $\boxed{Y_R}$

Una mejor forma de Multiplicar

- Intuición: dividimos la representación de los números a la mitad

  $x =$ | X_L | X_R | y $y =$ | Y_L | Y_R |

  esto implica que $x = 2^{\ell/2}x_L + x_R$ y $y = 2^{\ell/2}y_L + y_R$, so...

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

  Reducimos el problema de multiplicas dos números de ℓ bits a un problema de multiplicar *cuatro* números de $\ell/2$ bits...

Una mejor forma de Multiplicar

- Intuición: dividimos la representación de los números a la mitad

  $x =$ | X_L | X_R | y $y =$ | Y_L | Y_R |

  esto implica que $x = 2^{\ell/2} x_L + x_R$ y $y = 2^{\ell/2} y_L + y_R$, so...

$$\begin{aligned} xy &= (2^{\ell/2} x_L + x_R)(2^{\ell/2} y_L + y_R) \\ &= 2^{\ell} x_L y_L + 2^{\ell/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

  Reducimos el problema de multiplicas dos números de ℓ bits a un problema de multiplicar *cuatro* números de $\ell/2$ bits...

$$T(\ell) = 4T(\ell/2) + O(\ell)$$

Una mejor forma de Multiplicar

- Intuición: dividimos la representación de los números a la mitad

  $x =$ | X_L | X_R | y $y =$ | Y_L | Y_R |

  esto implica que $x = 2^{\ell/2} x_L + x_R$ y $y = 2^{\ell/2} y_L + y_R$, so...

$$\begin{aligned} xy &= (2^{\ell/2} x_L + x_R)(2^{\ell/2} y_L + y_R) \\ &= 2^{\ell} x_L y_L + 2^{\ell/2}(x_L y_R + x_R y_L) + x_R y_R \end{aligned}$$

  Reducimos el problema de multiplicas dos números de ℓ bits a un problema de multiplicar *cuatro* números de $\ell/2$ bits...

$$T(\ell) = 4T(\ell/2) + O(\ell)$$

$$T(\ell) = \Theta(\ell^2)$$

Una mejor forma de Multiplicar (2)

Una mejor forma de Multiplicar (2)

- Nuevamente, tenemos

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

Una mejor forma de Multiplicar (2)

- Nuevamente, tenemos

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

  pero note que $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, entonces

Una mejor forma de Multiplicar (2)

- Nuevamente, tenemos

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

pero note que $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, entonces

$$xy = 2^{\ell}x_Ly_L + 2^{\ell/2}((x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R) + x_Ry_R$$

Una mejor forma de Multiplicar (2)

- Nuevamente, tenemos

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

pero note que $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, entonces

$$xy = 2^{\ell}x_Ly_L + 2^{\ell/2}((x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R) + x_Ry_R$$

Solo tenemos 3 multiplicaciones: x_Ly_L, $(x_L + x_R)(y_R + y_L)$, y x_Ry_R

Una mejor forma de Multiplicar (2)

- Nuevamente, tenemos

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

pero note que $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, entonces

$$xy = 2^{\ell}x_Ly_L + 2^{\ell/2}((x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R) + x_Ry_R$$

Solo tenemos 3 multiplicaciones: x_Ly_L, $(x_L + x_R)(y_R + y_L)$, y x_Ry_R

$$T(\ell) = 3T(\ell/2) + O(\ell)$$

Una mejor forma de Multiplicar (2)

- Nuevamente, tenemos

$$\begin{aligned} xy &= (2^{\ell/2}x_L + x_R)(2^{\ell/2}y_L + y_R) \\ &= 2^{\ell}x_Ly_L + 2^{\ell/2}(x_Ly_R + x_Ry_L) + x_Ry_R \end{aligned}$$

pero note que $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, entonces

$$xy = 2^{\ell}x_Ly_L + 2^{\ell/2}((x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R) + x_Ry_R$$

Solo tenemos 3 multiplicaciones: x_Ly_L, $(x_L + x_R)(y_R + y_L)$, y x_Ry_R

$$T(\ell) = 3T(\ell/2) + O(\ell)$$

lo cual, nos lleva a la siguiente complejidad.

$$T(\ell) = O(\ell^{\log_2 3}) = O(\ell^{1.59})$$