Arithmetic Operations

Antonio Carzaniga

Faculty of Informatics
Università della Svizzera italiana

March 2, 2012

Outline

- Representing numbers
- Adding numbers
- Multiplying numbers

Representing Numbers

- How do we (human beings) represent numbers?

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: 0, 1, 2, . . . , 9 (why ten?)

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$n = 3 \;\; 1 \;\; 4 \;\; 1 \;\; 5 \;\; 9 \;\; 2$$

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$n = \boxed{3}\boxed{1}\boxed{4}\boxed{1}\boxed{5}\boxed{9}\boxed{2}$$

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$
\begin{array}{rc}
 & \begin{array}{ccccccc} d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \end{array} \\
n = & \begin{array}{|c|c|c|c|c|c|c|} \hline 3 & 1 & 4 & 1 & 5 & 9 & 2 \\ \hline \end{array}
\end{array}
$$

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$\begin{array}{cccccccc} & d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \\ n = & \boxed{3} & \boxed{1} & \boxed{4} & \boxed{1} & \boxed{5} & \boxed{9} & \boxed{2} \end{array}$$

$$0 \leq d_i < b$$

$$n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0$$

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$\begin{array}{cccccccc} & d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \\ n = & \boxed{3} & \boxed{1} & \boxed{4} & \boxed{1} & \boxed{5} & \boxed{9} & \boxed{2} \end{array}$$

$$0 \leq d_i < b$$

$$n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0$$

$n =$

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$\begin{array}{rl} & d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0 \\ n = & \boxed{3}\boxed{1}\boxed{4}\boxed{1}\boxed{5}\boxed{9}\boxed{2} \end{array} \qquad \begin{array}{l} 0 \le d_i < b \\ n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0 \end{array}$$

$$n = 3 \times 1000000 + 1 \times 100000 + 4 \times 10000 + 1 \times 1000 +$$
$$+5 \times 100 + 9 \times 10 + 2 \times 1 = 3141592$$

Representing Numbers

- How do we (human beings) represent numbers?

- Using the ***decimal notation***
  - ten symbols: $0, 1, 2, \ldots, 9$ (why ten?)
  - a *polynomial* in $b = 10$

- For example

$$\begin{array}{cc} & \begin{array}{ccccccc} d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \end{array} \\ n = & \begin{array}{|c|c|c|c|c|c|c|} \hline 3 & 1 & 4 & 1 & 5 & 9 & 2 \\ \hline \end{array} \end{array} \qquad \begin{array}{l} 0 \le d_i < b \\ n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0 \end{array}$$

$$n = 3 \times 1000000 + 1 \times 100000 + 4 \times 10000 + 1 \times 1000 +$$
$$+5 \times 100 + 9 \times 10 + 2 \times 1 = 3141592$$

- How do computers represent numbers?

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$n = 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1$$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$n = \boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{1}$$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$\begin{array}{c c c c c c c c} & d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \\ n = & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \end{array}$$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$\begin{array}{rl} & d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0 \\ n = & \boxed{1\,|\,0\,|\,1\,|\,1\,|\,0\,|\,1\,|\,1} \end{array} \qquad \begin{array}{l} 0 \le d_i < b \\ n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0 \end{array}$$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$\begin{array}{rl} & d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0 \\ n = & \boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{1} \end{array} \qquad \begin{array}{l} 0 \leq d_i < b \\ n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0 \end{array}$$

$n =$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$\begin{array}{cccccccc} & d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \\ n = & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} \end{array}$$

$$0 \le d_i < b$$

$$n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0$$

$$n = 1 \times 64_{\text{ten}} + 1 \times 16_{\text{ten}} + 1 \times 8_{\text{ten}} + 1 \times 2_{\text{ten}} + 1 \times 1_{\text{ten}} = \ldots$$

Representing Numbers in a Computer

- Computers work well with the ***binary representation***
  - two symbols: 0, 1 (why two?)
  - a *polynomial* in $b = 2$

- For example

$$\begin{array}{ll} & d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0 \\ n = & \boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{1} \end{array} \qquad \begin{array}{l} 0 \le d_i < b \\ n = d_6 b^6 + d_5 b^5 + \cdots + d_1 b^1 + d_0 b^0 \end{array}$$

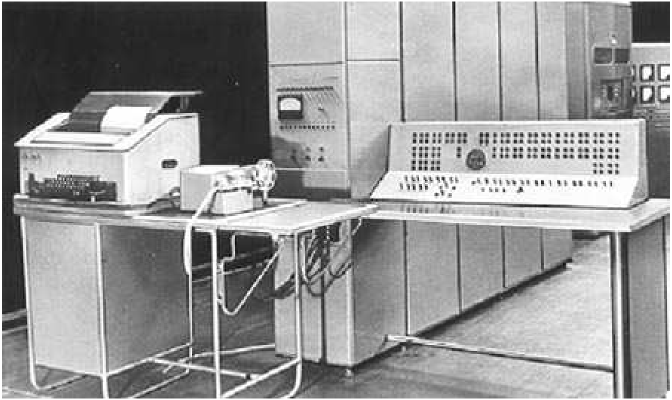
$$n = 1 \times 64_{\text{ten}} + 1 \times 16_{\text{ten}} + 1 \times 8_{\text{ten}} + 1 \times 2_{\text{ten}} + 1 \times 1_{\text{ten}} = \ldots$$

- The usual questions
  - Is this representation correct?
  - How much does it cost? (This time in terms of space)
  - Can we do better?

A *Ternary* Computer?

A *Ternary* Computer?

- A *ternary* computer was actually built!



The ***Setun*** was a *ternary* (or *trinary*) computer developed in 1958 and used at Moscow State University until about 1965

Correctness

- What does it even mean?

Correctness

- What does it even mean?
- There is *at least one* representation for each value

Correctness

- What does it even mean?

- There is *at least one* representation for each value

- A representation should be unambiguous
  - i.e., there is *at most one* value for each representation

Correctness

- What does it even mean?
- There is *at least one* representation for each value
- A representation should be unambiguous
  - i.e., there is *at most one* value for each representation
- We don’t care to say more about this

Space Complexity

- How many symbols do we need to represent a value *N*?

Space Complexity

- How many symbols do we need to represent a value N?
- Or, what is the largest N we can represent with ℓ symbols?

Space Complexity

- How many symbols do we need to represent a value N?

- Or, what is the largest N we can represent with ℓ symbols?

- With ℓ symbols we have
  $N \leq d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

- How many symbols do we need to represent a value N?

- Or, what is the largest N we can represent with ℓ symbols?

- With ℓ symbols we have
  $N \leq d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

$$N \leq (b-1)b^{\ell-1} + (b-1)b^{\ell-2} + \cdots + (b-1)b^1 + b - 1$$

Space Complexity

- How many symbols do we need to represent a value N?

- Or, what is the largest N we can represent with ℓ symbols?

- With ℓ symbols we have
  $N \leq d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

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

- How many symbols do we need to represent a value N?

- Or, what is the largest N we can represent with ℓ symbols?

- With ℓ symbols we have
  $N \leq d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

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

- With ℓ symbols, the highest value is $N = b^\ell - 1$

Space Complexity

- How many symbols do we need to represent a value N?

- Or, what is the largest N we can represent with ℓ symbols?

- With ℓ symbols we have
  $N \le d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \le d_i \le b-1)$

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

- With ℓ symbols, the highest value is $N = b^\ell - 1$

- So, $\ell = \lceil \log_b (N+1) \rceil$

Space Complexity

- How many symbols do we need to represent a value N?

- Or, what is the largest N we can represent with ℓ symbols?

- With ℓ symbols we have
  $N \leq d_{\ell-1}b^{\ell-1} + d_{\ell-2}b^{\ell-2} + \cdots + d_1 b^1 + d_0 b^0 \quad (0 \leq d_i \leq b-1)$

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

- With ℓ symbols, the highest value is $N = b^\ell - 1$

- So, $\ell = \lceil \log_b (N+1) \rceil \quad \Rightarrow \quad \boxed{\boxed{\ell = \Theta(\log N)}}$

Space Complexity (2)

- The ***space complexity*** for N is $\ell = \Theta(\log N)$

Space Complexity (2)

- The ***space complexity*** for N is $\ell = \Theta(\log N)$
- Can we do better?

- The ***space complexity*** for N is $\ell = \Theta(\log N)$
- Can we do better?
- No!

Space Complexity (2)

- The ***space complexity*** for N is $\ell = \Theta(\log N)$
- Can we do better?
- No!
  - there are exactly b^ℓ combinations of ℓ symbols chosen from an alphabet Σ with $|\Sigma| = b$
  - i.e., you can not *express* more than b^ℓ values with ℓ symbols

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

Adding Numbers

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively
- How do we *add* two numbers represented in base-b notation?

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

- How do we *add* two numbers represented in base-b notation?

- **Theorem:** the sum of three base-b digits (whose values are $x, y, z \leq b - 1$) can be represented with *two* base-b digits

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

- How do we *add* two numbers represented in base-b notation?

- **Theorem:** the sum of three base-b digits (whose values are $x, y, z \leq b - 1$) can be represented with *two* base-b digits

  Proof:

  - *case* $b = 2$*:* $\ell = 2$ since $1 + 1 + 1 = 3_{\text{ten}} = 11_{\text{two}}$

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

- How do we *add* two numbers represented in base-b notation?

- **Theorem:** the sum of three base-b digits (whose values are $x, y, z \leq b - 1$) can be represented with *two* base-b digits

  Proof:

  - *case $b = 2$:* $\ell = 2$ since $1 + 1 + 1 = 3_{\text{ten}} = 11_{\text{two}}$
  - *case $b \geq 3$:*

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

- How do we *add* two numbers represented in base-b notation?

- **Theorem:** the sum of three base-b digits (whose values are $x, y, z \le b - 1$) can be represented with *two* base-b digits

  Proof:

  - *case* $b = 2$*:* $\ell = 2$ since $1 + 1 + 1 = 3_{\text{ten}} = 11_{\text{two}}$
  - *case* $b \ge 3$*:*

    1. we can represent $x + y + z$ in $\ell = \lceil \log_b (x + y + z + 1) \rceil$

Adding Numbers

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

- How do we *add* two numbers represented in base-b notation?

- **Theorem:** the sum of three base-b digits (whose values are $x, y, z \leq b - 1$) can be represented with *two* base-b digits

  Proof:

  - *case* $b = 2$*:* $\ell = 2$ since $1 + 1 + 1 = 3_{\text{ten}} = 11_{\text{two}}$
  - *case* $b \geq 3$*:*

    1. we can represent $x + y + z$ in $\ell = \lceil \log_b (x + y + z + 1) \rceil$
    2. $x + y + z + 1 \leq 3b$, because x, y, z are three base-b digits, therefore $\ell = \lceil \log_b (x + y + z + 1) \rceil \leq \log_b 3b = \log_b 3 + 1$

- We can now represent n_1 and n_2 with $\Theta(\log n_1)$ and $\Theta(\log n_2)$ bits, respectively

- How do we *add* two numbers represented in base-b notation?

- **Theorem:** the sum of three base-b digits (whose values are $x, y, z \leq b - 1$) can be represented with *two* base-b digits

  Proof:

  - *case* $b = 2$*:* $\ell = 2$ since $1 + 1 + 1 = 3_{\text{ten}} = 11_{\text{two}}$
  - *case* $b \geq 3$*:*

    1. we can represent $x + y + z$ in $\ell = \lceil \log_b (x + y + z + 1) \rceil$
    2. $x + y + z + 1 \leq 3b$, because x, y, z are three base-b digits, therefore $\ell = \lceil \log_b (x + y + z + 1) \rceil \leq \log_b 3b = \log_b 3 + 1$
    3. $b \geq 3$, therefore $\log_b 3 \leq 1$, therefore $\ell \leq 2$

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

Adding Numbers (2)

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

Adding Numbers (2)

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

- We know that three base-*b* digits add up to a number of up to *two* base-*b* digits, so this is our building block

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

- So, given x and y

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

Adding Numbers (2)

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

Adding Numbers (2)

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

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

Adding Numbers (2)

- We know that three base-b digits add up to a number of up to *two* base-b digits, so this is our building block

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

- So, given x and y

$$\begin{array}{rcccccc}
 & & & & c_1 & c_0 & 0 \\
x = & & x_{\ell-1} & \ldots & & x_1 & x_0 \\
y = & & y_{\ell-1} & \ldots & & y_1 & y_0 \\
x+y = & \multicolumn{6}{l}{\underbrace{c_{\ell-1} \quad s_{\ell-1} \quad \ldots \quad s_1 \quad s_0}_{\ell+1}}
\end{array}$$

Adding Numbers (3)

- Given two *arrays* of ℓ base-b digits, A and B

Adding Numbers (3)

- Given two *arrays* of ℓ base-b digits, A and B

Add(A, B)

$R = 0$ // $[0, \ldots, 0]$ of size $\ell + 1$
$c = 0$
for $i = 1$ **to** ℓ
        $(c, R[i]) =$ **AddThreeDigits**$(A[i], B[i], c)$
$R[\ell + 1] = c$
return R

- Given two *arrays* of ℓ base-b digits, A and B

```
ADD(A, B)
R = 0 // [0, . . . , 0] of size ℓ + 1
c = 0
for i = 1 to ℓ
        (c, R[i]) = ADDTHREEDIGITS(A[i], B[i], c)
R[ℓ + 1] = c
return R
```

- Is it correct?

Adding Numbers (3)

- Given two *arrays* of ℓ base-b digits, A and B

ADD(A, B)

$R = 0$ // $[0, \ldots, 0]$ of size $\ell + 1$
$c = 0$
for $i = 1$ **to** ℓ
    $(c, R[i]) =$ **ADDTHREEDIGITS**$(A[i], B[i], c)$
$R[\ell + 1] = c$
return R

- Is it correct? Yes

Adding Numbers (3)

- Given two *arrays* of ℓ base-b digits, A and B

```
ADD(A, B)
R = 0 // [0, . . . , 0] of size ℓ + 1
c = 0
for i = 1 to ℓ
        (c, R[i]) = ADDTHREEDIGITS(A[i], B[i], c)
R[ℓ + 1] = c
return R
```

- Is it correct? Yes

- How long does it take?

- Can we do better?

Complexity

Complexity

- We are interested in $T(\ell)$ (remember that $\ell = \Theta(\log N)$)

Complexity

- We are interested in $T(\ell)$ (remember that $\ell = \Theta(\log N)$)

```
ADD(A, B)
R = 0 // [0, . . . , 0] of size ℓ + 1
c = 0
for i = 1 to ℓ
        (c, R[i]) = ADDTHREEDIGITS(A[i], B[i], c)
R[ℓ + 1] = c
return R
```

Complexity

- We are interested in $T(\ell)$ (remember that $\ell = \Theta(\log N)$)

```
ADD(A, B)
R = 0 // [0, . . . , 0] of size ℓ + 1
c = 0
for i = 1 to ℓ
        (c, R[i]) = ADDTHREEDIGITS(A[i], B[i], c)
R[ℓ + 1] = c
return R
```

$$\boxed{T(\ell) = \Theta(\ell)}$$

- We are interested in $T(\ell)$ (remember that $\ell = \Theta(\log N)$)

```
ADD(A, B)
  R = 0 // [0, ..., 0] of size ℓ + 1
  c = 0
  for i = 1 to ℓ
        (c, R[i]) = ADDTHREEDIGITS(A[i], B[i], c)
  R[ℓ + 1] = c
  return R
```

$$\boxed{T(\ell) = \Theta(\ell)}$$

- Can we do better?

- We are interested in $T(\ell)$ (remember that $\ell = \Theta(\log N)$)

```
ADD(A, B)
R = 0 // [0, ..., 0] of size ℓ + 1
c = 0
for i = 1 to ℓ
        (c, R[i]) = ADDTHREEDIGITS(A[i], B[i], c)
R[ℓ + 1] = c
return R
```

$$\boxed{T(\ell) = \Theta(\ell)}$$

- Can we do better? No!
  - we have to at least look at the ℓ symbols from the input values
  - we must assign at least $\ell + 1$ symbols for the result

Multiplying Numbers

- We can now add two numbers

- Now, how do we *multiply* two numbers?

Multiplying Numbers

- We can now add two numbers
- Now, how do we *multiply* two numbers?
- Remember that our representation is a polynomial

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

- We can now add two numbers
- Now, how do we *multiply* two numbers?
- Remember that our representation is a polynomial

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

multiplying x by a simple polynomial in b, say $y = y_i b^i$, we obtain

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

- We can now add two numbers
- Now, how do we *multiply* two numbers?
- Remember that our representation is a polynomial

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

  multiplying x by a simple polynomial in b, say $y = y_i b^i$, we obtain

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

- Muliplying by b^i is equivalent to *shifting* our representation to the left by i positions
  - *left* means in the direction of the most significant bits

Multiplying Binary Numbers

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

  where x_i is either 0 or 1

Multiplying Binary Numbers

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

  where x_i is either 0 or 1

- For example, let $x = 1001_{\text{two}}$ and $y = 1011_{\text{two}}$

Multiplying Binary Numbers

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

  where x_i is either 0 or 1

- For example, let $x = 1001_{\text{two}}$ and $y = 1011_{\text{two}}$

  $x \times y =$

$$\begin{array}{cccccc} 1 & 0 & 0 & 1 & & (1001 \times 1) \end{array}$$

Multiplying Binary Numbers

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

where x_i is either 0 or 1

- For example, let $x = 1001_{\text{two}}$ and $y = 1011_{\text{two}}$

$x \times y =$

$$\begin{array}{rccccccl}
 & & 1 & 0 & 0 & 1 & & (1001 \times 1) \\
+ & 1 & 0 & 0 & 1 & & & (1001 \times 1 \text{ shifted by } 1)
\end{array}$$

Multiplying Binary Numbers

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

  where x_i is either 0 or 1

- For example, let $x = 1001_{\text{two}}$ and $y = 1011_{\text{two}}$

  $x \times y =$

$$\begin{array}{cccccccl} & & & 1 & 0 & 0 & 1 & (1001 \times 1) \\ + & & 1 & 0 & 0 & 1 & & (1001 \times 1 \text{ shifted by } 1) \\ + & 0 & 0 & 0 & 0 & & & (1001 \times 0 \text{ shifted by } 2) \end{array}$$

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

where x_i is either 0 or 1

- For example, let $x = 1001_{\text{two}}$ and $y = 1011_{\text{two}}$

$x \times y =$

				1	0	0	1	(1001×1)
+			1	0	0	1		(1001×1 shifted by 1)
+		0	0	0	0			(1001×0 shifted by 2)
+	1	0	0	1				(1001×1 shifted by 3)

Multiplying Binary Numbers

- Let's now focus on binary numbers (i.e., base $b = 2$)

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

where x_i is either 0 or 1

- For example, let $x = 1001_{\text{two}}$ and $y = 1011_{\text{two}}$

$x \times y =$

$$\begin{array}{ccccccccccl}
 & & & & & 1 & 0 & 0 & 1 & & (1001 \times 1) \\
+ & & & & 1 & 0 & 0 & 1 & & & (1001 \times 1 \text{ shifted by } 1) \\
+ & & & 0 & 0 & 0 & 0 & & & & (1001 \times 0 \text{ shifted by } 2) \\
+ & & 1 & 0 & 0 & 1 & & & & & (1001 \times 1 \text{ shifted by } 3) \\
\hline
= & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & & (1001 \times 1011)
\end{array}$$

Multiplying Numbers (2)

- Given two *arrays* of ℓ binary digits, A and B

Multiplying Numbers (2)

- Given two *arrays* of ℓ binary digits, A and B

```
MULTIPLY(A, B)
R = 0
T = A
for i = 1 to ℓ
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

Multiplying Numbers (2)

- Given two *arrays* of ℓ binary digits, A and B

```
MULTIPLY(A, B)
R = 0
T = A
for i = 1 to ℓ
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

- Is it correct?

Multiplying Numbers (2)

- Given two *arrays* of ℓ binary digits, A and B

```
MULTIPLY(A, B)
R = 0
T = A
for i = 1 to ℓ
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

- Is it correct? Yes

Multiplying Numbers (2)

- Given two *arrays* of ℓ binary digits, A and B

```
MULTIPLY(A, B)
R = 0
T = A
for i = 1 to ℓ
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

- Is it correct? Yes

- How long does it take?

- Can we do better?

- Again we are interested in $T(\ell)$

Complexity

- Again we are interested in $T(\ell)$

```
MULTIPLY(A, B)
R = 0
T = A
for i = 0 to ℓ − 1
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

- Again we are interested in $T(\ell)$

```
MULTIPLY(A, B)
R = 0
T = A
for i = 0 to ℓ − 1
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

$$\boxed{T(\ell) = \Theta(\ell^2)}$$

Complexity

- Again we are interested in $T(\ell)$

```
MULTIPLY(A, B)
R = 0
T = A
for i = 0 to ℓ − 1
        if B[i] == 1
                R = ADD(R, T)
        T = SHIFTLEFT(T)
return R
```

$$\boxed{T(\ell) = \Theta(\ell^2)}$$

- Can we do better?

Complexity

- Again we are interested in $T(\ell)$

```
MULTIPLY(A, B)
R = 0
T = A
for i = 0 to ℓ − 1
    if B[i] == 1
        R = ADD(R, T)
    T = SHIFTLEFT(T)
return R
```

$$\boxed{T(\ell) = \Theta(\ell^2)}$$

- Can we do better? Yes!

A Better Way to Multiply

A Better Way to Multiply

- Intuition: let’s try to split numbers (their representations) in half

A Better Way to Multiply

- Intuition: let's try to split numbers (their representations) in half

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

A Better Way to Multiply

- Intuition: let's try to split numbers (their representations) in half

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

  which means $x = 2^{\ell/2}x_L + x_R$ and $y = 2^{\ell/2}y_L + y_R$, so…

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

  we reduced the problem of multiplying two numbers of ℓ bits into the problem of multiplying *four* numbers of $\ell/2$ bits…

A Better Way to Multiply

- Intuition: let's try to split numbers (their representations) in half

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

  which means $x = 2^{\ell/2}x_L + x_R$ and $y = 2^{\ell/2}y_L + y_R$, so…

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

  we reduced the problem of multiplying two numbers of ℓ bits into the problem of multiplying *four* numbers of $\ell/2$ bits…

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

- Intuition: let's try to split numbers (their representations) in half

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

which means $x = 2^{\ell/2}x_L + x_R$ and $y = 2^{\ell/2}y_L + y_R$, so...

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

we reduced the problem of multiplying two numbers of ℓ bits into the problem of multiplying *four* numbers of $\ell/2$ bits...

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

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

Almost There

- Again, we have

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

- Again, we have

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

but notice that $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, so

- Again, we have

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

but notice that $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, so

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

- Again, we have

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

but notice that $x_L y_R + x_R y_L = (x_L + x_R)(y_R + y_L) - x_L y_L - x_R y_R$, so

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

Only 3 multiplications: $x_L y_L$, $(x_L + x_R)(y_R + y_L)$, and $x_R y_R$

Almost There

- Again, we have

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

but notice that $x_Ly_R + x_Ry_L = (x_L + x_R)(y_R + y_L) - x_Ly_L - x_Ry_R$, so

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

Only 3 multiplications: x_Ly_L, $(x_L + x_R)(y_R + y_L)$, and x_Ry_R

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

Almost There

- Again, we have

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

but notice that $x_L y_R + x_R y_L = (x_L + x_R)(y_R + y_L) - x_L y_L - x_R y_R$, so

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

Only 3 multiplications: $x_L y_L$, $(x_L + x_R)(y_R + y_L)$, and $x_R y_R$

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

which, as we will see, leads to a much better complexity

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