CIT 596: ALGORITHMS & COMPUTATION

Basic Algorithms for Arithmetic

Arithmetic on Big Numbers

- When we perform an arithmetic operation on numbers of bounded size or precision—like the Java primitives `int` or `float`—we count that as a single computational step.
- Some applications, like cryptography and scientific computing, use larger or more precise numbers, where a single arithmetic operation may take much longer.
- In this context, we will analyze running time by counting the number of arithmetic operations on ***individual digits***.
- Computers do this in binary, but we'll use base ten.
- We focus on positive integers, given as arrays of digits, indexed in reverse order.

Big Numbers as Arrays

An n-digit number x can be represented as an array

$$x = [x_{n-1}, x_{n-2}, \dots, x_1, x_0],$$

or just $x = x_{n-1} \dots x_0$ for short, where

$$x = \sum_{i=0}^{n-1} x_i \cdot 10^i \,.$$

For example,

$$93751 = 9 \cdot 10^4 + 3 \cdot 10^3 + 7 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0.$$

We will always have $n = \Theta(\log x)$.

Grade-School Addition

Input: n-digit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.

Output: $(n+1)$-digit number $z = x + y$.


```
ADD(x, y)
    c_0 = 0
    for i = 0 to n − 1
        s = c_i + x_i + y_i
        z_i = s % 10
        c_{i+1} = (s − z_i)/10
    z_n = c_n
    return z_n … z_0
```

i	4	3	2	1	0
c_i	1	0	1	1	0
x_i		5	3	3	4
y_i		7	5	9	6
z_i	1	2	9	3	0

Correctness of ADD

Loop invariant: After iteration i of the **for** loop,

$$c_{i+1}z_i \ldots z_0 = x_i \ldots x_0 + y_i \ldots y_0.$$

Initialization: In the $i = 0$ iteration, we set $c_1 z_0 = s = x_0 + y_0$.

Maintenance: Fix $1 \le i < n$, and suppose

$$c_i z_{i-1} \ldots z_0 = x_{i-1} \ldots x_0 + y_{i-1} \ldots y_0$$

after iteration $i - 1$. Then we set $c_{i+1}z_i = s = c_i + x_i + y_i$ in iteration i, so

$$(c_{i+1}z_i - c_i) \cdot 10^i = (x_i + y_i) \cdot 10^i.$$

Adding these two equations together gives $c_{i+1}z_i \ldots z_0 = x_i \ldots x_0 + y_i \ldots y_0$.

Termination: If $c_n z_{n-1} \ldots z_0 = x_{n-1} \ldots x_0 + y_{n-1} \ldots y_0$ after iteration $n - 1$, then setting $z_n = c_n$ gives the correct output.

Running Time of ADD

ADD(x, y)
    $c_0 = 0$
    for $i = 0$ **to** $n - 1$
        $s = c_i + x_i + y_i$
        $z_i = s \,\%\, 10$
        $c_{i+1} = (s - z_i)/10$
    $z_n = c_n$
    return $z_n \ldots z_0$

$\Theta(1)$

$n \cdot \Theta(1) = \Theta(n)$ time

Could this be faster? Not asymptotically; it takes $\Theta(n)$ time just to read the inputs.

Grade-School Multiplication

Input: n-digit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.

Output: $2n$-digit number $z = x \cdot y$.

i	3	2	1	0
x_i		5	9	6
y_i		5	9	6
$t_{0,i}$	3	5	7	6
$t_{1,i}$	5	3	6	4
$t_{2,i}$	2	9	8	0

$\textsc{Mult}(x, y)$
- $z = 0$
- **for** $j = 0$ **to** $n - 1$
  - $c = 0$
  - **for** $i = 0$ **to** $n - 1$
    - $p = c + (x_i \cdot y_j)$
    - $t_{j,i} = p \,\%\, 10$
    - $c = \left(p - t_{j,i}\right)/10$
  - $t_{j,n} = c$
  - $z = z + (t_{j,n} \dots t_{j,0}) \cdot 10^j$
- **return** z

Analysis of MULT

MULT(x, y)
- $z = 0$
- **for** $j = 0$ **to** $n - 1$
  - $c = 0$
  - **for** $i = 0$ **to** $n - 1$
    - $p = c + (x_i \cdot y_j)$
    - $t_{j,i} = p \,\%\, 10$
    - $c = (p - t_{j,i})/10$

    $O(n)$
  - $t_{j,n} = c$
  - $z = z + (t_{j,n} \ldots t_{j,0}) \cdot 10^j$ $O(n)$
- **return** z

Loop invariant: After iteration j of the outer **for** loop,

$$z = x \cdot (y_j \ldots y_0).$$

$n \cdot O(n) =$

Running time: $O(n^2)$

Could this be faster? Yes!