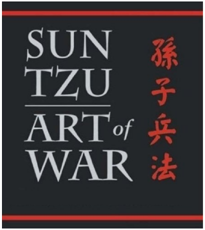

THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Algorithms and Data Structures

Lecture 6 Karatsuba’s Algorithm

Jiamou Liu
The University of Auckland



故用兵之法，十則圍之，五則攻之，倍則分之， 敵則能戰之，少則能守之，不若則能避之。

``It is the rule in war, if ten times the enemy's strength, surround them; if five times, attack them; if double, be able to divide them; if equal,engage them; if fewer, be able to evade them; if weaker, be able to avoid them.''

---``Chapter III Strategic Attack" 500BC

The Multiplication Problem

- Multiplication of integers is considered an elementary operation.
- However, this is constrained by the number of bits that can be processed at once by the CPU.
- For 64-bit system, the largest integer is normally bounded by 2^{63}.

The Multiplication Problem

- Multiplication of integers is considered an elementary operation.
- However, this is constrained by the number of bits that can be processed at once by the CPU.
- For 64-bit system, the largest integer is normally bounded by 2^{63}.

There are cases when large integer multiplication is needed:

- Cryptography
- Financial calculation
- Scientific computing
- Bio-informatics
- Random number generation
- Blockchain: Ethereum smart contract integers occupy 256 bits.

- For large integers, multiplication is no longer an elementary operation, and the time complexity is no longer $O(1)$.
- It is thus important to investigate the algorithm for performing multiplication of two integers.

Multiplication problem

- **INPUT:** Two arrays $x[0..n-1]$, $y[0..n-1]$ representing binary numbers of equal number of bits
- **OUTPUT:** Array $z[0..2n-1]$ that represents the product $x \times y$

- For large integers, multiplication is no longer an elementary operation, and the time complexity is no longer $O(1)$.
- It is thus important to investigate the algorithm for performing multiplication of two integers.

Multiplication problem

- **INPUT:** Two arrays $x[0..n-1]$, $y[0..n-1]$ representing binary numbers of equal number of bits
- **OUTPUT:** Array $z[0..2n-1]$ that represents the product $x \times y$

Remark. 1. We assume that x, y represent integers reading backwards, e.g.,

$$x = [0, 0, 0, 1] \text{ represents } (1000)_2 = 8$$

2. For x, y with different lengths, pad the shorter one with 0, e.g.,

$$x = [0, 0, 0, 1] \,\&\, y = [1, 0, 1, 0] \text{ represents } (1000)_2 \times (101)_2 = (40)_{10} = (101000)_2$$

$$\text{Output: } z = [0, 0, 0, 1, 0, 1]$$

Long Multiplication: The "Grade School" Algorithm

Described by Al-Khwārizmī in "Al-Kitab al-Mukhtasar fi Hisab al-Jabr wal-Muqabala" (The Compendious Book on Calculation by Completion and Balancing)

$$
\begin{array}{rcccccccccccccc}
 & & & & & & & & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\times & & & & & & & & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
 & & & & & & & & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
 & & & & & 1 & 1 & 0 & 0 & 1 & 0 & 1 & & & \\
 & & & & 1 & 1 & 0 & 0 & 1 & 0 & 1 & & & & \\
 & & 1 & 1 & 0 & 0 & 1 & 0 & 1 & & & & & & \\
\hline
 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1
\end{array}
$$

Question. How to implement this algorithm using a programming language?

To perform LongMultiplication($x[0..n-1], y[0..n-1]$):

1. **Multiplication:**
   1. Starting with $y[0]$, multiply it by each digit of x.
   2. Write each partial product, shifted one position to the left for each successive digit of the bottom number.
2. **Addition:**
   1. Sum up all the partial products, aligning them vertically.
   2. Carry over any digits when necessary.
3. **Result:** The result is the combined sum of the partial products.

function LongMultiplication(arrays $x[0..n-1]$, $y[0..n-1]$)

    INPUT: arrays x, y where each element is 0/1-valued

    Create a new 0-valued array $z[0..2n-1]$

    for $i \leftarrow 0$ to $n-1$ **do** ▷ Outer loop: Iterate through y

        $carry \leftarrow 0$

        if $y[i] = 1$ **then**

            for $j \leftarrow 0$ to $n-1$ **do** ▷ Inner loop: Iterate through x

                $product \leftarrow x[j] + z[i+j] + carry$

                $z[i+j] \leftarrow product \% 10_2$

                $carry \leftarrow product / 10_2$

            $z[i+n] \leftarrow carry$

    return z

```
                6 5 4 3 2 1 0

                1 1 0 0 1 0 1    x
      ×         1 0 1 1 0 0 1    y
-------------------------------
                1 1 0 0 1 0 1
          1 1 0 0 1 0 1
        1 1 0 0 1 0 1
    1 1 0 0 1 0 1
-------------------------------
1 0 0 0 1 1 0 0 0 1 1 1 0 1
```

Time complexity of long multiplication

- Outer for-loop: repeats n times
- Inner for-loop: repeats n times in worst case
- Each iteration in the inner loop: Constant number of (single-bit) addition, modulo, division, and assignment

Therefore the running time of long multiplication is $\Theta(n^2)$.

A Faster Multiplication?

Kolmogorov's Conjecture

A.Kolmogrov (1952):
`` Is there a more efficient algorithm?"

A Faster Multiplication?

Kolmogorov's Conjecture

A.Kolmogrov (1960):

Dean at Moscow State Univesity

`` Prove that there is no more efficient algorithm for integer multiplication "

A. Karatsuba :

23 year old grad student

`` There is a more efficient algorithm —— divide and conquer "

Observation

- $(2x+3)(5x+7) = 2\times 5x^2 + (2\times 7 + 3\times 5)x + 3\times 7$

  4 multiplications, 3 additions

Observation

- $(2x+3)(5x+7) = 2 \times 5x^2 + (2 \times 7 + 3 \times 5)x + 3 \times 7$

  4 multiplications, 3 additions

- $(2+3)(5+7) = 2 \times 5 + 3 \times 7 + 2 \times 7 + 3 \times 5$
  $\Rightarrow 2 \times 7 + 3 \times 5 = (2+3) \times (5+7) - 2 \times 5 - 3 \times 7$
  $\Rightarrow (2x+3)(5x+7) =$
  $2 \times 5x^2 + 3 \times 7 + ((2+3) \times (5+7) - 2 \times 5 - 3 \times 7)x$

  3 multiplications, 6 additions

Observation

- $(2x+3)(5x+7) = 2\times 5x^2 + (2\times 7 + 3\times 5)x + 3\times 7$

  4 multiplications, 3 additions

- $(2+3)(5+7) = 2\times 5 + 3\times 7 + 2\times 7 + 3\times 5$
  $\Rightarrow 2\times 7 + 3\times 5 = (2+3)\times(5+7) - 2\times 5 - 3\times 7$
  $\Rightarrow (2x+3)(5x+7) =$
  $2\times 5x^2 + 3\times 7 + ((2+3)\times(5+7) - 2\times 5 - 3\times 7)x$

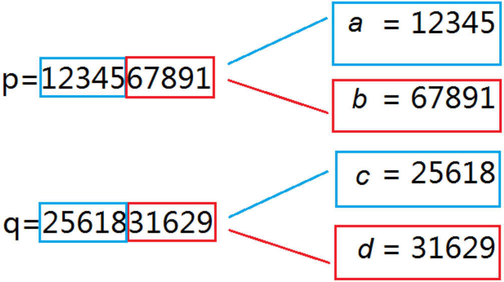
  3 multiplications, 6 additions

General Version

$$(ax+b)(cx+d) = acx^2 + bd + ((a+b)(c+d) - ac - bd)x$$

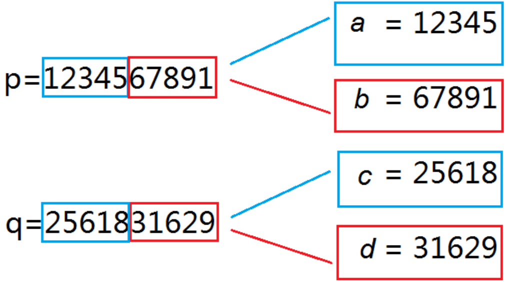
Karatsuba's Algorithm

Example:



Karatsuba's Algorithm

Example:



$$
\begin{aligned}
p \times q &= (a \times 10^5 + b) \times (c \times 10^5 + d) \\
&= ac \times 10^{10} + bd + ((a+b)(c+d) - ac - bd) \times 10^5
\end{aligned}
$$

Function karatsuba($x[0..n-1], y[0..n-1]$)

- **Input**: Two integers x, y, represented as arrays of bits.
- **Base Case**: If the integers have only one digit, multiply them directly.
- **Recursive Step**:
  - Split each integer into two halves: a ″high″ part and a ″low″ part.
  - Recursively compute:
    - z_0: the product of the ″low″ parts.
    - z_2: the product of the ″high″ parts.
    - z_1: the product of the sum of the ″low″ and ″high″ parts of both numbers.
  - Combine the results using:
    - $z_2 \ll 2k$ to shift z_2 by two halves (to the left).
    - $(z_1 - z_2 - z_0) \ll k$ to shift and adjust z_1.
    - Add z_0 directly.
- **Output**: The combined result gives the final product of the two large integers.

function KaratsubaMultiplication(arrays $x[0..n-1]$, $y[0..n-1]$)
    if $n = 1$ **then**
        return $x[0] \times y[0]$
    $k \leftarrow \lfloor n/2 \rfloor$
    Divide x into two parts: $x_{high} \leftarrow x[k..n-1]$, $x_{low} \leftarrow x[0..k-1]$
    Divide y into two parts: $y_{high} \leftarrow y[k..n-1]$, $y_{low} \leftarrow y[0..k-1]$
    $z_0 \leftarrow$ **KaratsubaMultiplication**(x_{low}, y_{low})
    $z_1 \leftarrow$ **KaratsubaMultiplication**$(x_{low} + x_{high}, y_{low} + y_{high})$
    $z_2 \leftarrow$ **KaratsubaMultiplication**(x_{high}, y_{high})
    return $(z_2 \ll 2k) + ((z_1 - z_2 - z_0) \ll k) + z_0$

Time Complexity of Karatsuba's Algorithm

Question. What is the time complexity of Karatsuba's algorithm?

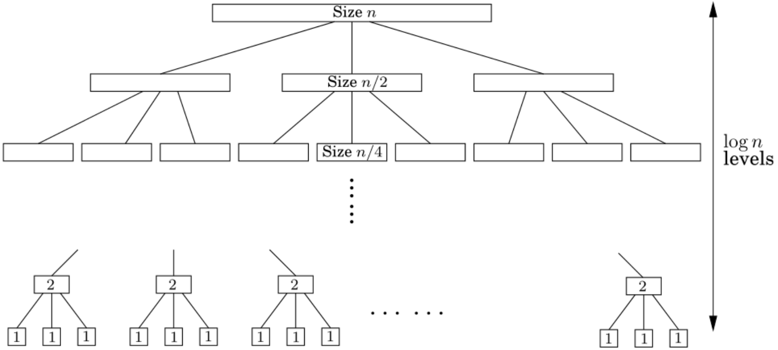
Time Complexity

Let $T(n)$ be the time complexity of multiplying two length-n integers.

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 3T(\lceil n/2 \rceil) + cn & \text{otherwise} \end{cases}$$

where c is a constant.

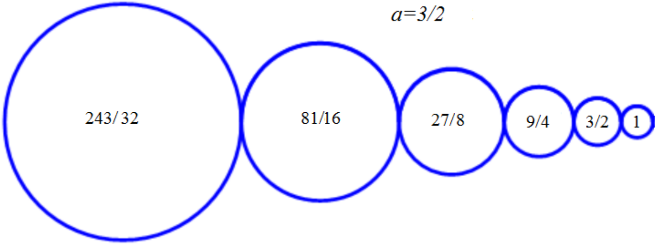
Tree of Recursive Calls



Time Complexity

$$T(n) = nc\left(1 + \frac{3}{2} + \frac{3^2}{2^2} + \ldots + \frac{3^{k-1}}{2^{k-1}} + \frac{3^k}{2^k}\right)$$

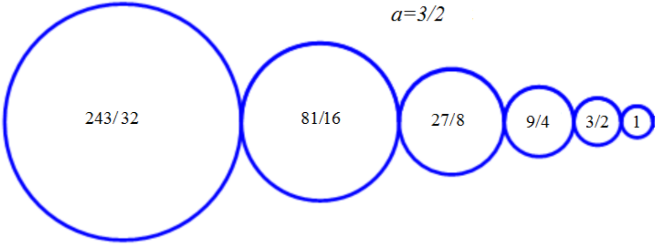
Geometric Series



Time Complexity

$$T(n) = nc\left(1 + \frac{3}{2} + \frac{3^2}{2^2} + \ldots + \frac{3^{k-1}}{2^{k-1}} + \frac{3^k}{2^k}\right)$$

Geometric Series



Therefore

$$T(n) = nc\frac{(3/2)^{k+1} - 1}{3/2 - 1} \leq 3nc\frac{3^k}{2^k}$$

Solving Recurrence

Time Complexity

Recall $n = 2^k$. Therefore $k = \log_2 n$. We have

$$T(n) \leq 3nc\frac{3^{\log_2 n}}{2^{\log_2 n}} = 3nc\frac{3^{\log_2 n}}{n} = 3c \times 3^{\log_2 n}$$

Solving Recurrence

Time Complexity

Recall $n = 2^k$. Therefore $k = \log_2 n$. We have

$$T(n) \leq 3nc\frac{3^{\log_2 n}}{2^{\log_2 n}} = 3nc\frac{3^{\log_2 n}}{n} = 3c \times 3^{\log_2 n}$$

Note that $3 = 2^{\log_2 3}$. So

$$\begin{aligned} T(n) &\leq 3c \times (2^{\log_2 3})^{\log_2 n} \\ &= 3c \times 2^{\log_2 3 \times \log_2 n} \\ &= 3c \times 2^{\log_2 n \times \log_2 3} \\ &= 3c \times (2^{\log_2 n})^{\log_2 3} \\ &= 3c \times n^{\log_2 3} \\ &\leq 3c \times n^{1.59} \end{aligned}$$

Therefore $T(n)$ is $O(n^{1.59})$, a big improvement from $O(n^2)$!!

Summary

SUMMARY

In this lecture, we start to explore an important algorithm design strategy: **divide and conquer**.
We looked at Karatsuba's algorithm as a first example of a divide and conquer algorithm.

- Multiplication problem: Finding the product of two (large) integers
- **Classical approach:** Long multiplication method dated back to ancient time.

  Running time $O(n^2)$
- **Karatsuba's algorithm:**
  - Dividing integer arrays into halves
  - Reduce the multiplication problem into three sub-problems.
  - Recursively solve the sub-problems
  - Add the results to produce a result of the original problem

  Running time $O(n^{1.59})$

This is the first time in 5000 years when human multiply numbers asymptotically faster than quadratic time.


THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND