Exponentiation

- Given a and n, compute a^n.
- $a \times a \times \cdots \times a$ $\quad n$ times
- n-1 multiplications
- $a^{16} = (((a^2)^2)^2)^2$ only 4 multiplications
- $a^{11} = ((a^2)^2)^2 \times a^2 \times a$ only 5 multiplications
- Repeated squaring technique

Repeated Squaring

- Exp(a, n) :
  - If n is even
    - return (Exp(a, $n/2$))^2
  - If n is odd
    - return (Exp(a, $(n-1)/2$))$^2 \times a$
  - If n is 1
    - return a

Number of multiplications

$$T(n) \leq T(n/2) + 2$$

$$T(n) \leq 2 \log n$$

Another implementation

- Exp(a, n) :
  - If n is even
    - return (Exp(a^2, $n/2$))
  - If n is odd
    - return (Exp(a^2, $(n-1)/2$)) $\times$ a
  - If n is 1
    - return a

Iterative implementation

- Input: a, n
- Initialize

  $a\_power\_n \leftarrow 1;$ // this will be a^n at the end

  $a\_two\_power \leftarrow a;$ // this will be $a^{2^\wedge i}$ after i iterations

- while ($n > 0$)
  - If (n is odd) then $a\_power\_n \leftarrow a\_two\_power \times a\_power\_n;$
  - $a\_two\_power \leftarrow a\_two\_power \times a\_two\_power;$
  - $n \leftarrow n/2$ //integer part after division by 2 //or right-shift by 1 bit

Repeated squaring

- Does it give the minimum number of multiplications?
- What about a^{15} ?
  - can be done in 5 multiplications
- Given n, what the minimum number of multiplications required for a^n ? No easy answer.
- Can apply repeated squaring for other operations like Matrix powering.
- Apparently proposed by Pingala (200 BC ?).

Fibonacci numbers

- $F(n) = F(n\text{-}1) + F(n\text{-}2)$
- Used by Pingala to count the number of patterns of short and long vowels.
- Here again we are repeating the same operation n times.
- Can repeated squaring be used?
- Can we compute $F(n)$ in $O(\log n)$ arithmetic operations?

Fibonacci numbers

- $F(n) = F(n\text{-}1) + F(n\text{-}2)$
- if n is even
  - $F(n) = F(n/2)\,F(n/2\text{-}1) + F(n/2+1)F(n/2)$
    $= F(n/2)\,(2F(n/2+1) - F(n/2))$
  - $F(n+1) = F(n/2)\,F(n/2) + F(n/2+1)F(n/2+1)$
- similarly, if n is odd

Fibonacci numbers

$$\begin{pmatrix} F_3 \\ F_2 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

$$\begin{pmatrix} F_5 \\ F_4 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_3 \\ F_2 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}^2 \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}^{n/2} \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

Integer Multiplication

- Addition: Adding two n bit numbers
  School method $O(n)$ time
- $O(n)$ time is necessary to write the output.
- Multiplication: multiplying two n bit numbers
  School method $O(n^2)$ time
- is $O(n^2)$ time is necessary?

Integer Multiplication

- Kolmogorov (1960) conjectured that $O(n^2)$ time is necessary.
- In a week, Karatsuba found $O(n^{1.59})$ time algorithm.
- Karatsuba quoted in Jeff Erickson's Algorithms
- "After the seminar I told Kolmogorov about the new algorithm and about the disproof of the n^2 conjecture. Kolmogorov was very agitated because this contradicted his very plausible conjecture. At the next meeting of the seminar, Kolmogorov himself told the participants about my method, and at that point the seminar was terminated."

Special case: Integer Squaring

- If we have a squaring algorithm, does it directly give us a multiplication algorithm?
- Input: n bit integer a
- Break it into two chunks $n\text{-}1$ bits and 1 bit
- $a = \varepsilon + 2\,b$
- $a^2 = \varepsilon^2 + 2\,b\,\varepsilon + b^2$
- $T(n) = T(n\text{-}1) + O(n) = O(n^2)$

Squaring: divide and conquer

- Input: n bit integer a
- Break it into two chunks: $n/2$ bits and $n/2$ bits
- $a = b + c\ 2^{n/2}$.
- $a^2 = b^2\ + 2\ b\ c\ 2^{n/2} + c^2\ 2^n$
- Need to compute the multiplication bc recursively.
- How to compute bc with the square function?
- $2\ bc = (b+c)^2\ - b^2 - c^2$

Squaring: divide and conquer

- Input: n bit integer a
- Break it into two chunks: $n/2$ bits and $n/2$ bits
- $a^2 = b^2 + (b^2 + c^2 - (b-c)^2) 2^{n/2} + c^2 2^n$
- Squaring an n bit number reduced to
  - squaring three $n/2$ bit numbers
  - and few additions and left shifts $O(n)$
- $T(n) = 3 T(n/2) + O(n)$

Running time analysis

- $T(n) = 3\ T(n/2) + O(n)$
  - $T(n) \le c\ n + 3\ T(n/2)$ for some constant c
  - $T(n) \le c\ n + 3cn/2 + 3^2\ T(n/2^2)$
  - $T(n) \le c\ n + 3cn/2 + 3^2\ cn/2^2 + 3^3\ T(n/2^3)$
  - $T(n) \le c\ n + 3cn/2 + 3^2\ cn/2^2 + \cdots + 3^{k-1}\ cn/2^{k-1} + 3^k\ T(n/2^k)$
  - Assuming $n = 2^k$ and $T(1) = 1$
  - $T(n) \le c\ n\ (1 + 3/2 + 3^2/2^2 + \cdots + 3^{k-1}/2^{k-1}) + 3^k$
  - $T(n) \le 2\ c\ n\ (3^k/2^k - 1) + 3^k \le (2c+1)\ 3^k = O(3^k) = O(3^{\log n}) = O(n^{\log 3})$

Karatsuba's multiplication

- Input: n bit integers a *and* b
- Break integers into two chunks: $n/2$ bits and $n/2$ bits
- $a = a_0 + a_1 \, 2^{n/2}$.
- $b = b_0 + b_1 \, 2^{n/2}$.
- $ab \; = a_0 \, b_0 \; + (a_0 \, b_1 + a_1 \, b_0) \, 2^{n/2} + a_1 \, b_1 \, 2^n$
- Want to compute the three terms $a_0 \, b_0$, $(a_0 \, b_1 + a_1 \, b_0)$, $a_1 \, b_1$ with only three multiplications and a few additions
- $T(n) = 3 \; T(n/2) + O(n)$
- $T(n) = O(n^{\log 3}) = O(n^{1.585})$

Toom-Cook multiplication

- Break it into three chunks: $n/3$ bits each
- $a = a_0 + a_1\, 2^{n/3} + a_2\, 2^{2n/3}$
- $b = b_0 + b_1\, 2^{n/3} + b_2\, 2^{2n/3}$
- $ab = a_0 b_0 + (a_0 b_1 + a_1 b_0)\, 2^{n/3} + (a_0 b_2 + a_1 b_1 + a_2 b_0)\, 2^{2n/3}$

  $+ (a_1 b_2 + a_2 b_1)\, 2^{3n/3} + a_2 b_2\, 2^{4n/3}$
- In how many multiplications of n/3 bit numbers, can you find these five terms?

Toom-Cook multiplication

- $ab = a_0 b_0 + (a_0 b_1 + a_1 b_0)\, 2^{n/3} + (a_0 b_2 + a_1 b_1 + a_2 b_0)\, 2^{2n/3}$

  $+ (a_1 b_2 + a_2 b_1)\, 2^{3n/3} + a_2 b_2\, 2^{4n/3}$

- In how many multiplications of n/3 bit numbers, can you find these five terms?
- If six multiplications
  - $T(n) = 6\, T(n/3) + O(n) \Longrightarrow T(n) = O(n^{(\log 6)/(\log 3)}) = O(n^{1.63})$
- If five multiplications
  - $T(n) = 5\, T(n/3) + O(n) \Longrightarrow T(n) = O(n^{(\log 5)/(\log 3)}) = O(n^{1.46})$

Toom-Cook multiplication

- Homework: find these five terms using five multiplications and a few additions
  - $a_0 b_0$
  - $a_0 b_1 + a_1 b_0$
  - $a_0 b_2 + a_1 b_1 + a_2 b_0$
  - $a_1 b_2 + a_2 b_1$
  - $a_2 b_2$

Toom-Cook multiplication

- Homework: find these five terms using five square operations and a few additions
  - P^2
  - PQ
  - $2PR + Q^2$
  - QR
  - R^2

Integer multiplication history

- Karatsuba: break integers into two parts: $O(n^{1.585})$
- Toom-Cook: break integers into three parts: $O(n^{1.46})$
- What if break into more parts?
  - can get better and better time complexity by increasing the number of parts
  - for k parts, we will get $O(k^2 n^{\log(2k-1) / \log k})$
  - we can get $O(n^{1+\varepsilon})$ for any constant $\varepsilon > 0$

Integer multiplication history

- for k parts, we will get $O(k^2 n^{\log(2k-1) / \log k})$
- we can get $O(n^{1+\varepsilon})$ for any constant $\varepsilon > 0$
- Exercise: how many parts to break into for $O(n^{1.1})$
- Theoretically faster, but not necessarily faster in practice due to large constants.
- For multiplying 64 bit integers, Karatsuba may not be faster than school method. A combination of the two may be faster.

Integer multiplication history

- [1960] Karatsuba $O(n^{1.585})$
- [1963, 1966] Toom-Cook: $O(n^{1.46})$
- [1981] Donald Knuth: $O(n\ 2^{\sqrt{(2 \log n)}} \log n)$
- [1971] Schönhage–Strassen: $O(n \log n \log\log n)$
- [2007, 2008] Fürer, De-Kurur-Saha-Saptharishi: $O(n \log n\ 2^{\log^* n})$
- [2019] Harvey-van der Hoeven: $O(n \log n)$
- Conjecture: $O(n \log n)$ can not be improved

Matrix multiplication

- Input: $n \times n$ matrices A *and* B
- Break matrices into four parts: $n/2 \times n/2$ each

$$A = \begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix} \qquad B = \begin{pmatrix} B_0 & B_1 \\ B_2 & B_3 \end{pmatrix}$$

$$AB = \begin{pmatrix} A_0B_0 + A_1B_2 & A_0B_1 + A_1B_3 \\ A_2B_0 + A_3B_2 & A_2B_1 + A_3B_3 \end{pmatrix}$$

- Want to compute the four matrices with only seven multiplications and a few additions

Polynomial multiplication

- Input: degree d polynomials P *and* Q
- $P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d.$
- $Q(x) = q_0 + q_1 x + q_2 x^2 + \cdots + q_d x^d.$
- $P(x)\, Q(x) = p_0 + (p_0q_1 + p_1q_0)x + \cdots + p_dq_d\, x^{2d}.$
- Assume integer multiplication in unit time.
- School multiplication $O(d^2)$ time.
- Can we do better?