CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 1: Basics of algorithms

Divide, Delegate and Combine (Divide and Conquer)

You cannot do everything and be efficient!

Integer multiplication

Problem Description

Input:	Two n-digit non-negative integers x, y
Compute:	$x \times y$

We know that this has a simple algorithm (we studied in school).

What is the time complexity of that algorithm?

Primitive operations:

Adding two single digit numbers takes $O(1)$ time.

Multiplying two single digit numbers takes $O(1)$ time.

Inserting a zero at the end of a number takes $O(1)$ time.

Total number of primitive operations

$O(n)$ operations to multiply 1 digit of y with x.

$O(n)$ such operations. Totally $O(n^2)$ operations.

Integer Multiplication

Can we do better than $O(n^2)$?

Recursive Algorithm

`Mult`$((u, v))$

- Let a, b the first and the second half of u respectively.
- Let c, d the first and the second half of v respectively.
- Output
  $\mathbf{10^n} \cdot \texttt{Mult}(\mathbf{a}, \mathbf{c}) + \mathbf{10^{n/2}} \cdot (\texttt{Mult}(\mathbf{a}, \mathbf{d}) + \texttt{Mult}(\mathbf{b}, \mathbf{c})) + \texttt{Mult}(\mathbf{b}, \mathbf{d})$.

Recursive Algorithm

$\texttt{Mult}((u, v))$

- Let a, b the first and the second half of u respectively.
- Let c, d the first and the second half of v respectively.
- Output
  $\mathbf{10^n} \cdot \texttt{Mult}(\mathbf{a}, \mathbf{c}) + \mathbf{10^{n/2}} \cdot (\texttt{Mult}(\mathbf{a}, \mathbf{d}) + \texttt{Mult}(\mathbf{b}, \mathbf{c})) + \texttt{Mult}(\mathbf{b}, \mathbf{d})$.

Running time analysis of the algorithm.

$$\begin{aligned}
T(n) &= 4 \cdot T(n/2) + O(n) \\
&= 4 \cdot (4 \cdot T(n/4) + O(n/2)) + O(n) \\
&= \vdots \\
&= O(n^2).
\end{aligned}$$

Analysing Recursive Algorithms

Master Theorem

Theorem

Let

$$T(n) = a \cdot T(n/b) + \Theta(n^c),$$

where $a, b, c \in \mathbb{N}$, $a \geq 1, b > 1$ *and* $c \geq 0$. *Then*

- $T(n) = \Theta(n^c)$ *if* $a < b^c$,
- $T(n) = \Theta(n^c \log n)$ *if* $a = b^c$, *and*
- $T(n) = \Theta(n^{\log_b a})$ *if* $a \geq b^c$.

If $T(n) = 4 \cdot T(n/2) + O(n)$ then $a = 4, b = 2, c = 1$, i.e. $a \geq b^c$. Therefore, we get $T(n) = O(n^2)$.

Karatsuba's algorithm

`Mult((u, v))`

- Let a, b the first and the second half of u respectively.
- Let c, d the first and the second half of v respectively.
- Output
  $\mathbf{10^n} \cdot \texttt{Mult}(\mathbf{a}, \mathbf{c}) + \mathbf{10^{n/2}} \cdot (\texttt{Mult}(\mathbf{a}, \mathbf{d}) + \texttt{Mult}(\mathbf{b}, \mathbf{c})) + \texttt{Mult}(\mathbf{b}, \mathbf{d})$.

Can we do better?

Above algorithm makes 4 calls to $\texttt{Mult}(\cdot)$.

Can we make only 3 calls and get the same answer?

If we can do that then, $T(n) = 3 \cdot T(n/2) + O(n)$.

We will be able to get $T(n) = O(n^{\log 3}) = O(n^{1.584})$.

How do we save?

Karastuba's algorithm

How do we save?

- We do not really need $a \cdot d$ and $b \cdot c$ individually.
  We only need their addition, i.e. $a \cdot d + b \cdot c$.

- Can this be computed using a single multiplication?

- $(a + b) \cdot (c + d) - a \cdot c - b \cdot d$.

- Now we can compute all the terms we need using only 3 multiplications and a few additions and subtractions.
  Namely, these multiplications: $(a + b) \cdot (c + d), a \cdot c$, and $b \cdot d$.

Karatsuba's algorithm

$\texttt{Mult}((u, v))$

- Let a, b the first and the second half of u respectively.

- Let c, d the first and the second half of v respectively.

- Let $M_1 \leftarrow \texttt{Mult}(a, c)$, $M_2 \leftarrow \texttt{Mult}(a + b, c + d)$, and $M_3 \leftarrow \texttt{Mult}(b, d)$.

- Output $10^n \cdot M_1 + 10^{n/2} \cdot (M_2 - M_1 - M_3) + M_3$.

Karatsuba's algorithm

$\texttt{Mult}((u,v))$

- Let a, b the first and the second half of u respectively.
- Let c, d the first and the second half of v respectively.
- Let $M_1 \leftarrow \texttt{Mult}(a,c)$, $M_2 \leftarrow \texttt{Mult}(a+b, c+d)$, and $M_3 \leftarrow \texttt{Mult}(b,d)$.
- Output $10^n \cdot M_1 + 10^{n/2} \cdot (M_2 - M_1 - M_3) + M_3$.

Running time analysis of the algorithm.

$$\begin{aligned} T(n) &= 3 \cdot T(n/2) + O(n) \\ &= 3 \cdot (3 \cdot T(n/4) + O(n/2)) + O(n) \\ &= \vdots \\ &= O(n^{\log 3}) = O(n^{1.584}) \end{aligned}$$

The story behind Integer Multiplication

Andrei kolmogorov conjectured that the high school algorithm cannot be improved further.

He organised a seminar (with multiple future meetings) at Moscow State University to prove this and related conjectures. (1960)

Within less than a week a student (Anatoly Karatsuba) found a clever algorithm. (1960)

Andrei Toom broke the numbers down into 3 parts and showed that 5 multiplications are enough.
Thereby getting $T(n) = 5 \cdot T(n/3) + O(n) = n^{\log_3 5}$. (1963)

The story behind Integer Multiplication

Stephen Cook observed that Andrei Toom's idea can be generalised.

Thereby getting the time down to $O(C(r) \cdot n^{\log_r (2r-1)})$ (1966).

Schönhage and Strassen broke the $n^{1+\varepsilon}$ barrier.

They got the running time down to $O(n \log n \log \log n)$ (1971).

The next breakthrough was due to Martin Fürer.

He got the time down to $O(n \cdot \log n \cdot 2^{O(\log^* n)})$ (2007).

The story behind Integer Multiplication

Finally the best imaginable result for integer multiplication has been proved.

Harvey and Hoeven gave an upper bound of $O(n \cdot \log n)$ as recently as (2019)!