Algorithms & Complexity

How quickly we can/can't solve problems

Complexity: Broad classification of problems

Algorithms: Focuses within P.

Insertion Sort vs Merge Sort
\[ O(n^2) \quad O(n \log n) \]

Key Topics:
- Dynamic Programming
- Graph Algorithms
- Linear Programming
Example Algorithm

Multiplication

\[ \begin{array}{c}
\text{331} \\
\times \text{389} \\
\hline
\text{24} \\
\text{24} \\
\hline
\text{2649} \\
\text{2649} \\
\text{993} \\
\hline
\text{129428}
\end{array} \]

\[ n \text{ intermediate products of length } n \]

How many steps for \( n \) digit multiplication?

First: \( n \) digit addition:

- single pass, right to left, constant time each
- \( \Rightarrow O(n) \) time.

Multiplication:

- multiply all pairs
- \( \Rightarrow O(n^2) \) time

Can we do better?
Karatsuba multiplication

\[
\begin{array}{l}
A \quad = \quad 10^{n/2} \cdot A_1 + A_2 \\
B \quad = \quad 10^{n/2} B_1 + B_2 \\
AB \quad = \quad 10^n \cdot A_1 B_1 + 10^{n/2} (A_1 B_2 + B_1 A_2) + A_2 B_2
\end{array}
\]

Number of digit products $= \frac{n}{2}$ digit products $+ 3$ length-$O(n)$ additions

\[
f(n) = 4f\left(\frac{n}{2}\right) + O(n) \quad = \quad O(n^2)
\]

$(A_1 + A_2)(B_1 + B_2) = A_1 B_1 + A_2 B_2 + A_1 B_2 + B_1 A_2$

So sufficient to compute 3 products of \(\frac{n}{2}\) digits:

$A_1 B_1, A_2 B_2, \text{ and } (A_1 + A_2)(B_1 + B_2)$.

\[
f(n) = 3f\left(\frac{n}{2}\right) + O(n) \quad = \quad O(n^{1.585})
\]
Fancier Results

Toom-Cook:

split 3 ways, do 5 multiplications

$O(n^{\log_3 5}) \approx n^{1.465}$

4 ways, do 7 multiplications

$n^{\log_4 7} \approx n^{1.4103}$

K ways, do $2K - 1$ multiplications

$c(K) \cdot n^{\log (2K - 1)}$ time

↑ exponential in K.

Schnönhage-Strassen: $O(n \log n \log \log n)$ with an FFT

Simpler algorithms: better constants

Fancier algorithms: better asymptotics.