Divide and Conquer Algorithms

Main Idea:

1. Divide the problem into smaller subproblems
2. Solve each subproblem recursively
3. Combine the solutions of the subproblems in a "meaningful" way

Runtime Analysis:

* Tools to solve recurrence relations
* The "Master Theorem"

The Master Theorem

Story: Divide-and-conquer algorithm breaks a problem of size n into:

- k smaller problems
- each one of size n/b
- with cost of $O(n^d)$ to combine the results together

Formally: Consider the recurrence relation $T(n) = kT(n/b) + O(n^d)$, when $k, b > 1$. Then:

$$T(n) = \begin{cases} O\left(n^d\right) & \text{if } \left(k/b^d\right) < 1 \\ O(n^d \log n) & \text{if } \left(k/b^d\right) = 1 \\ O(n^{\log_b k}) & \text{if } \left(k/b^d\right) > 1 \end{cases}$$

Integer Multiplication

- Given n-digit *positive* integers x and y
- **Goal:** compute $x * y$
- **Easy:** do "grade-school" method
- **Q:** What's the runtime?
  - $O(n^2)$ (yikes)

NaiveMult(x, y):
$r = 0$
for $i = 1..n$:
  $r \mathrel{+}= (x \cdot y[i]) \ll (i-1)$
return r

Shorthand for:
$x + x + \cdots + x$
($y[i]$ times)

		3	4
*		3	9
	3	0	6
1	0	**2**	
1	3	**2**	6

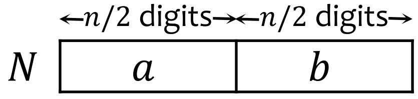
Splitting a Number

- $376280 = 3 \cdot 10^5 + 7 \cdot 10^4 + 6 \cdot 10^3 + 2 \cdot 10^2 + 8 \cdot 10^1 + 0 \cdot 10^0$

  $= (3 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0) \cdot 10^3 + 2 \cdot 10^2 + 8 \cdot 10^1 + 0 \cdot 10^0$

  $= 376 \cdot 10^3 + 280$

- **Observation 1:** N an n-digit number (assume n is even)
- N can be split into $n/2$ low-order digits & $n/2$ high-order digits:
  - $N = a \cdot 10^{n/2} + b$



Divide and Conquer Multiplication

- **Input:** N_1 and N_2, two n-digit numbers (assume n is a power of 2)
- Split N_1 and N_2 into $n/2$ low-order digits & $n/2$ high-order digits:
  - $N_1 = a \cdot 10^{n/2} + b$
  - $N_2 = c \cdot 10^{n/2} + d$

	← $n/2$ digits →	← $n/2$ digits →
N_1	a	b
N_2	c	d

- Compute $N_1 \times N_2 = a \times c \cdot 10^n + (a \times d + b \times c) \cdot 10^{n/2} + b \times d$
  - $m_1 = (a + b) \times (c + d)$ time: $O(n) + T(n/2)$
  - $m_2 = a \times c$ time: $T(n/2)$
  - $m_3 = b \times d$ time: $T(n/2)$
  - **Return:** $m_2 \cdot 10^n + (m_1 - m_2 - m_3) \cdot 10^{n/2} + m_3$. time: $O(n)$
- $T(n)$ = time to multiply two n-digit numbers
  - $T(n) = 3T(n/2) + O(n) \Longrightarrow k = 3, b = 2 \Longrightarrow$
    $T(n) = O\left(n^{\log_2 3}\right) = O(n^{1.585})$.

Divide and Conquer Multiplication

- **Conclusions & Remarks:**
  - Karatsuba algorithm (1962) was the first known algorithm for multiplication that is asymptotically faster $O\left(n^{\log_2 3}\right) = O(n^{1.585})$ than long multiplication.
  - Extending some ideas, the fastest multiplication $O(n \log n)$ algorithm was recently (2019) devised by Harvey and van der Hoeven!