DS 320 Algorithms for Data Science **Lecture #12 Worksheet**
Spring 2024 Prof. Kira Goldner

Divide & Conquer III: Integer and Matrix Multiplication

The Problem: Integer Multiplication

Your input for the *integer multiplication* problem is two n-digit numbers x and y. The goal is to output their product, $x \cdot y$.

What's the **naïve algorithm** and what's its **running time**?

[*Hint:* Seriously, don't think too hard about which algorithm. You can imagine the digits are numbered $x = x_n x_{n-1} \cdots x_1$ and $y = y_n x_{n-1} \cdots y_1$. What's the algorithm for multiplication? What's the running time?]

Step 1: Define your recursive subproblem.

One idea is to split each number into two parts: $x = 10^{n/2}a + b$ and $y = 10^{n/2}c + d$. Then

$$xy = 10^n ac + 10^{n/2}(ad + bc) + bd.$$

Additions and multiplications by powers of 10 (just shifts) are linear-time, so this reduces the problem to smaller multiplication problems:

$$T(n) = a\,T(n/b) + \Theta(f(n)).$$ What are a, b, and $f(n)$?

Which gives what running time?

The Speed Up:

We actually only need to make *three* recursive calls: ac, bd, and $(a+b)(c+d)$.

Step 2: Combine the solutions to your subproblems.

Show why this is enough to compute xy as above.

Then, what's our running time, via our number of subproblems, size of subproblem, and running time to divide/combine? Figure out these parameters and then solve the recurrence.

$T(n) = a\,T(n/b) + O(f(n))$ What are a, b, and $f(n)$?

$\Longrightarrow$

Matrix Multiplication: Strassen's Algorithm

The Problem: Matrix Multiplication

Your input for the *matrix multiplication* problem is two $n \times n$ matrices A and B. The goal is to output their product, $C = AB$. Recall that the ik^{th} entry of C is given by $c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$.

What's the **running time** of the **naïve algorithm** here and why?

Step 1: Define your recursive subproblem.

We divide each matrix into four submatrices.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

What running time does this give us?

$T(n) = a\,T(n/b) + \Theta(f(n))$ What are a, b, and $f(n)$?

$\Longrightarrow$

The Speed Up:

We compute only the following products.

- $P1 = A(F - H)$
- $P2 = (A + B)H$
- $P3 = (C + D)E$
- $P4 = D(G - E)$
- $P5 = (A + D)(E + H)$
- $P6 = (B - D)(G + H)$
- $P7 = (A - C)(E + F)$

Step 2: Combine the solutions to your subproblems.

Show why this is enough to solve the matrix multiplication problem.

Then the running time:

$$T(n) = a\,T(n/b) + \Theta(f(n)) \qquad \text{What are } a, b, \text{ and } f(n)?$$
$$\Longrightarrow$$

Reference

Theorem 1 (The Master Theorem)**.** *Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the non-negative integers by the recurrence*

$$T(n) = a\,T(n/b) + f(n),$$

where we interpret n/b as $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{if } f(n) = O(n^{\log_b a - \varepsilon}) \text{ for a constant } \varepsilon > 0 \\ \Theta(n^{\log_b a} \log_2 n) & \text{if } f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & \text{if } f(n) = \Omega(n^{\log_b a + \varepsilon}) \text{ for a constant } \varepsilon > 0 \text{ and} \\ & \quad af(n/b) \leq cf(n) \text{ for a constant } c < 1 \text{ and for all sufficiently large } n. \end{cases}$$