Algorithms Spring 2024

Tutorials — ~~Friday — 2:30 — 3:20pm~~
Tues/Th — after 2:00pm
Saturday / Friday morning

Motivation

1. Composability
2. Scaling — Complexity
3. Design better algorithms

Multiplication

Input: Two n-bit numbers n_1, n_2

Output: $n_1 \times n_2$

Attempt 1 We count n_1, n_2 times.

How much time does it take to add?
$O(n)$
Doing it n_2 times

$$O(n \times n_2) = O(n\,2^n)$$: max value of n_2 is 2^n

Attempt 2 Grade school approach 41×42

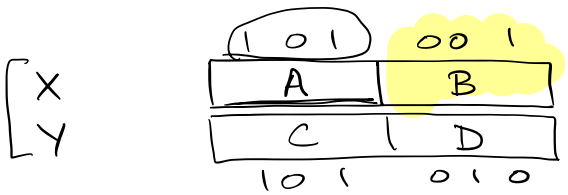
```
        1 0 1 0 0 1
      x 1 0 1 0 1 0
      -------------
        1 0 1 0 0 1
    1 0 1 0 0 1
+ 1 0 1 0 0 1
  -------------------
  1 1 0 1 0 1 1 1 0 1 0
```

Each addition is $O(n)$ & atmost n additions
$O(n^2)$

Karatsuba Algorithm (Anatoli Karatsuba, 1962 Russia)

Two inputs X & Y

Use Divide & Conquer



101 , 101000

$X = 2^{n/2}A + B$
$Y = 2^{n/2}C + D$
} Division condition

$$XY = 2^n AC \oplus 2^{n/2} BC \oplus 2^{n/2} AD \oplus BD \quad (*)$$

6 ← 110
12 1100

Original problem of multiplying two length n numbers is now 4 mult. of length $n/2$

$$T(n) = 4T\left(\frac{n}{2}\right) + cn$$

$T(n)$: Amount of work done on input of length n

$4T(\frac{n}{2})$: 4 mult. of size $n/2$

cn: some linear amount of work for additions and shifts

{3 $O(n)$ add
{3 shifts

$T(n) \in O(n^2)$ Why?

We will see how to solve recurrences later!

Can we do something differently?

Rewrite (*) as

$$XY = (2^n - 2^{n/2})AC + 2^{n/2}\underbrace{(A+B)(C+D)}_{E \times F} + (1 - 2^{n/2})BD$$

$$= 2^n AC - 2^{n/2}AC + 2^{n/2}AC + 2^{n/2}AD + 2^{n/2}BC + 2^{n/2}BD$$

$$+ BD - 2^{n/2}BD$$

Observe: Mathematical re-expression has saved us 1 multiplication.

$$T(n) = 3T\left(\frac{n}{2}\right) + c'n$$

$$T(n) \in O(n^{\lg_2 3}) \approx O(n^{1.585})$$

- Karp — Fast Fourier Transf. $O(n \lg^2 n)$
- Schönhage & Strassen, 1971 — $O(n \lg n \lg\lg n)$
- Fürer, 2007 — $O(n \lg n \, 2^{O(\lg^* n)})$

Open Qs: Is $O(n \lg n)$ possible?

Matrix Multiplication — Strassen – 1969

Multiply two matrices of size $n \times n$

Matrices $\leftarrow X, Y$

the individual entries of the matrices are assumed small, i.e, added & multiplied in constant time.

$X \times Y = Z \longrightarrow$ entries of $z_{ij} = \left(\text{—— } x_i \text{ ——}\right) \cdot \begin{pmatrix} | \\ y_j \\ | \end{pmatrix}$

multiply row i with col j to get z_{ij}

What is the standard Matrix Multiply algorithm complexity — $O(n^3)$

Why $\longrightarrow$ n^2 entries, $O(n)$ work for each of them.

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, \quad Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

A, B, C, D, E, F, G, H are sub-matrices.

$$XY = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

$$T(n) = 8\, T\left(\frac{n}{2}\right) + cn^2$$

$$q_1 = (A+D)(E+H), \quad q_2 = D(G-E), \quad q_3 = (B-D)(G+H)$$

$$q_4 = (A+B)H, \quad q_5 = (C+D)E, \quad q_6 = A(F-H)$$

$$q_7 = (C-A)(E+F)$$

$$XY = \begin{pmatrix} q_1+q_2+q_3-q_4 & q_4+q_6 \\ q_2+q_5 & q_1-q_5+q_6+q_7 \end{pmatrix}$$

$q_1+q_2+q_3-q_4 = AE+AH+DE+DH+DG-DE+BG+BH-DG-DH$

$-AH-BH$

$\longrightarrow T(n) = 7T\left(\frac{n}{2}\right) + cn^2$

$T(n) \in O(n^{\log_2 7}) \approx O(n^{2.81})$

1987 – Coppersmith-Winograd – $O(n^{2.376})$ – not practical

2011/2014 – Virginia Williams – used tensors for matrix multiplications – $O(n^{2.3737})$ overheads are too large.

2022 – Google – Alpha tensor $\longrightarrow$ found a decomposition with 47 multiplications

Consider 4×4 matrix multiplication

Strassen – 49 multiplications