6.046 — DC, Strassen, Fibonacci, Polynomial Multiplication (1)

Lecture 3

Divide and conquer

1) divide the problem (instance) into ≥ 1 subproblems
2) conquer each subproblem recursively
3) combine solution

Merge sort:

1) divide: trivial — $1 \quad \lfloor \frac{n}{2} \rfloor \; \lfloor \frac{n}{2} \rfloor + 1 \quad n$
2) conquer: recursively sort each subarray
3) combine: linear time merge

Running time: $\lfloor \, \rfloor \; \lceil \, \rceil$ don't matter

$$T(n) = \underbrace{2\,T\left(\frac{n}{2}\right)}_{\text{size of subproblem}} + \underbrace{\Theta(n)}_{\text{divide \& conquer time}}$$

case 2 $(k = 0)$

$$T(n) = \Theta(n \lg n)$$

Binary search:

find x in sorted array

1) divide: compare x with middle
2) conquer: recurse in one subarray
3) combine: trivial

$$T(n) = 1\,T\left(\frac{n}{2}\right) + \Theta(1)$$

$$n^{\log_2 1} = n^0 = 1$$

case 2, $k = 0$: $\Theta(1) = \Theta(n^0 \lg^0 n) \Rightarrow T(n) = \Theta(\lg n)$

Powering a number:

given number x

integer $n \geq 0$, compute x^n

Naive alg: $\underbrace{x \cdot x \cdot x \cdots}_{n} = x^n \quad \Theta(n)$

divide and conquer

$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & \text{if } n \text{ even} \\ x^{\frac{n-1}{2}} \cdot x^{n-1/2} \cdot x & \text{if } n \text{ is odd} \end{cases}$$

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

case 2, $k = 0$: $T(n) = \Theta(\lg n)$

My comment: can also take powers of two representation of n

[redacted]

$\Rightarrow \leq \lg n$ addition of powers of 2; only the largest power of two needs to be computed and intermediate values stored.

Fibonacci numbers

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

Naive recursive:

time $\Omega(\varphi^n)$ $\quad \varphi = \frac{1+\sqrt{5}}{2} > 1$

Bottom-up algorithm:

compute $F_0, F_1, F_2 \ldots F_n$ $\quad$ time $\Theta(n)$

Naive recursive squaring

$F_n = \varphi^n / \sqrt{5}$ rounded to nearest integer $\rightarrow F_n$

floating point ... not allowed

Recursive squaring

Thm: [redacted] $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$

implies $\Theta(\lg n)$ time

Proof: by induction on n

base: $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^1 = \begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix}$ ✓

step: $\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \underbrace{\begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}}_{= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \text{ by IH}} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

$\Rightarrow \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$

My comment

$\begin{vmatrix} (1-\lambda) & 1 \\ 1 & (0-\lambda) \end{vmatrix} = 0$

$-\lambda(1-\lambda) + 1 = 0$

$\lambda = \frac{1 \pm \sqrt{1+4}}{2}$

$\lambda_1 = 1.618\ldots$

$\lambda_2 = -0.618\ldots$

floats

diagonalization for squaring would be on floats not integers

My comment

not considered the growth of binary representation of F_n

Matrix multiplication

Input: $A = [a_{ij}]$ $B = [b_{ij}]$

Output: $C = [c_{ij}] = A \cdot B$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Standard alg. $\Theta(n^3)$

for $i \leftarrow 1$ to n
  do for $j \leftarrow 1$ to n
    do for $k \leftarrow 1$ to n
$c_{ij} \leftarrow 0$ do $c_{ij} \leftarrow a_j + a_{ik} b_{kj}$

Divide and conquer alg:

Idea: $n \times n$ matrix
$= 2 \times 2$ block matrix of $\frac{n}{2} \times \frac{n}{2}$ submatrices

$$\underbrace{\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right]}_{C} = \underbrace{\left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right]}_{A} \cdot \underbrace{\left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]}_{B}$$

$r = ae + bg$
$s = af + bh$
$t = ce + dg$
$u = cf + dh$

8 recursive multiplications of $\frac{n}{2} \times \frac{n}{2}$ matrices + 4 additions $\Theta(n^2)$

$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$
$= \Theta(n^3)$ case 1, not better

Strassen's algorithm

Idea: reduce # of multiplications $\rightarrow 7$

$P_1 = a \cdot (f - h)$
$P_2 = (a+b) \cdot h$
$P_3 = (c+d) \cdot e$
$P_4 = d(g - e)$
$P_5 = (a+d)(e+h)$
$P_6 =$ [redacted] $(b-d)(g+h)$
$P_7 = \cancel{f}(a-c)(e+f)$

$r = P_5 + P_4 - P_2 + P_6$
$s = P_1 + P_2$
$t = P_3 + P_4$
$u = P_5 + P_1 - P_3 - P_7$

check u

$u = (\cancel{ae} + \cancel{ah} + \cancel{de} + dh) + (\cancel{af} - \cancel{ah}) - (\cancel{ce} + \cancel{de})$
$- (\cancel{ae} + \cancel{af} - \cancel{ce} - cf) = dh + cf$

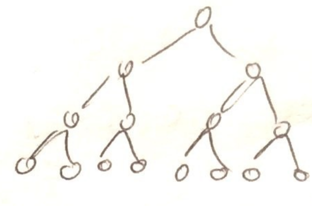
Strassen

1) divide A, B
[redacted] compute terms for products $\Theta(n^2)$
2) conquer recursively computing $P_1 \dots P_7$
3) combine r, s, t, u $\Theta(n^2)$

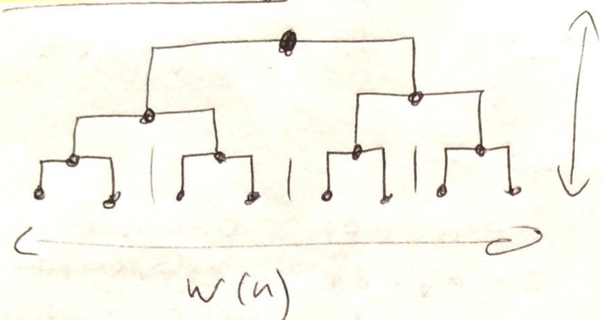
$T(n) = 7T(\frac{n}{2}) + \Theta(n^2) = \Theta(n^{\lg 7}) = \Theta(n^{2.81})$ $n \geq 32$ get improvements

VLSI layout (Very Large Scale Integration)

Problem: Embed a complete binary tree on n leaves in a grid with minimum area (bounding box)
constraints: 1) orthogonal edges, 2) no crossing wires



Naive embedding



one subtree — have to add 1

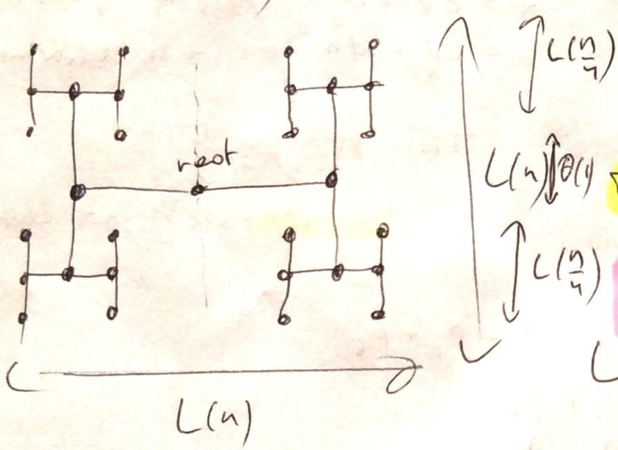
$$H(n) = H\left(\tfrac{n}{2}\right) + \Theta(1) = \Theta(\lg n)$$

$$W(n) = 2W\left(\tfrac{n}{2}\right) + \Theta(1) = \Theta(n)$$

do not have to add 1

Naive Area $= \Theta(n \lg n)$

Goal: $W(n) = \Theta(\sqrt{n})$
$H(n) = \Theta(\sqrt{n})$
$\Rightarrow$ Area $= \Theta(n)$



e.g. $\log_4 2 = \frac{1}{2}$, then $n^{\log_4 2} = \sqrt{n}$, other a, b possible

$$T(n) = 2T\left(\tfrac{n}{4}\right) + O(n^{1/2-\varepsilon})$$

need case 1!

recurrence provides a formulation of design constraints.

$$L(n) = 2L\left(\tfrac{n}{4}\right) + \Theta(1) = \Theta(\sqrt{n})$$ ✓ case 1

from a recurrence [redacted]
to a design [redacted] that is scalable