LECTURE 2
8/28/18

- USEFUL FACTS

$(a^b)^c = (a^c)^b = a^{bc}$

$a^{\log_b c} = c^{\log_b a}$

→ proof

- note $a = b^{\log_b a}$

$(b^{\log_b a})^{\log_b c} = (b^{\log_b c})^{\log_b a}$

- RECURRENCE

- think of tree

c → helpful to introduce constant

$T(n) = 4 \cdot T(\frac{n}{2}) + O(n)$

$T(n)$ → 1 problem of size n at cost $c \cdot n$ | $1 \cdot cn$

$T(\frac{n}{2})\ T(\frac{n}{2})\ T(\frac{n}{2})\ T(\frac{n}{2})$ → 4 problems of size $\frac{n}{2}$: each $c \cdot \frac{n}{2}$ | $4 \cdot c\frac{n}{2}$

$T(\frac{n}{4}) \dots$ → 16 problems of size $\frac{n}{4}$: each $c \cdot \frac{n}{4}$ | $16c\frac{n}{4}$

⇒ generalize

→ 4^i problems of size $\frac{n}{2^i}$ each at cost $c\frac{n}{2^i}$ | $4^i c \frac{n}{2^i}$

- runtime

$T(n) = 1cn + 4c\frac{n}{2} + 16c\frac{n}{4} + \dots + 4^i c \frac{n}{2^i}$

$= cn\left(1 + (\frac{4}{2})^1 + (\frac{4}{2})^2 + \dots + (\frac{4}{2})^i + \dots + (\frac{4}{2})^{\log_2 n}\right)$

$= cn(1 + 2 + 4 + \dots + n) \le cn(2n) = O(n^2)$

$n = 2^{\log_2 n}$; bound $2n$ because $1 + \frac{1}{2} + \frac{1}{4} \dots \le 2$

→ last term is $\log_2 n$

- every time we go down the layer, problem size halves

→ base case can be solved linearly

- when $T(n) = 3T(\frac{n}{2}) + cn$

$= cn\left(1 + (\frac{3}{2})^1 + (\frac{3}{2})^2 + \dots + (\frac{3}{2})^i + \dots + (\frac{3}{2})^{\log_2 n}\right)$

$= cn\left(1 + (\frac{3}{2})^1 + (\frac{3}{2})^2 + \dots + \frac{n^{\log_2 3}}{n}\right)$

$\le cn\left(\frac{n^{\log_2 3}}{n}\right) = cn^{\log_2 3}$

BINARY SEARCH

- divide and conquer
- find key k in $z[0, 1, \dots n-1]$ in sorted order

→ compare k with $z[\frac{n}{2}]$ and depending on result repeat it on lower or upper half

$T(n) = T(\frac{n}{2}) + O(1) = O(\log n)$

- NOTE

$1 + v + v^2 + \dots + v^m$

$v > 1$	$O(v^m)$
$v = 1$	$O(m)$
$v < 1$	$O(1)$

$a^{\log_b n} = n^{\log_b a}$

↳ width

$T(n)$

$T(\frac{n}{b}) \quad \cdots \quad T(\frac{n}{b})$

$T(\frac{n}{b^2}) \cdots$

$\log_b n$

↳ height of tree ↳ or depth

- **GENERAL RECURSION**

$$\boxed{T(n) = a\,T\left(\frac{n}{b}\right) + \underset{cn^d}{\underset{\shortparallel}{O(n^d)}}}$$

→ 1 problem of size n at cost cn^d	$1 \cdot cn^d$
→ a problems of size $\frac{n}{b}$ at cost $c(\frac{n}{b})^d$	$a \cdot c(\frac{n}{b})^d$
→ a^2 problems of size $\frac{n}{b^2}$ at cost $c(\frac{n}{b^2})^d$	$a^2 c(\frac{n}{b^2})^d$
→ a^i problems of size $\frac{n}{b^i}$ at cost $c(\frac{n}{b^i})^d$	$a^i c(\frac{n}{b^i})^d$

$$T(n) = cn^d\left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2 + \dots + \left(\frac{a}{b^d}\right)^i + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$

↳ because we divide n by b in every step

⇒ **DISCUSS CASES - MASTER THEOREM**

- **CASE 1** - bottom heavy tree

$$\frac{a}{b^d} > 1 \qquad cn^d\left(\frac{a}{b^d}\right)^{\log_b n} = O\left(\frac{n^d\, n^{\log_b a}}{b^{\log_b n^d}}\right)$$

→ most of the work is done on leaves $= O\left(\frac{n^d\, n^{\log_b a}}{n^d}\right) = O(n^{\log_b a})$

- **CASE 2** - balanced tree → all operations cost the same

$$\frac{a}{b^d} = 1 \qquad cn^d \log_b n = O(n^d \log n)$$

- **CASE 3** - top heavy tree → most of work is done on top

$$\frac{a}{b^d} < 1 \qquad cn^d \cdot O(1) = O(n^d)$$

- trees have different assymptotic behavior

- **MATRIX MULTIPLICATION**
  - input: 2 $n \times n$ matrices
  - output: $n \times n$ matrix

$$i\,\boxed{x}\;\boxed{y}^{\,j} = \boxed{z_\circ}_{\,ij}$$

$$z_{ij} = \langle x[i,*], y[*,j] \rangle = \sum_{k=1}^{n} x_{ik} y_{kj}$$

→ algorithm

$$\left.\begin{array}{ll} \forall i \in [n] & \times n \\ \quad \forall j \in [n] & \times n \\ \qquad z = \sum x_{ik} y_{kj} & \} O(n) \end{array}\right\} O(n^3)$$

⇒ RECURSIVE ALGORITHM

- divide matrices to blocks

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

→ divided problem into 8 subproblems

$$\boxed{T(n) = 8\,\underbrace{T\left(\tfrac{n}{2}\right)}_{\text{get } \frac{n}{2}\times\frac{n}{2} \text{ matrix}} + \underbrace{O(n^2)}_{\text{adding matrices}}}$$

→ use theorem

$a=8,\ b=2,\ d=2$

- ratio $\frac{a}{b^d} = \frac{8}{2^2} = 2 > 1$

- runtime $O(n^{\log_2 8}) = O(n^3)$

→ to get better runtime than $O(n^3)$

- we have to reduce $T(n) = 8\,T(\frac{n}{2}) + O(n^2)$

⇒ BETTER ALGORITHM - STRASSEN

$P_1 = A(F-H)$

$P_2 = (A+G)H$

⋮

$$\begin{bmatrix} P_5+P_4-P_2+P_6 & P_1+P_2 \\ P_3+P_4 & P_1+P_5-P_1-P_7 \end{bmatrix}$$

$$\boxed{T(n) = 7\,T(\tfrac{n}{2}) + O(n^2)}$$

$$= O(n^{\log_2 7}) = O(n^{2.81})$$

$\frac{a}{b^d} = \frac{7}{2^2} > 1 \quad O(n^{\log_2 7})$

- If you were to find a way to multiply $k \times k$ matrices in k^ω multiplications, then you can obtain

$$T(n) = k^\omega\, T\left(\tfrac{n}{k}\right) + O(n^2)$$

$$= O(n^\omega)$$