LECTURE 3

Outline:- 1) SOLVING RECURRENCES

2) MATRIX MULTIPLICATION

3) MEDIAN

Solving Recurrences

Example 1: $T[n] = T[n-1] + \sqrt{n}$

(unroll the recursion)

$$T[n] = T(n-1) + \sqrt{n}$$

$$= T(n-2) + \sqrt{n-1} + \sqrt{n}$$

$$= T[1] + \sqrt{2} + \sqrt{3} + \ldots + \sqrt{n}$$

* Each term $< \sqrt{n} \Rightarrow T[n] < n \cdot \sqrt{n} = n^{1.5}$

* Look at last $n/2$ terms

$$T(n) \geq \sqrt{n/2} + \sqrt{\frac{n}{2}+1} \ldots + \sqrt{n}$$

$\geq \frac{n}{2}$ terms each at least $\sqrt{\frac{n}{2}}$

$$\geq \frac{n}{2} \cdot \sqrt{\frac{n}{2}} = \frac{n^{1.5}}{2\sqrt{2}}$$

So $n^{1.5} \geq T(n) \geq n^{1.5}/2\sqrt{2}$

$$T(n) = 2T(n/3) + n$$

$$= 2\left[2T(n/9) + n/3\right] + n$$

$$= 2^2 T(n/3^2) + \frac{2n}{3} + n$$

$$= 2^2\left[2T\left(n/3^3\right) + \frac{n}{3^2}\right] + \frac{2n}{3} + n$$

$$= 2^3 T\left(n/3^3\right) + \frac{2^2}{3^2}n + \frac{2n}{3} + n$$

$$= 2^k T\left(n/3^k\right) + \left[\left(\frac{2^{k-1}}{3^{k-1}}\right).n + \cdots + \left(\frac{2}{3}\right)n + n\right]$$

$K = \log_3 n \iff 3^k = n$

$\Theta(\text{last term}) = O(n)$

$$= 2^{\log_3 n} + \Theta(n)$$

$$= n^{\log_3 2} + O(n) = \Theta(n)$$

MASTER THEOREM:

Suppose function $T: \mathbb{N} \to \mathbb{R}^+$ satisfies

$$T[n] = a\,T\left(\frac{n}{b}\right) + O(n^c) \quad \text{then}$$

where $b > 1$, $a, c > 0$ are constants

CASE 1: $c < \log_b a$ $\qquad T(n) = O(n^{\log_b a})$

CASE 2: $c = \log_b a$ $\qquad T[n] = O(n^c \log n)$

CASE 3: $c > \log_b a$ $\qquad T[n] = O(n^c)$

Matrix Multiplication

Input: X, Y $n \times n$ matrices

Goal: Compute $Z = X \cdot Y$

Z_{ij} = Inner product of i^{th} row of X & j^{th} column of Y

Naive Algo:

Compute all Z_{ij}'s using $\Theta(n)$ time separately

Runtime = $n^2 \cdot \Theta(n) = \Theta(n^3)$

Can we beat $\Theta(n^3)$??

DIVIDE AND CONQUER

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

X Y

$Z = X \cdot Y$

$n \times n$ matrix mult

$\Longrightarrow$ 8 $n/2 \times n/2$ matrix mult

$+ \Theta(n^2)$ additions

$$T(n) = 8T(n/2) + \Theta(n^2)$$

$$\Rightarrow T(n) = \Theta(n^3).$$

Strassen's Algorithm (1969)

Matrix-Multiply(X, Y : $n \times n$ matrices)

- Think of X, Y as a 2×2 array of $\frac{n}{2} \times \frac{n}{2}$ matrices

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

- Compute matrix products (using recursion),

$$P_1 = A(F - H) \quad , \; P_2 = (A + B)H \quad , P_3 = (C + D)E \, , P_4 = D(G - E)$$

$$P_5 = (A + D)(E + H) \, , P_6 = (B - D)(G + H) \, , P_7 = (A - C)(E + F)$$

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

MEDIAN

INPUT: A list of n numbers [NOT SORTED]

OUTPUT: Find the $\lceil n/2 \rceil^{th}$ smallest number

Example: $[5, 7, 8, 11, 10, 9, 3, 2, 12]$

NAIVE ALG: Sort the list & output the $\lceil n/2 \rceil^{th}$ entry in sorted list

$\Theta(n \log n)$

NOW: Randomized Algorithm running in $O(n)$ time

SELECT

INPUT: 1) An UNSORTED list of numbers

$\{a_1 .. a_n\}$

2) Integer k in $[1..n]$

OUTPUT: k^{th} smallest number among $\{a_1 .. a_n\}$

REMARK: $\text{MEDIAN}(a_1 .. a_n) = \text{SELECT}(\{a_1 .. a_n\}, \lceil \frac{n}{2} \rceil)$

SELECT $(\{a_1 .. a_n\}, k)$

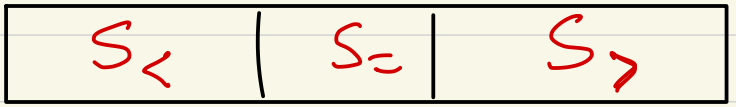
* PIVOT $v \leftarrow$ random element from $\{a_1 .. a_n\}$

* SPLIT $S_< \leftarrow \{a_i \mid a_i < v\}$

$S_= \leftarrow \{a_i \mid a_i = v\}$

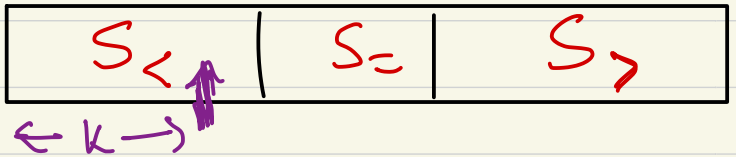
$S_> \leftarrow \{a_i \mid a_i > v\}$

IDEA: IMAGINE THE SORTED LIST



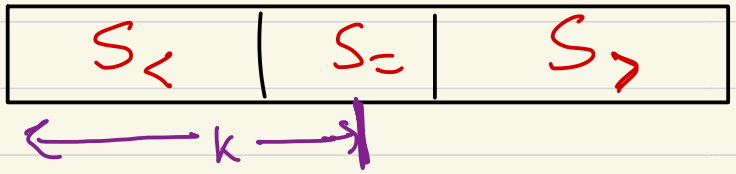
* CASES:

- $k \leq |S_<|$: return SELECT$(S_<, k)$



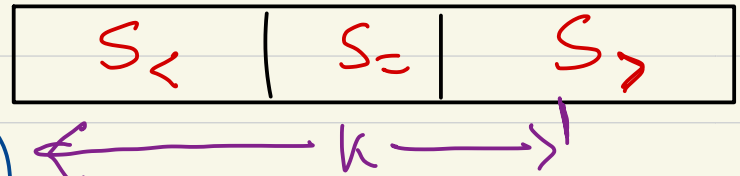
- $|S_<| < k \leq |S_<| + |S_=|$:

return v



- $|S_<| + |S_=| < k$

return SELECT$(S_>, k - |S_<| - |S_=|)$



Runtime Analysis

Randomised Algorithm: runtime depends on pivot choices.

For example:

→ If pivot $v = k^{th}$ smallest element

⇒ SELECT runs in CASE 2, returns $O(n)$ time

→ Imagine $k = n/2$ and each pivot is largest or smallest element

$\Theta(n^2)$ time

→ Happens with very low probability

Obs: 1) List size decreases to $\leq 3n/4$

whenever pivot is GOOD.

2) # of good pivots $= n/2$