13 Jan 2025 - Algorithms

- Stock purchase problem ; inductive / iterative approach
  store (OPT ($p_1, \dots, p_{n-1}$)) and update on seeing p_n

Multiplication

- School method : $O(n^2)$ time

$$A = \underbrace{A_L}_{\text{first}} \,\|\, \underbrace{A_R}_{\text{last}}$$

$$A = 2^{n/2} A_L + A_R$$

$$B = 2^{n/2} B_L + B_R$$

$$AB = 2^n A_L B_L + 2^{n/2} \underbrace{(A_L B_R + A_R B_L)}_{\text{find with 1 mult}} + A_R B_R$$

Four recursive MULT : $A_L B_L$, $A_L B_R$, $A_R B_L$, $A_R B_R$

$O(n)$ additions

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

$$\max\left(4^{\log_2 n}, n\right)$$

$$O(n^2)$$

$A_L B_L$, $A_R B_R$

want : $A_L B_R + A_R B_L$

$$(A_L + A_R)(B_L + B_R) = A_L B_L + A_R B_R + (A_L B_R + A_R B_L)$$

$$(A_L + A_R)(B_L + B_R) - A_L B_L - A_R B_R = A_L B_R + A_R B_L$$

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

Asymptotically,

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

$$\max\left(3^{\log_2 n}, n\right)$$

$$O\left(n^{\log_2 3}\right) = O\left(n^{1.585}\right)$$

- dividing into 3 parts:

$$A = \underbrace{A_1\ A_2\ A_3}_{\text{each } \frac{n}{3} \text{ bits}}$$

$$B = B_1\ B_2\ B_3$$

$$T(n) = 9\,T\left(\frac{n}{3}\right) + O(n)$$
$$= O(n^2)$$

$$A = 2^{\frac{2n}{3}} A_1 + 2^{n/3} A_2 + A_3$$
$$B = 2^{\frac{2n}{3}} B_1 + 2^{n/3} B_2 + B_3$$

Matrix Multiplication

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ & \vdots & & \\ A_{n1} & & & A_{nn} \end{bmatrix} \begin{bmatrix} B_{11} & \cdots & B_{1n} \\ \vdots & & \vdots \\ B_{n1} & & B_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} C_{11} & & \\ & \ddots & \\ & & C_{nn} \end{bmatrix}$$

$\left\{ \begin{array}{l} O(n) \text{ mult} \\ \quad + O(n) \text{ add} \end{array} \right.$

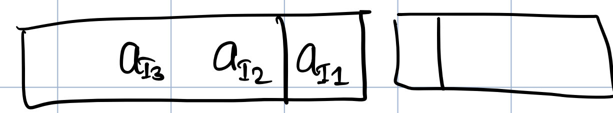
n^2 entries : each using $O(n)$ operations

$O(n^3)$ total

$$\begin{array}{cc} & \frac{n}{2} \\ \frac{n}{2} & \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right] \end{array} \left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right]$$

$$= \begin{bmatrix} A_1 B_1 + A_2 B_3 & A_1 B_2 + A_2 B_4 \\ A_3 B_1 + A_4 B_3 & A_3 B_2 + A_4 B_4 \end{bmatrix}$$

- Recursively find all products



- Add appropriate matrices

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

$$= O(n^3)$$

$$A_1B_1 + A_2B_3$$

$$\alpha = \underbrace{(A_1 + A_2)(B_1 + B_3)}_{1} - \underbrace{A_1B_3}_{2} - \underbrace{A_2B_1}_{3}$$

$$A_1B_2 + A_2B_4$$

$$\beta = \underbrace{(A_1 + A_2)(B_2 + B_4)}_{4} - A_1B_4 - A_2B_2$$

$$A_3B_1 + A_4B_3$$

$$\gamma = \underbrace{(A_3 + A_4)(B_1 + B_3)}_{5} - \underbrace{A_3B_3}_{6} - \underbrace{A_4B_1}_{7}$$

$$A_3B_2 + A_4B_4$$

$$\delta = \underbrace{(A_3 + A_4)(B_2 + B_4)}_{7} - \underbrace{A_3B_4}_{8} - \underbrace{A_4B_2}_{9}$$

$$\alpha + \gamma = \left(\sum A\right)\left(B_1 + B_3\right) - \left(A_1 + A_3\right) B_3 \quad 2$$
$$1 \qquad\qquad - \left(A_2 + A_4\right) B_1 \quad 3$$

$$\beta + \delta = \left(\sum A\right)\left(B_2 + B_4\right) - \left(A_1 + A_3\right) B_4 \quad 4$$
$$6 \qquad\qquad - \left(A_2 + A_4\right) B_2 \quad 5$$

$$\alpha + \beta + \gamma + \delta = \left(\sum A\right)\left(\sum B\right) - \left(A_1 + A_3\right)\left(B_3 + B_4\right)$$
$$- \left(A_2 + A_4\right)\left(B_1 + B_2\right)$$

$$\alpha + \beta = \left(A_1 + A_2\right)\left(\sum B\right) - \left(B_3 + B_4\right) A_1$$
$$- \left(B_1 + B_2\right) A_2$$

$$T(n) = 7\,T(n/2) + O(n^2)$$

$$7^{\log_2 n}$$

$$n^{\log_2 7}$$

Current best for matrix mult : $O(n^{2.37})$

Horse problem

$A_1 > B_1 > C_1 > D_1 > E_1$

$$\begin{array}{ccccccccc}
A_1 & > & \boxed{A_2} & > & \boxed{A_3} & > & A_4 & > & A_5 \\
\boxed{B_1} & > & \boxed{B_2} & > & B_3 & > & B_4 & > & B_5 \\
\boxed{C_1} & > & C_2 & > & C_3 & > & C_4 & > & C_5 \\
D_1 & > & D_2 & > & D_3 & > & D_4 & > & D_5 \\
E_1 & > & E_2 & > & E_3 & > & E_4 & > & E_5
\end{array}$$

A_2

A_3

B_1

B_2

C_1

Selection problem

Input : $A[1, 2, \dots, n]$, $k \in \mathbb{N}$

↓

array of integers

o/p : k^{th} smallest element

useful if i/p changes dynamically

- Heaps → Creation : $O(n)$
- BSTs → Creation : $O(n \log n)$
- Sorting → $O(n \log n)$
- Find smallest and remove } k times | $O(kn)$

Heaps: $O(k \log n + n)$

↳ Good when k is small

$k = n/2$ (median) ⇝ as good as sorting

Goal: find k^{th} smallest element in $O(n)$ time.

16 Jan 2025

Integer multiplication : Karatsuba's algorithm

$$A = A_L \| A_R = A_L \cdot 2^{n/2} + A_R$$

$$B = B_L \| B_R = B_L \cdot 2^{n/2} + B_R$$

Find $A_L B_L$, $A_R B_R$

$$(A_L + A_R) \cdot (B_L + B_R)$$

$$T(n) = 3\,T\left(\frac{n}{2}\right) + O(n)$$

$$n^{\log_k (2k-1)}$$

$$= O\left(n^{\log_2 3}\right)$$

$$A = A_1 A_2 A_3 = A_1 \cdot 2^{2n/3} + A_2 \cdot 2^{n/3} + A_3$$

$$B = B_1 B_2 B_3 = B_1 \cdot 2^{2n/3} + B_2 \cdot 2^{n/3} + B_3$$

$$T(n) = 9\,T\left(\frac{n}{3}\right) + O(n)$$

$$= O(n^2)$$

k parts $\longrightarrow$ $2k-1$ multiplications

Matrix Multiplications

$$\left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right]\left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right]$$

$$T(n) = 7\,T(n/2) + O(n)$$

$$= O(n^{\log_2 7}) = O(n^{2.8\ldots})$$

Matrix multiplication

→ 1969 : Strassen

$S_1 = A_1 + A_2$

$S_2 = A_1 - A_2$

$\cdot = B_1 + B_3$

$\vdots = B_2 + B_4$

$\vdots$

$S_{10} = \ldots$

$P_1 =$

$P_2 =$

$\vdots$

$P_7 =$

Integer multiplication

→ Karatsuba 1961

→ Based on FFT 1962

$O(n \log n \log \log n)$

Strassen

Given $A[1, 2, \ldots, n]$, find the k^{th} smallest element

Suppose we have an algorithm B which can find the median in $O(n)$ time.

Heaps: $O(k \log n + n)$

extracting smallest takes $\log n$ time

$A_1, A_2, \ldots, A_{1000}$. Find 100^{th} shortest element

$\left.A_1' \leq A_2' \leq \cdots \leq A_{1000}'\right\}$ sorted

median finding algorithm can get us $\underset{\text{using B}}{\boxed{A_{500}'}}$ in $O(n)$

Compare with A_{500}' : $A_1'', A_2'', \ldots, A_{500}'$

Using B

$\boxed{A_{250}'}$

$\{ A_1' , \ldots , A_{250}' \}$

Use $B \rightarrow A_{125}'$

$\{ A_1' , \ldots , A_{125}' \} \rightsquigarrow$ elements $\leq A_{125}'$

Use $B \rightarrow A_{63}'$

$\{ A_{63}' , \ldots , A_{125}' \} \rightsquigarrow$ elements $\geq A_{63}'$

Use $B \rightarrow A_{94}'$

$$T(n) = \underbrace{O(n)}_{\text{time taken by } B} + \underbrace{O(n)}_{\text{comparing with median}} + T(n/2)$$

$$\Rightarrow \quad T(n) = 1 \cdot T\left(n/2\right) + cn$$

$$= cn + \frac{cn}{2} + \frac{cn}{4} + \cdots \leq 2cn = O(n)$$

$A_1, A_2, \ldots, A_{1000}$

To find A_{100}, add 800 elements smaller than the minimum

call B $\rightarrow$ returns 900^{th} smallest element of new list

$= 100^{th}$ smallest element of old list

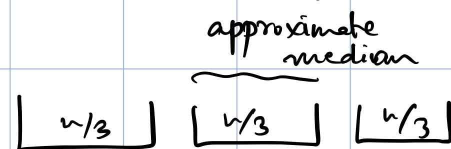
$\text{Rank}_A(x) = 1 + \#$ of elements $< x$ in A

Smallest element has rank 1

largest element has rank n

Approximate Median: x is an appropriate median for array A

if $\frac{n}{3} \leqslant \text{rank}_A(x) \leqslant \frac{2n}{3}$



If we have an algorithm C which can return some approximate median in $O(n)$ time. Then we can find the k^{th} smallest element in $O(n)$ time.

$A_1 \quad , \quad A_2 \quad , \quad \ldots \quad , \quad A_{1000}$

$A_1' < A_2' < \ldots < A_{1000}'$

B returns $\quad A_{450}' = x$

$\underbrace{\text{elements} \leq x}_{450}$

compare all elements with x

$$T(n) \leq \underbrace{O(n)}_{\text{time for comparisions}} + T\left(\frac{2n}{3}\right) = cn + \frac{2cn}{3} + \frac{4cn}{9} + \cdots$$

$$= 3\,cn = O(n)$$

Idea: Find an approximate median by making a recursive call to a subset of A.

SELECT (A, n, k)

SELECT ($A[\ldots]$, some length of the indices , $\ell/2$)

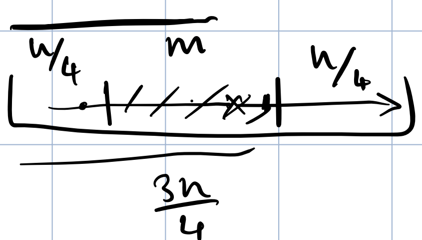
some indices

use this as the approximate median

Use median of $A[1, 2, \ldots, n/2] \longrightarrow m$

$\geq n/4$ are less than m

$\geq n/4$ are greater than m



SELECT (A , n, k) :

Input : A , $|A| = n$

Output : k^{th} smallest element of A

$\frac{3n}{4}$

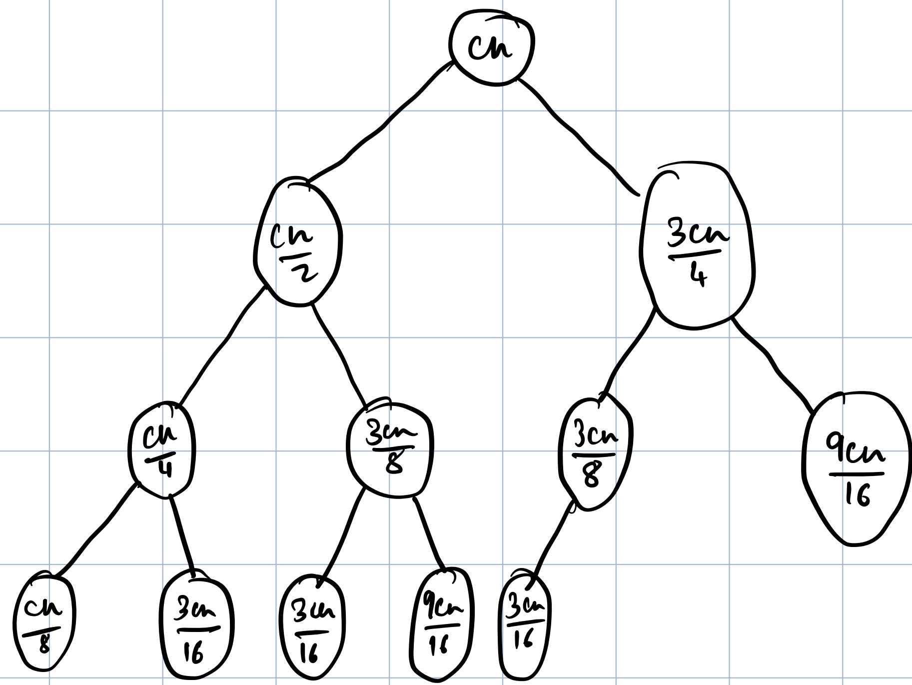
1. Find m = SELECT $\left(\left[1, \dots, \frac{n}{2}\right], \frac{n}{2}, \frac{n}{4} \right)$
2. Find $L = \{ x \in A : x \leq m \}$

   $R = \{ x \in A : x > m \}$
3. SELECT (L , |L| , k) or SELECT (R , |R| , k − |L|)

worst case

$$T(n) = T\left(\frac{n}{2}\right) + O(n) + T\left(\frac{3n}{4}\right)$$

improve



$$cn + \frac{cn}{2} + \frac{3cn}{4} + \frac{cn}{4} + \frac{3cn}{8} + \frac{3cn}{8} + \frac{9cn}{16}$$