QUICK SELECT.

X; PICK $k \in X$

$< k$ $\quad$ $> k$

k

x_1 $\qquad$ x_2

WORST CASE

RUN TIME $= \theta(n^2)$

Best Case $= \theta(n)$

AVERAGE RUN TIME

$= O(n)$.

BLUM, FLOYD, PRATT
RIVEST, TARJAN 1973:

PICK the PIVOT M
using the FOLLOWING
ALGORITHM & then
RUN Quick Select.

$X = k_1, k_2, .., k_n$

$$M$$

× × × ×
× × × ×
× M_1 × M_2 ... × ... × $M_{\frac{n}{5}}$
× × × ×
× × × ×
G_1 G_2 ... $G_{\frac{n}{5}}$

GROUP X into groups of
SIZE 5 EACH, FIND the
MEDIAN OF EACH GROUP.
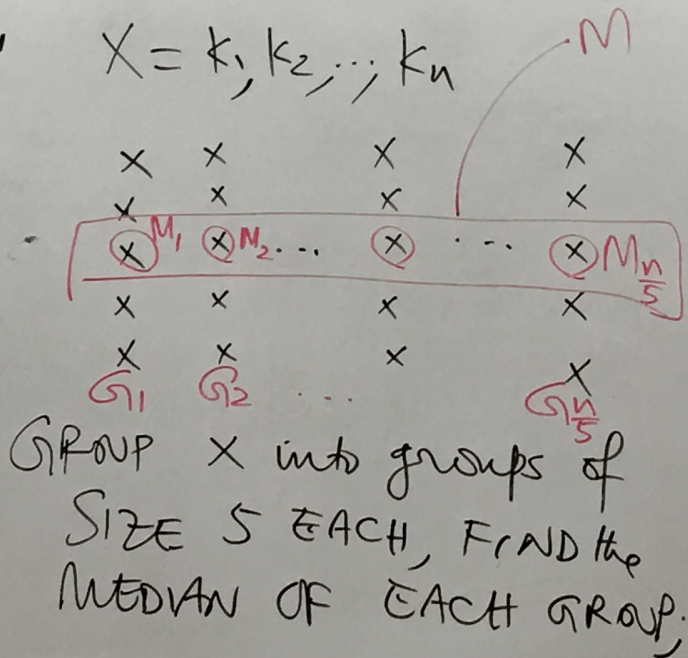
Recursively FIND the
MEDIAN M of these
Medians. Use M as the
PIVOT in Quick select.

How many groups have a MEDIAN $\leq M$?

There are $\frac{n}{10}$ such GROUPS.

$a, b, c \leq M_i \leq M.$

$$c \quad M_i \leq M$$

G_i

IN EACH of these $\frac{n}{10}$ GROUPS, there will be at least 3 elements $\leq M.$

$\Rightarrow$ There are $\geq \frac{3}{10} n$ elements of X that are $\leq M$.

$\Rightarrow |X_2| \leq \frac{7}{10} n$.

$\boxed{<M}$ $\boxed{>M}$ LIKEWISE WE CAN

X_1 X_2 SHOW: $|X_1| \leq \frac{7}{10} n$.

Let $T(n)$ be the RUN TIME OF THIS ALG. ON ANY INPUT OF SIZE n, ON ANY C.

Then $T(n) = 9 \cdot \frac{n}{5} + T\left(\frac{n}{5}\right) + n + T\left(\frac{7}{10} n\right)$

GROUP MEDIANS

PARTITION IN QuickSelect

$= T\left(\frac{n}{5}\right) + T\left(\frac{7}{10} n\right) + dn$, d being a constant.

CLAIM: $T(n) \le cn$ for some CONSTANT c.

PROOF BY INDUCTION:

INDUCTION STEP: Assume the hypothesis for all inputs of size up to $(n-1)$. We'll prove it for n.

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + dn$$
$$\le c \cdot \frac{n}{5} + c \cdot \frac{7}{10}n + dn \quad \text{(USING the HYPOTHESIS)}.$$

i.e., $T(n) \le 0.9cn + dn$

RHS $\le cn$ if $0.9cn + dn$

$\le cn \Rightarrow 0.1cn \ge dn$

$\Rightarrow \boxed{c \ge 10d}$.

$\Rightarrow T(n) \le 10dn = O(n)$.

ie, $T(n) \leq 0.9cn + dn$

RHS $\leq cn$ if $0.9cn + dn$

$\leq cn \Rightarrow 0.1cn \geq dn$

$\Rightarrow \boxed{c \geq 10d}$.

$\Rightarrow T(n) \leq 10dn = O(n)$.

THEOREM: Selection on n elements can be done in $O(n)$ TIME.

$\square$

FLOYD & RIVEST (1975):

1. PICK A RANDOM SAMPLE S FROM X

2. IDENTIFY TWO elements from X. Call these l_1 and l_2. l_1 & l_2 will be s.t. they bracket the ith smallest element of X

and $\left| \{ q \in X : l_1 \leq q \leq l_2 \} \right|$ is SMALL, with a high probability.

③ Separate the set
$$Y = \{ q \in X : l_1 \leq q \leq l_2 \}$$
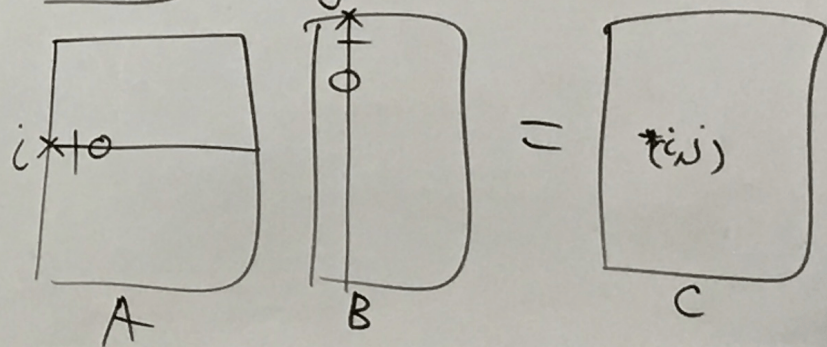④ Make Sure that the

i^{th} smallest element of X is in Y and perform an appropriate selection in Y.

# of COMPARISONS
$$= n + \text{Min} \{ i, (n-i) \} + \sigma(n)$$

MATRIX MULTIPLICATION

INPUT: $A_{n \times n}$, $B_{n \times n}$

OUTPUT: $C = AB$.

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

for $i = 1$ to n do
   for $j = 1$ to n do
      $C[i,j] = 0.0;$
      for $k = 1$ to n do
         $C[i,j] = C[i,j] + A[i,k] * B[k,j];$

RUN TIME OF this algorithm is $\Theta(n^3)$, SINCE WE SPEND n Multiplications and $(n-1)$ ADDITIONS FOR EACH element in the product
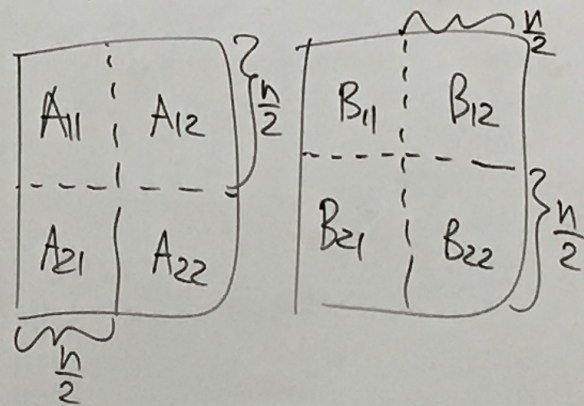
$$\text{Total} = n^2 (n + n - 1) = \Theta(n^3)$$

STRASSEN'S ALGORITHM.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

STRASSEN CAME UP WITH AN ALGORITHM TO MULTIPLY TWO 2×2 MATRICES USING 7 MULTIPLICATIONS.

and 19 ADDITIONS.

Let A and B be $n\times n$ MATRICES.

Let $T(n)$ be the RUN TIME of this alg. on any two $n\times n$ MATRICES.

Then, $T(n) = 7T\left(\dfrac{n}{2}\right) + 19 \cdot \dfrac{n^2}{4}$

$\qquad = 7T\left(\dfrac{n}{2}\right) + \Theta(n^2).$

$$a = 7, \quad b = 2, \quad f(n) = \Theta(n^2).$$

$$n^{\log_b a} = n^{\log_2 7}.$$

$$\Rightarrow T(n) = \Theta\left(n^{\log_2 7}\right).$$

V. PAN $\rightarrow O\left(n^{2.79}\right)$.

1983 COPPERSMITH & WINOGRAD

$$O\left(n^{2.376}\right).$$

2011. V. WILLIAMS.

$$O\left(n^{2.373}\right).$$

Worst case linear time selection

- In the last lecture we introduced the quick select algorithm and showed that it's worst case run time is $\Omega(n^2)$. It has a best case and average case run time of $O(n)$.

- We also started our discussion on the worst case linear time algorithm of BFPRT.

- The BFPRT algorithm is nothing but the quick select algorithm wherein the pivot is picked using a special algorithm.

- If $X = k_1, k_2, \ldots, k_n$ is the input sequence, the algorithm groups the elements of X into groups of size 5 each. Let these groups be $G_1, G_2, \ldots, G_{n/5}$. The median of each of these groups is found. Let these medians be $M_1, M_2, \ldots, M_{n/5}$, respectively. The median M of these medians is found recursively and used as the pivot.

- The entire algorithm can be summarized as follows:

  BFPRT(X, i)
      0) **if** $|X| = 1$ **then** output k_1 and quit;
      1) Group X into groups $G_1, G_2, \ldots, G_{n/5}$ each of size 5;
          Find the median M_i of G_i for $i = 1, 2, \ldots, \frac{n}{5}$;
          Let $S = \{M_1, M_2, \ldots, M_{n/5}\}$;
      2) $M =$BFPRT$(S, \frac{n}{10})$; (M is the median of group medians);
      3) Partition X into $X_1 = \{q \in X : q < M\}$ and $X_2 = \{q \in X : q > M\}$;
      4) **if** $|X_1| = i - 1$ **then** output M and quit;
          if $|X_1| \geq i$ **then** BFPRT(X_1, i);
          else BFPRT$(X_2, i - |X_1| - 1)$;

Analysis of the BFPRT algorithm

- The element M we pick as the pivot is expected to be an approximate median of X. In this case we can expect X_1 and X_2 to be nearly of the same size and subsequently, our divide and conquer algorithm can be expected to yield a good performance.

- We will prove that X_1 and X_2 will be nearly of the same size.

- Note that out of the $n/5$ groups we have created in step 1, half of the groups will have a median $\leq M$ and the other half of the groups will have a median $\geq M$. Let G_i be one group whose median is $\leq M$. If the median of G_i is M_i, then $M_i \leq M$. In this group G_i there are three elements that are less than or equal to M_i (by definition of the median) and all of these elements will also be $\leq M$ (since $M_i \leq M$). This means that at least $3 \times \frac{n}{10}$ elements of X will be $\leq M$. This in turn means that $|X_2| \leq \frac{7}{10}n$.

- Along the same lines, we can also show that $|X_1| \leq \frac{7}{10}n$.

- Now we are ready to write a recurrence relation for the run time of the BFPRT algorithm.

- Let $T(n)$ be the run time of the BFPRT algorithm on any input of size n and for any i.

- Step 1 takes $\Theta(n)$ time since we can find the median of each group in $\Theta(1)$ time. Step 2 takes $T(n/5)$ time. Partitioning in step 3 can be done in $\Theta(n)$ time. In step 4, in the worst case, we recurse on X_1 or X_2. We know that the size of neither is more than $\frac{7}{10}n$. As a result, we get:

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + \Theta(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + dn$$

  for some constant d.

- We can prove by induction that the above recurrence relation solves to: $T(n) = O(n)$.

- Induction Hypothesis: $T(n) \leq cn$ for some constant c. The base case can be proven easily.

- Induction step: Assume that the hypothesis holds for all the inputs of size up to $n - 1$. We'll prove it for inputs of size n.

- $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + dn \leq c\frac{n}{5} + c\frac{7}{10}n + dn = 0.9cn + dn$. The RHS will be $\leq cn$ if $c \geq 10d$.

- Thus we conclude that $T(n) \leq 10dn$, i.e., $T(n) = O(n)$. We get the following:

  Theorem. We can select the ith smallest element from any given sequence of n elements in $O(n)$ time. $\square$

A randomized algorithm

- Floyd and Rivest (1975) have given a randomized algorithm that employs random sampling. The steps in this algorithm are:

    1) Pick a random sample S from the input sequence X;
    2) Find two elements l_1 and l_2 from S such that:

the ith smallest element of X (call it q) has a value in the interval $[l_1, l_2]$ and $|\{x \in X : l_1 \le x \le l_2\}|$ is 'small' with a high probability.

3) Identify the set $Y = \{x \in X : l_1 \le x \le l_2\}$;

4) Make sure that q is in Y and perform an appropriate selection in Y;

- We can show that the number of comparisons made by the above algorithm is $n + \min\{i, n - i\} + \widetilde{o}(n)$. This is one of the best-known algorithms for selection.

Matrix Multiplication

- Matrix multiplication is a very important problem in science and engineering with numerous applications.

- Input for this problems are two matrices A and B of size $n \times n$ each. The goal is to compute the product C of A and B.

- We can come up with a simple cubic time algorithm for this problem as follows:

> **for** $i = 1$ **to** n **do**
>> **for** $j = 1$ **to** n **do**
>>> $C[i, j] = 0.0$;
>>> **for** $k = 1$ **to** n **do**
>>>> $C[i, j] = C[i, j] + A[i, k] * B[k, j]$;

- The above algorithm takes n multiplications and $n - 1$ additions for each output element and there are a total of n^2 elements to be output. Thus the total time is $n^2(2n-1) = \Theta(n^3)$.

Strassen's algorithm

- Strassen has given an elegant divide-and-conquer algorithm for matrix multiplication that takes subcubic time.

- Consider the problem of multiplying two 2×2 matrices. A straight forward algorithm for this problem will take 8 multiplications. Strassen has come up with a way of mutiplying them with only 7 multiplications.

- Let the matrices of interest be $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$.

- Strassen computed the product $C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$ of A and B as follows:

3

$d_1 = (a_{11} + a_{22})(b_{11} + b_{22});$
$d_2 = (a_{21} + a_{22})b_{11};$
$d_3 = a_{11}(b_{12} - b_{22});$
$d_4 = a_{22}(b_{21} - b_{11});$
$d_5 = (a_{11} + a_{12})b_{22};$
$d_6 = (a_{21} - a_{11})(b_{11} + b_{12});$
$d_7 = (a_{12} - a_{22})(b_{21} + b_{22});$
$c_{11} = d_1 + d_4 - d_5 + d_7;$
$c_{12} = d_3 + d_5;$
$c_{21} = d_2 + d_4;$
$c_{22} = d_1 - d_2 + d_3 + d_6.$

- If A and B are generic $n \times n$ matrices, we could use the above algorithm to derive a divide-and-conquer recursive algorithm to multiply them. W.l.o.g. assume that $n = 2^k$ for some integer k.

- Partition A and B as: $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$, where A_{ij} and B_{ij} are $\frac{n}{2} \times \frac{n}{2}$ submatrices, for $1 \le i, j \le 2$.

- After partitioning A and B as above, use Strassen's formulas to multiply A and B. What is a scalar multiplication (or addition) in the above formulas will now become a submatrix multiplication (or addition). Submatrix addition is easy. Submatrix multiplication is done recursively.

- Let $T(n)$ be the run time of Strassen's algorithm to multiply two $n \times n$ matrices. Then, we have:
$$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2).$$

- Note that we have to do seven submatrix multiplications (each taking $T\left(\frac{n}{2}\right)$ time) and 18 submatrix additions (each taking $\frac{n^2}{4}$ time).

- Using the Master theorem, we can solve for $T(n)$ to get: $T(n) = \Theta(n^{\log_2 7})$.

- Since the publications of Strassen's algorithm in 1969, a number of improvements have been made: V. Pan (1978): $O(n^{2.796})$; Coppersmith and Winograd (1983): $O(n^{2.376})$; V. Williams (2014): $O(n^{2.373})$.