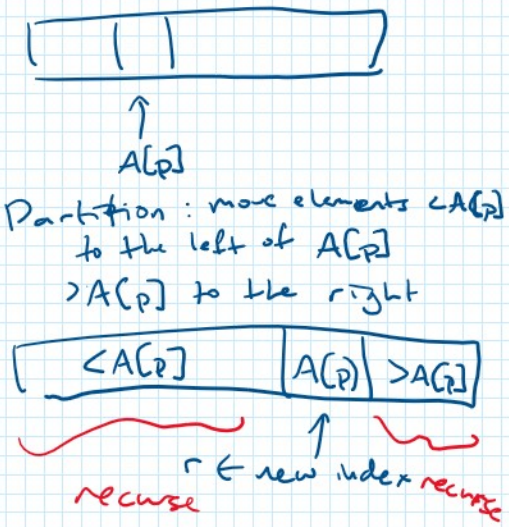
Lecture 2020-03-04

Thursday, 4 March, 2021 10:47

```
QuickSort(A[1..n])
  if n>1:
    Choose a pivot element A[p]
    r ← PARTITION(A,p)
    QuickSort(A[1..r-1])
    QuickSort(A[r+1..n])
```



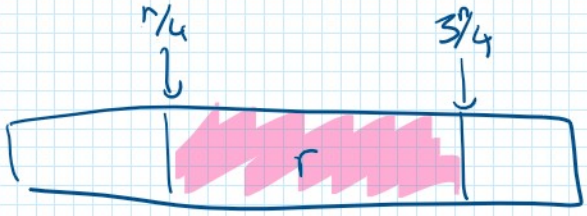
time of PARTITION

$$T(n) = O(n) + \max_r \{T(r-1) + T(n-r)\}$$

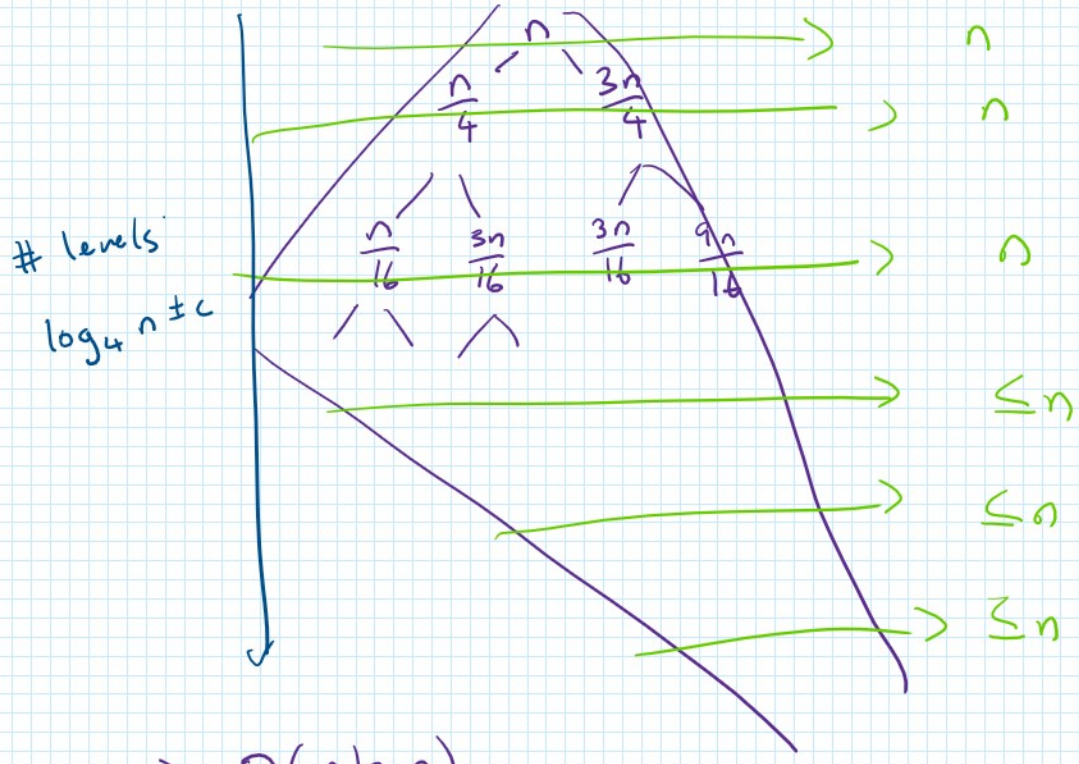
$$\leq O(n) + T(0) + T(n-1)$$

$$= O(n^2).$$

by magic (?)



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



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

$\longrightarrow O(n \log n)$

How to ensure pivot is somewhere in the middle?

CS 473 $\longrightarrow$ random pivot: in expectation $r = \frac{n}{2}$

w/ high probability

$\frac{n}{4} \le r \le \frac{3n}{4}$.

This is 374!

reduce this problem to SELECTION problem.

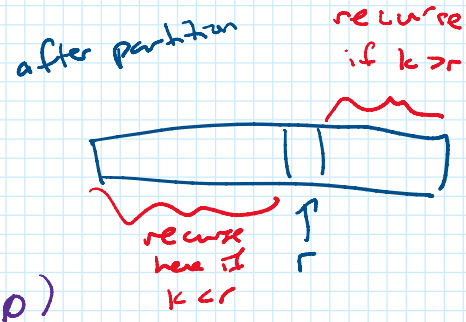
SELECTION $(A[1..n], k)$ returns k-th smallest element of A.

QuickSort: choose <u>median</u> as pivot

SELECTION $(A[1..n], \frac{n}{2})$.

following idea of QuickSort

```
QuickSelect(A[1..n], k)
    if base case
        brute force
    else
        Choose a pivot A[p]
        r ← PARTITION(A[1..n], p)
        if k < r
            return QuickSelect(A[1..r-1], k)
        else if k > r
            return QuickSelect(A[r+1..n], k-r)
        else (if k = r)
            return A[r]
```



worst case? $O(n^2)$.

$\longrightarrow$ CS 473 random pivot works w.h.p.

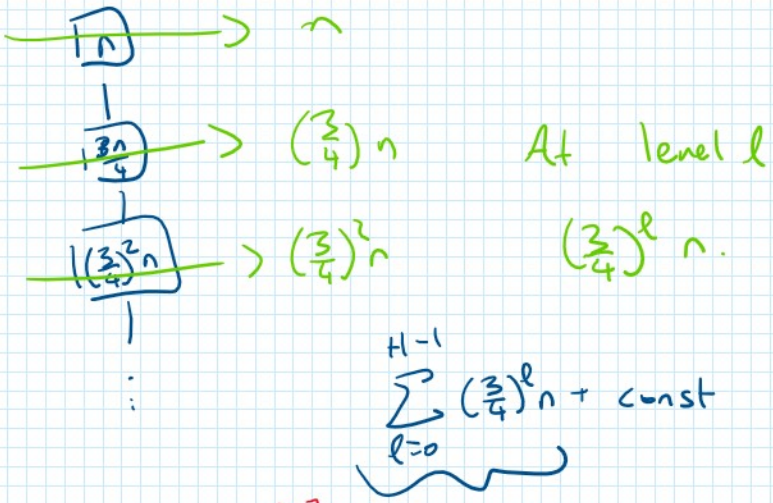
$\longrightarrow O(n)$ time.

magic:

$\frac{n}{4} < r < \frac{3n}{4}$ $\quad T(n) \le O(n) + T(\frac{3n}{4})$

magic

$\frac{n}{4} < r < \frac{3n}{4}$ $\quad T(n) \le O(n) + T(\frac{3n}{4})$



$\sum_{l=0}^{H-1} (\frac{3}{4})^l n + \text{const}$

Geometric series

If $|a| < 1$

$\sum_{i=0}^{\infty} a^i = \frac{1}{1-a}$

$n \sum_{l=0}^{H-1} (\frac{3}{4})^l \le n \sum_{l \ge 0}^{\infty} (\frac{3}{4})^l \le O(n)$

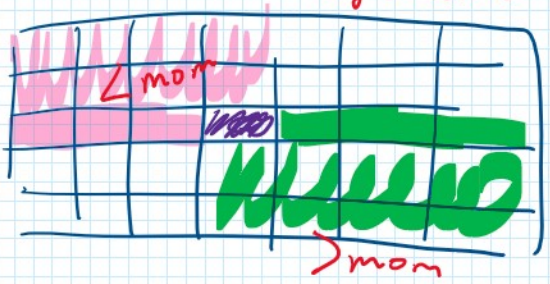
BFPRT / Median of Medians alg for SELECTION

Split $A[1..n]$ into $\frac{n}{5}$ chunks of size 5.

① Find the median of each chunk in $O(1)$ time.
($O(n)$ in total)

Imagine each chunk sorted, chunks sorted by median.

② find median of these medians recursively: $T(\frac{n}{5})$



③ use MoM as pivot

MoM Select($A[1..n], k$)
if base case
brute force

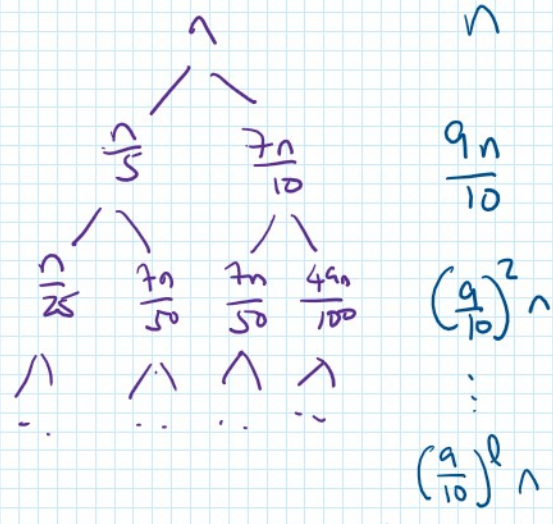
$\frac{3n}{10} < r < \frac{7n}{10}$

pink = # green
$= \frac{1}{2} \cdot \frac{3}{5} = \frac{3}{10}$

$$T(n) \le O(n) + T(\tfrac{n}{5}) + T(\tfrac{7n}{10})$$

if base case
    brute force
else
    $m \leftarrow n/5$
    for $i \leftarrow 1$ to m
        $M[i] \leftarrow$ Median of Five $(A[5i-4..5i])$
    mom $\leftarrow$ MoM Select $(M[1..m], m/2)$
    $r \leftarrow$ Partition $(A[1..n], mom)$
    if $k < r$
        return MoM Select $(A[1..r-1], k)$
    else
        :

$$T(n) \leq O(n) + T(\tfrac{n}{5}) + T(\tfrac{7n}{10})$$

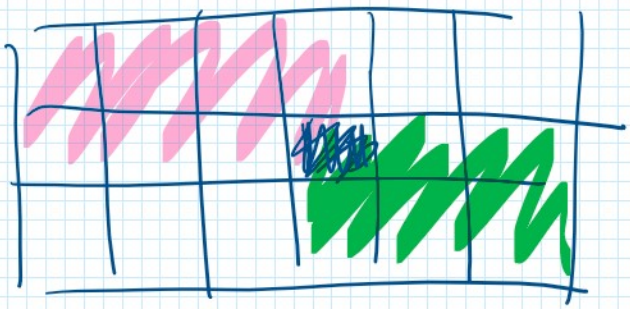


geometric series

$T(n) = O(n)$.

Why 5?
what if $\frac{n}{3}$ chunks of size 3

imagine sort each chunk
sort chunks by median



recurse on $\leq$ (not pink)

pink elements $\geq \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$

$$T(n) = O(n) + T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) = O(n \log n)$$

7? 3? 9?

→ median of odd size is "nice"

→ brute force median of m is faster when m is smaller

5 smallest odd number so that MoM Select is $O(n)$

5 smallest odd number so that MoM select is $O(n)$

QuickSort

in practice, $p=1$ "good in average"

in practice random p "very good, w.h.p."

in theory p = MoM "excellent in theory"
slow in practice.

!! working w/ bits for rest of lecture

solve multiplication "faster"

multiplying two n-digit numbers

```
  123
 x456
 ----
  738
 615
492
-----
56088
```

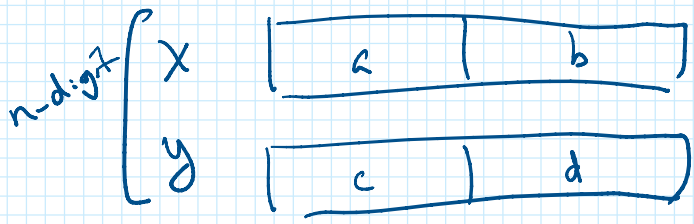
multiply 2 1-digit #s n^2 times.

(+ additions)

$\rightarrow O(n^2)$

Kolmogorov's conjecture $\Omega(n^2)$

Karatsuba (~23 yo?) $o(n^2)$



$x = a \cdot 10^{n/2} + b$

$y = c \cdot 10^{n/2} + d$

$$xy = \boxed{a}\boxed{b} \cdot \boxed{c}\boxed{d}$$

$$= (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d)$$

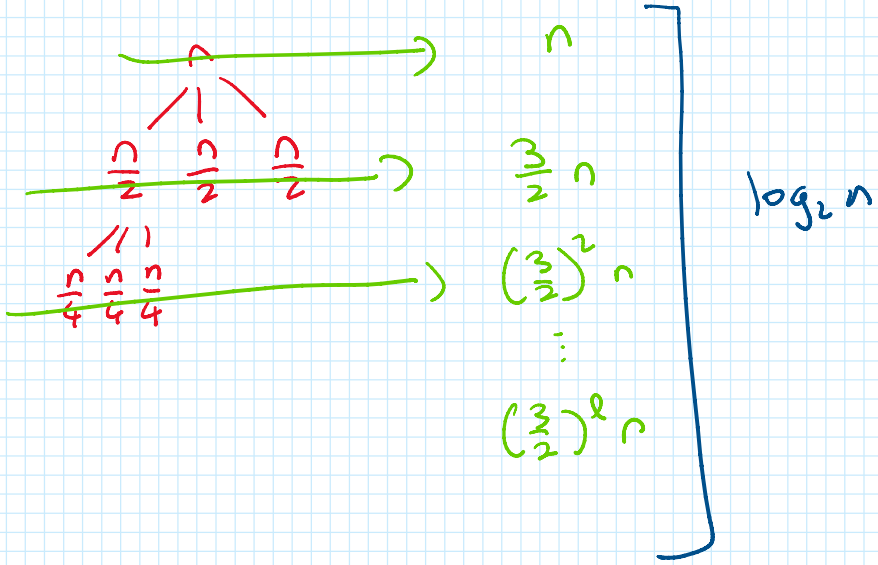
$$= ac \cdot 10^{n} + (ad+bc) \cdot 10^{n/2} + bd.$$

$$xy = \boxed{ac}\,\boxed{ad+bc}\,\boxed{bd}$$

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

$$(a-b)(c-d) = ac - (ad+bc) + bd$$

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



$$T(n) = n \sum_{i=0}^{H} c^i$$

$$= \Theta(c^H)$$

$$\sum_{\ell=0}^{H} \left(\frac{3}{2}\right)^\ell = \Theta\left(\left(\frac{3}{2}\right)^H\right)$$

$$= \Theta\left(\left(\frac{3}{2}\right)^{\log_2 n}\right)$$

$$= \Theta\left(n^{\log_2 \left(\frac{3}{2}\right)}\right)$$

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

$$= \Theta\left(n^{1.7\ldots}\right)$$

Fast Multiplication (x, y, n):

if $n < 4$

brute force

if $n < T$
brute force

$= \Theta(n^{1.5})$
$= \Theta(n^{1.7\ldots})$

else
$m \leftarrow \lceil n/2 \rceil$
$a \leftarrow \lfloor x/10^m \rfloor$; $b \leftarrow x \bmod 10^m$
$c \leftarrow \lfloor y/10^m \rfloor$; $d \leftarrow y \bmod 10^m$
$ac \leftarrow$ FastMultiply(a, c, m)
$bd \leftarrow$ FastMult(b, d, m)
$adbc \leftarrow ac + bd -$ FastMult$(a-b, c-d, m)$
return $ac \cdot 10^{2m} + adbc \cdot 10^m + bd$.

Can we do better than Karatsuba?

Yes... split into 3 chunks
grind at recurrence to get fewer than 9 recursive calls...

etc.

30 years before Karatsuba, Gauss
turn the crank on # chunks

↪ FFT. $O(n \log n \log\log n)$.

"but this is too slow to solve the problem I want"
→ paper to drawer
→ forgotten.

↪ in 2019, in fact "better FFT-like techniques"
→ $O(n \log n)$

Theory ~s Practice (orders of magnitude are approximate)

Theory vs Practice (orders of magnitude are approximate)

$x, y \geq 1000$	Karatsuba faster
$n \geq 10$	splitting to 3 chunks faster
$n \geq 1000$	FFT's faster
$n \geq 10^{100}$	$n \log n$ alg faster