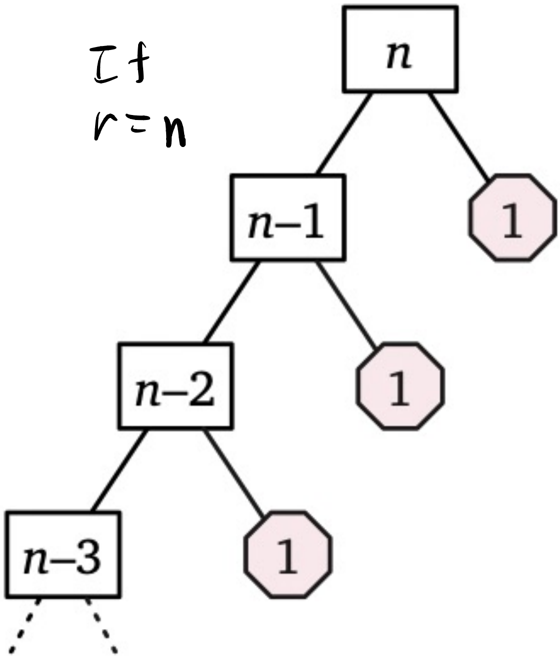
```
QuickSort(A[1..n]):
  if (n > 1)
      Choose a pivot element A[p]
      r ← Partition(A, p)
      QuickSort(A[1..r−1])      ⟨⟨Recurse!⟩⟩
      QuickSort(A[r+1..n])      ⟨⟨Recurse!⟩⟩
```

```
Partition(A[1..n], p):
  swap A[p] ↔ A[n]
  ℓ ← 0                 ⟨⟨#items < pivot⟩⟩
  for i ← 1 to n−1
      if A[i] < A[n]
          ℓ ← ℓ+1
          swap A[ℓ] ↔ A[i]
  swap A[n] ↔ A[ℓ+1]
  return ℓ+1
```

$$T(n) = \max_{1 \le r \le n} \left(T(r-1) + T(n-r) \right) + \Theta(n)$$

If
$r = n$



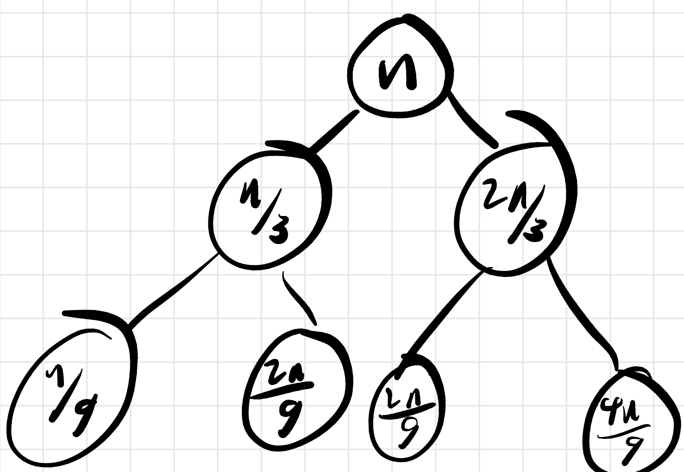
So $T(n) \ge c \cdot n + (n-1) + (n-2) + \dots$
$= \Omega(n^2)$

worst case really is in $\Theta(n^2)$

Usually, things are better.
Subproblems are ~ same size & recurrence is more like

$$T(n) \le T(n/3) + T(2n/3) + \Theta(n)$$

if $r \in [n/3, 2n/3]$



$\Sigma = n$

$\Sigma = n$

$\Sigma = n$

levels = $\log_{3/2} n = O(\log n)$

$T(n) = O(n \log n)$

full levels = $\log_3 n = \Omega(\log n)$

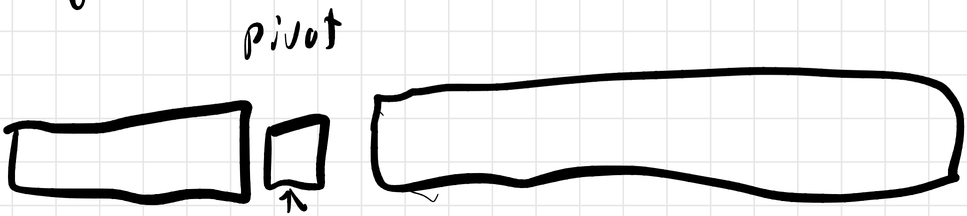
$T(n) = \Omega(n \log n)$

$\Rightarrow T(n) = \Theta(n \log n)$

Median Selection

(in this class) **median**: the element at index $\lceil n/2 \rceil$ after sorting

Hoare '61: (same paper as quicksort)



rank r

$r < \lceil n/2 \rceil$ so search left side.

need recursive calls to accept a **rank** as a parameter

Selection (formally):

Given $A[1..n]$ & k, return element of rank k

(lies in position k in sorted order)

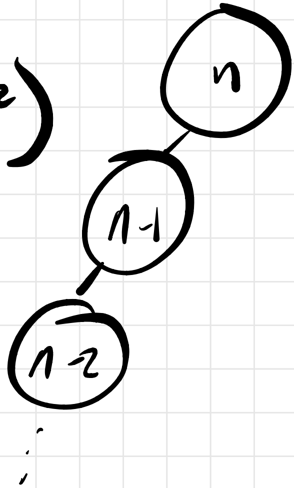
```
QuickSelect(A[1..n], k):
    if n = 1
        return A[1]
    else
        Choose a pivot element A[p]
        r ← Partition(A[1..n], p)
        if k < r
            return QuickSelect(A[1..r−1], k)
        else if k > r
            return QuickSelect(A[r+1..n], k−r)
        else
            return A[r]
```

rank of pivot

worst-case

$$T(n) = \max_{1 \le l \le n-1} T(l) + \Theta(n)$$

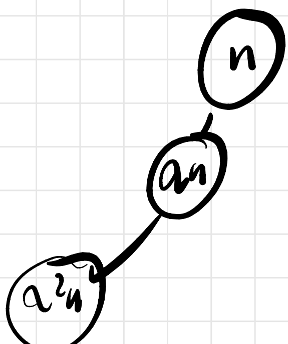
$T(n) = \Theta(n^2)$



if $l = n-1$ each time

Suppose $l = \alpha \cdot n$ for some constant $\alpha < 1$

$$T(n) \le T(\alpha n) + \Theta(n)$$



n

αn

$\alpha^2 n$

$\vdots$

decreasing geometric...

$$T(n) = \Theta(n)$$

Median of Medians

BFPRT '70s.

Divide input into $\lceil n/5 \rceil$ **blocks** of 5 elements each.
(assume $n \mid 5$)

Find median of each block.

Find median of these medians (mom) using our algorithm recursively.

Use mom as the pivot.

MomSelect(A[1..n], k):

moves rank k element to position k & returns it.

```
MOMSELECT(A[1 .. n], k):
  if n ≤ 25 ⟨⟨or whatever⟩⟩
      use brute force
  else
      m ← ⌈n/5⌉
      for i ← 1 to m
          MEDIANOFFIVE([A[5i − 4 .. 5i]) ⟨⟨Moves median to index 5i − 2.⟩⟩
          swap A[i] ↔ A[5i − 2]
      MOMSELECT(A[1 .. m], ⌈m/2⌉) ⟨⟨Recursion! Also, moves mom to index ⌈m/2⌉.⟩⟩

      r ← PARTITION(A[1 .. n], ⌈m/2⌉)

      if k < r
          return MOMSELECT(A[1 .. r − 1], k) ⟨⟨Recursion!⟩⟩
      else if k > r
          return MOMSELECT(A[r + 1 .. n], k − r) ⟨⟨Recursion!⟩⟩
      else
          return A[r]
```

Analysis

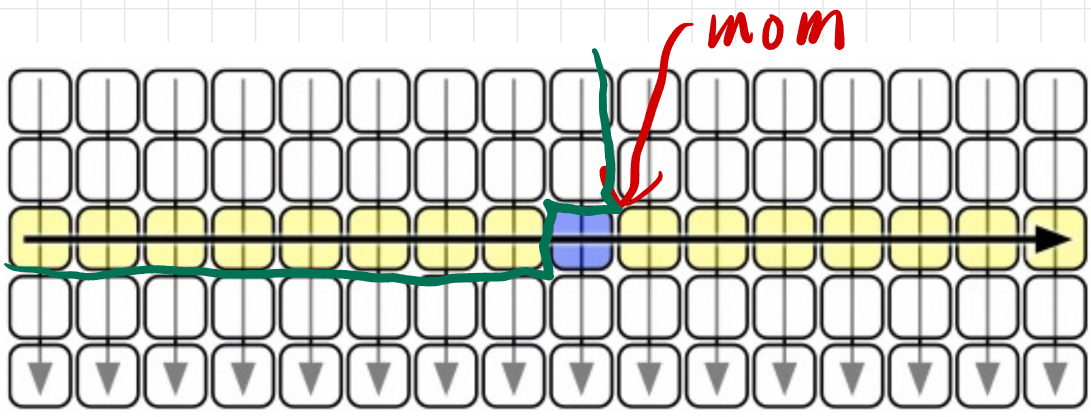
imagine...

Lay out A as a $5 \times \lceil n/5 \rceil$ grid.

Each column is one of the blocks of 5.

Sort each column $\begin{array}{c}\text{smallest}\\ \downarrow \\ \text{biggest}\end{array}$

Sort collection of columns by their medians.



mom $> \lceil \lceil n/5 \rceil / 2 \rceil$ # block medians

3 elements $\leq$ to each block median within a block

$\Rightarrow\ > \frac{3n}{10}$ elements $<$ mom

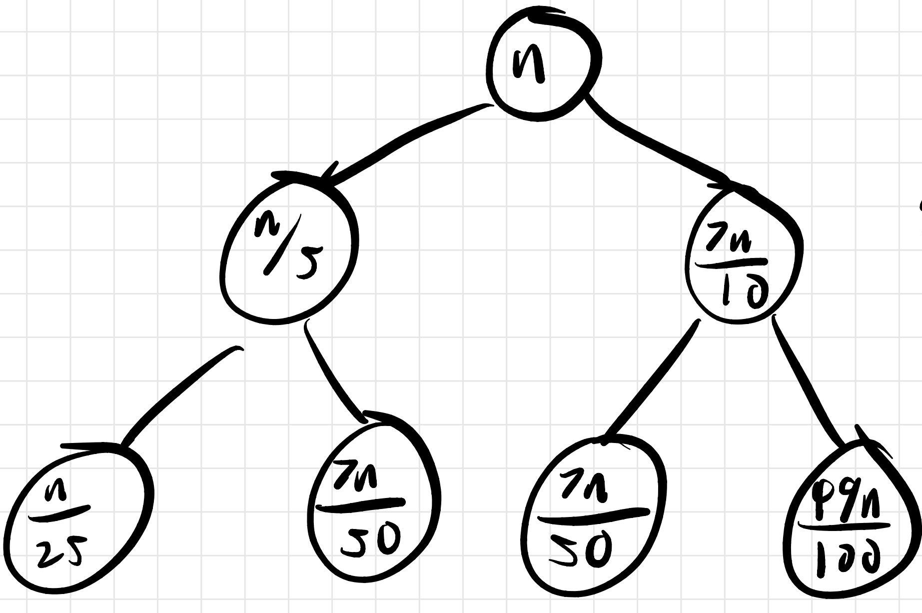
$\Rightarrow\ < \frac{7n}{10}$ elements $>$ mom

by symmetry, $< \frac{7n}{10}$ elements $<$ mom

So second call on

$\leq \frac{7n}{10}$ elements.

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



$\Sigma = n$

$\Sigma = \frac{9n}{10}$

$\Sigma = \frac{81n}{100}$

⋮

level i sums to $(9/10)^i n$

$T(n) = \Theta(n)$

Why 5?

5 is odd

so .n 3?

the second call would have size

$$n - (2 \cdot (n/3)/2)$$

$$= 2n/3$$

$$T(n) \le T(n/3) + T(2n/3) + \Theta(n)$$

$$= \Theta(n \log n)$$

Don't use MomSelect!

Pick a pivot uniformly at random.

For both QuickSort & Select.

Expected time is $O(n \log n)$