Order Statistics
(aka Selection Problems)

Part II: Linear-Time Median Finding

Select(list, pos)

Previously we attempted to…

Place the n elements of the list into groups of 3 and find the median of those groups and create Med3List.

MoM3=Select(Med3List, n/6);

Partition the original list around MoM3 into LeftList and RightList and figure out the position of MoM3.

if pos==MoM3pos then

DONE!

elseif pos<MoM3pos then

Select(LeftList, pos);

else

Select(RightList, pos-MoM3pos)

…but this ran worst-case O($n$log$n$) time.

Were we close?

We've seen via recurrence trees that eliminating some items as we go down level-by-level has some nice asymptotic advantages.

What if we could eliminate some more values before our recursion…

Select(list, pos)

Let’s try something a little different…

Place the *n* elements of the list into groups of 5 and find the median of those groups and create Med5List.

MoM5=Select(Med5List, ***n*/10**);

Partition the original list around MoM5 into LeftList and RightList and figure out the position of MoM5.

if pos==MoM5pos then

  DONE!

elseif pos<MoM5pos then

  Select(LeftList, pos);

else

  Select(RightList, pos-MoM5pos)

…how will this run in the worst-case?

How bad is that last call?

After partitioning around the MoM5, in the worst case possible, how many elements are there in the sub-list that we are going to call Select( ) on recursively?

What's The Worst Runtime?

Find the Med5s:	$\Theta(n)$
Find the MoM5:	$T(n/5)$
Partition around MoM5:	$\Theta(n)$
Worst Case Recursion:	$T(7n/10)$

It’s linear!

Next, let’s try to narrow-in on the constant coefficient…