Selection Problem: Find the $k^{th}$ smallest number in a list (Assume distinct numbers).

ex: 19 13 18 12 17 16 15, $k=6 \Rightarrow$ ans = 16

Simple solution:
- sort list, return $k^{th}$ element $\Rightarrow \mathcal{O}(n \log n)$

Can actually be solved in $\mathcal{O}(n)$!
Surprising b/c problem seems harder than finding min/max, but is as computationally easy!

Algorithm is similar to Quicksort.

Quick Quicksort Review (overview on slides):

ex: 19 13 18 12 17 16 15

(1) Choose a pivot, ex: 15

(2) Partition list around pivot into $L, G$

\[
\begin{array}{ccccccc}
19 & 13 & 18 & 12 & 17 & 16 & 15 \\
\uparrow & & & & & & \\
\downarrow & & & & & & \\
\end{array}
\]

move $i$ right, move $j$ left until


swap
Eventually: 12 13 15 19 17 16 18
\[ QS() \]
\[ QS() \]
(3) Recursively call \( QS() \) on \( L \) and \( G \)

\textbf{Time?} \quad \text{Time (partition)} \times \# \text{partitions} = O(n) \times ?

\# \text{Partitions:}

\text{worst-case?} \quad O(n): \text{if pivot is always largest/smallest}

\text{ex: pivot = 19} \quad 13 18 12 16 17 15 19
\[ QS() \]

\text{pivot = 12} \quad 12 18 13 16 17 15
\[ QS() \]

Each partition reduces the list size by \( \frac{1}{2} \), so may need \( n-1 \) partitions so total time = \( O(n) \cdot n = O(n^2) \)

To avoid: choose pivot randomly
Or with Median-of-3 (median of first, middle, and last element)

This choice of pivot leads to...

\text{average-case: pivot chosen such that each partition splits list roughly in half} \implies O(\log n) \text{ partitions, each takes } O(n). \text{ total: } O(n \log n)
Consider recurrence tree:

```
  Time
   \[ \frac{c_n}{2} \quad \frac{c_n}{2} \quad \frac{c_n}{2} \quad \frac{c_n}{2} \quad c_n \] \[ \frac{c_n}{4} \quad \frac{c_n}{4} \quad \frac{c_n}{4} \quad \frac{c_n}{4} \quad c_n \]
```

# levels \( = O(\log n) \)

Total: \( O(n\log n) \).

Anything useful here for Selection Problem?

**Note:** After partition, all elements:
- left of pivot are \( < \) pivot
- right " " " " \( > \) pivot

Can find \( k^{th} \) smallest by considering size of \( L \) (or \( G \))

**Specifically:**
- if \( |L| = k-1 \), \( k-1 \) elements are \( < \) pivot
  - so \( k^{th} \) element is pivot.

**ex:** \( k = 4 \)

```
  \[ |L| = 3 \quad \rightarrow \quad pivot \]
```
* if $|L| > k-1$, (or $|L| \geq k$) more than $k$ elements are $< \text{pivot}$

$\text{ex: } k = 4$

\[\begin{array}{c}
\text{-} \text{pivot} \\
\text{L} \ni 4
\end{array}\]

$\Rightarrow$ so $k^{\text{th}}$ smallest is in $L$!

* if $|L| < k-1$, fewer than $k-1$ elements are $< \text{pivot}$

$\text{ex: } k = 4$

\[\begin{array}{c}
\text{-} \text{pivot} \\
\text{L} \ni 3
\end{array}\]

$\Rightarrow$ so $k^{\text{th}}$ smallest is in $G'$
Rand-Select (A, k) // Returns k-th smallest number in A
   // A contains distinct numbers.

1. Choose a pivot, p, at random from A.
2. Partition A into 2 sublists L, G:
   \[ L = \{ \text{all elements} < p \} \]
   \[ G = \{ \text{all elements} > p \} \]
3. If \(|L| = k-1\) // k-1 elements are < p
   return p
4. If \(|L| > k-1\) // (|L| ≥ k) L contains k-th smallest
   return Rand-Select(L, k)
5. If \(|L| < k-1\) // (|L| < k-1) G contains k-th smallest
   return Rand-Select(G, k-|L|-1)

k needs to change!
Update k by removing all elements in L + pivot

ex: 19 13 18 12 17 16 15 k = 6, ans = 18

p = 15 12 13 15 19 17 16 18
\[ L \quad G \]

k-1 = 5
|L| = 2 < k-1
"new" k = 6 - 2 - 1 = 3, now recurse on G to find 3rd smallest
Run Time: Why is runtime less than Quicksort?

Now, we recurse only on one list L or S.

In QS, we recurse on both lists.
RunTime vs Quicksort?

Running Time of Rand-Select

- worst-case ~ Quicksort (pivot always largest/smallest)
- partition O(n) times, each partition takes O(n) ⇒ O(n^2)

Average-case:
Intuition: every time we find pivot and partition the list, we reduce the size of the sublist to consider

Runtime (informal)

Suppose \( n = 2^p \) (pivot) + split list in exactly \( \frac{n}{2} \) in each partition

Recurrence: \( T(n) = T\left(\frac{n}{2}\right) + cn \)

Master’s: \( a = 1, b = 2, c = 1 \) ⇒ root dominates
⇒ \( \Theta(n^c) = \Theta(n) \)

Recurrence Tree: \( \frac{cn}{2}, \frac{cn}{4}, \frac{cn}{8}, \ldots \)

\( T(n) = \sum_{i=0}^{\log_2 n} \frac{cn}{2^i} = 2cn \) by geometric series

Runtime (formal) - list may not be split in \( \frac{n}{2} \)

Since algorithm is randomized, need expected run time.

Thm: Rand-Select runs in expected (average) time \( O(n) \)
Review expected value

Interested in the expected (i.e. average) value of a variable - random variable that can take on multiple values

r.v. $X$ takes on possible values $x_1, x_2, x_3, \ldots, x_n$

Expected value of $X = x_1 \cdot Pr(X=x_1) + x_2 \cdot Pr(X=x_2) + \ldots + x_n \cdot Pr(X=x_n)$

$$= \sum_{i=1}^{n} x_i \cdot Pr(X=x_i)$$

$$= E[X]$$

ex: $X =$ roll of 6-sided die

$E[X] = 1 \cdot Pr(\frac{1}{6}) + 2 \cdot Pr(\frac{1}{6}) + \ldots + 6 \cdot Pr(\frac{1}{6}) = 3.5$.

We used to use $T(n)$ to denote absolute run-time, now let $T(n)$ denote average (expected) run-time.

$T(n) =$ expected run-time of Rand-Select on list of size $n$.

What is the randomized event that $T(n)$ depends on?

$\Rightarrow$ How the partition splits the list
$X$: r.v. that represents how list is partitioned. $X$ takes on 2 possible values: lucky, unlucky?

$X$ is:
- "lucky" - if partition splits the list such that $L, G$ both have $\leq 3n/4$ elements
  
ex: $|L| = \frac{n}{2}, |G| = \frac{n}{2}$
  $|L| = \frac{n}{3}, |G| = \frac{2n}{3}$
- "unlucky" - if partition splits the list such that either $L$ or $G$ has $\geq 3n/4$ elements
  
ex: $|L| = \frac{4n}{5}, |G| = \frac{n}{5}$

Now, we can write $T(n)$:

$T(n) = \text{expected run-time of Rand-Select on list of size } n$

= \text{exp. time to partition + exp. time to recurse on } L \text{ or } G.$

$\leq n + T(\text{recurse on } L \text{ or } G)$

Note this is worst-case time (not expected), hence we've replaced $\leq$ with $\leq L, G.$

$\leq n + T(\max (|L|, |G|))$
Let's find \( T(\max(141, 161)) \)

\[
T(\max(141, 161)) = \Pr(X \text{ is lucky}) \cdot (\text{time when } X \text{ lucky}) + \Pr(X \text{ is unlucky}) \cdot (\text{time when } X \text{ unlucky})
\]

\( \Pr(X \text{ lucky}) \)?

\( \Rightarrow \) Depends on where pivot ends up after partition.

Consider list split into 4:

- If pivot ends up in \( i \), "unlucky"
- If pivot ends up in \( j \), "lucky"

\[ \Pr(X \text{ is lucky}) = \Pr(X \text{ is unlucky}) = \frac{1}{2} \]

Back to \( T(n) \):

\[
T(n) \leq n + \frac{1}{2} \cdot (\text{exp time when lucky}) + \frac{1}{2} \cdot (\text{exp time when unlucky})
\]

\[
\leq n + \frac{1}{2} T(\leq \frac{3}{4} n) + \frac{1}{2} T(n-1)
\]

Want to get this into a form to use Master's