Now that we know we can't do better than $\mathcal{O}(n \log n)$
AC with comparison sorts, let's make some. We'll use D&C
to approach the problem. To do so, we need to perform
some amount of sorting and pass of subsorts to
smaller problems. We know one algorithm already which
does the former: Partition. Partition sorts 1 element
and divides the array into "smaller stuff" and
"bigger stuff". If we modify Quick Select, we can
get Quick Sort.

```
QuickSort (A, l, r)
    If l <= r
        Return
    A better
    base case
    but not needed
    \[ \begin{cases} 
        \text{If } r - l \leq 5 \\
        \text{IS}(A[l..r]) 
        \end{cases} \]
    p = Select Pivot (A, l, r)
    p = Partition (A, l, r, p)
    QuickSort (A, l, p - 1)
    QuickSort (A, p + 1, r)
    Return
```

**Ex.**
```
5 2 1 3 4 0 7 6
```

```
0 2 1 3 4 5 7 6
```

**Runtime**

Selecting a good pivot is a random process. A "good" pivot
is one which roughly divides the array in half to maximize
our D&C approach.

<table>
<thead>
<tr>
<th>25%</th>
<th>50%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>bad</td>
<td>good</td>
<td>bad</td>
</tr>
</tbody>
</table>
So good pivots (if selected totally at random) occur with probability 50%. We can improve this at the expense of performing additional calculations. We could technically run QuickSelect (AC \Theta(n)) to always select the very best pivot, but let's set this aside and stick with random pivots. Then for QuickSort's runtime, we get

\[ E(T(n)) \leq P(\text{bad pivot})E(T(n-1)) + P(\text{good pivot})(E(T(\frac{3n}{4})) + E(T(\frac{n}{4}))) + \Theta(n). \]

Here we assume the worst bad pivots and good pivots are selected to upper bound AC (we know \( \Omega(n \log n) \) is our lower bound). For clarity, I'll drop the expectations.

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

\[ \downarrow \quad \text{add work} \quad \downarrow \]

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

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

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

\[ MT \Rightarrow T(n) \in \Theta(n \log_3^2) \]

Uh-oh! We over-estimated. Let's draw a work tree using

\[ T(n) \leq T(\frac{3n}{4}) + T(\frac{n}{4}) + \Theta(n) \]

We'll overestimate work here by assuming the height is equal everywhere.

\[
\begin{align*}
\text{height} & = \frac{\log_n \frac{n}{3}}{
\begin{align*}
T(n) &= \sum_{k=0}^{\log_3 n} n = n \log_3 n \in \Theta(n \log n)
\end{align*}
\end{align*}
\]
Now we'll prove QuickSort is correct via induction.

\( p(n) \) = QuickSort(\( A, l, r \)) sorts \( A \) from \( l \) to \( r \) when \( r-l \leq n \).

When \( r-l \leq 1 \), this is clearly true.

Now assume \( p(n) \).

Let \( A \) be given and suppose \( l \) and \( r \) are such that \( r-l = n+1 \).

We previously proved Partition sorts its pivot and arranges (located at \( p \)) \( A \) into \( A' \) such that everything left of the pivot (down to \( l \)) is less than it and everything right of it (up to \( r \)) is bigger. So if we can sort the left and right sections of \( A' \), \( A' \) is sorted from \( l \) to \( r \).

But QuickSort(\( \overline{A'} \), \( l, p-1 \)) and QuickSort(\( A', p+1, r \)) do precisely this since \( p-1-l \leq n \) and \( r-(p+1) \leq n \). \( \square \)