Quicksort Partitioning

Partitioning trace (array contents before and after each exchange)
Quicksort partitioning

Repeat until i and j pointers cross.
• Scan i from left to right so long as a[i] < a[lo].
• Scan j from right to left so long as a[j] > a[lo].
• Exchange a[i] with a[j].

\[ K \quad R \quad A \quad T \quad E \quad L \quad E \quad P \quad U \quad I \quad M \quad Q \quad C \quad X \quad O \quad S \]

\[ \uparrow \quad \uparrow \quad \uparrow \]

lo i j

stop i scan because a[i] >= a[lo]
Quicksort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as $a[i] < a[lo]$.
- Scan j from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
Quicksort partitioning

**Repeat until** \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
Quicksort partitioning

Repeat until i and j pointers cross.

• Scan i from left to right so long as \( a[i] < a[lo] \).
• Scan j from right to left so long as \( a[j] > a[lo] \).
• Exchange \( a[i] \) with \( a[j] \).

stop j scan and exchange \( a[i] \) with \( a[j] \)
Quicksort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as $a[i] < a[lo]$.
- Scan j from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
QuickSort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \( a[i] < a[lo] \).
- Scan j from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
Quicksort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \( a[i] < a[lo] \).
- Scan j from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).

```
K C A T E L E P U I M Q R X O S
```

\( \uparrow \)  \( \uparrow \)  \( \uparrow \)
\( lo \)  \( i \)  \( j \)

stop i scan because \( a[i] \geq a[lo] \)
QuickSort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \( a[i] < a[lo] \).
- Scan j from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
QuickSort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \( a[i] < a[lo] \).
- Scan j from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
Quicksort partitioning

Repeat until \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).

stop \( j \) scan and exchange \( a[i] \) with \( a[j] \)
Quicksort partitioning

Repeat until i and j pointers cross.

• Scan i from left to right so long as \(a[i] < a[lo]\).
• Scan j from right to left so long as \(a[j] > a[lo]\).
• Exchange \(a[i]\) with \(a[j]\).
QuickSort partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as $a[i] < a[lo]$.
- Scan j from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
Quicksort partitioning

Repeat until i and j pointers cross.

• Scan i from left to right so long as a[i] < a[lo].
• Scan j from right to left so long as a[j] > a[lo].
• Exchange a[i] with a[j].

stop i scan because a[i] >= a[lo]
Quicksort partitioning

Repeat until $i$ and $j$ pointers cross.
- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
Quicksort partitioning

 Repeat until i and j pointers cross.

• Scan i from left to right so long as $a[i] < a[lo]$.
• Scan j from right to left so long as $a[j] > a[lo]$.
• Exchange $a[i]$ with $a[j]$.
QuickSort partitioning

Repeat until \(i\) and \(j\) pointers cross.

- Scan \(i\) from left to right so long as \(a[i] < a[lo]\).
- Scan \(j\) from right to left so long as \(a[j] > a[lo]\).
- Exchange \(a[i]\) with \(a[j]\).

\[\text{\texttt{K C A I E L E P U T M Q R X O S}}\]

\[\uparrow \quad \uparrow \quad \uparrow\]

\(\text{lo} \quad i \quad j\)

stop \(j\) scan and exchange \(a[i]\) with \(a[j]\)
Quicksort partitioning

Repeat until $i$ and $j$ pointers cross.

- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
Quicksort partitioning

Repeat until i and j pointers cross.

• Scan i from left to right so long as \( a[i] < a[lo] \).
• Scan j from right to left so long as \( a[j] > a[lo] \).
• Exchange \( a[i] \) with \( a[j] \).

\[ \]

\[ \]

stop i scan because \( a[i] \geq a[lo] \)
Quicksort partitioning

Repeat until i and j pointers cross.

• Scan i from left to right so long as a[i] < a[lo].
• Scan j from right to left so long as a[j] > a[lo].
• Exchange a[i] with a[j].

stop j scan because a[j] <= a[lo]
Quicksort partitioning

Repeat until \( i \) and \( j \) pointers cross.
- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).

When pointers cross.
- Exchange \( a[lo] \) with \( a[j] \).

pointers cross: exchange \( a[lo] \) with \( a[j] \)
Quicksort partitioning

Repeat until i and j pointers cross.
• Scan i from left to right so long as a[i] < a[lo].
• Scan j from right to left so long as a[j] > a[lo].
• Exchange a[i] with a[j].

When pointers cross.
• Exchange a[lo] with a[j].

partitioned!
Dijkstra 3-Way Partitioning
Dijkstra 3-way partitioning

- Let $v$ be partitioning element $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$
Dijkstra 3-way partitioning

- Let $v$ be partitioning element $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$

```
| P | A | B | X | W | P | P | V | P | D | P | C | Y | Z |
```

Invariant

```
<table>
<thead>
<tr>
<th>&lt;V</th>
<th>=V</th>
<th>&gt;V</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>lt</td>
<td>i</td>
<td>gt</td>
</tr>
</tbody>
</table>
```
Dijkstra 3-way partitioning

- Let \( v \) be partitioning element \( a[lo] \).
- Scan \( i \) from left to right.
  - \( (a[i] < v) \): exchange \( a[lt] \) with \( a[i] \) and increment both \( lt \) and \( i \)
  - \( (a[i] > v) \): exchange \( a[gt] \) with \( a[i] \) and decrement \( gt \)
  - \( (a[i] == v) \): increment \( i \)
Dijkstra 3-way partitioning

• Let $v$ be partitioning element $a[lo]$.
• Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$

\[\begin{array}{ccccccccc}
\end{array}\]

\textbf{Invariant}

\[\begin{array}{|c|c|c|}
\hline
< V & = V & \text{gray} & > V \\
\hline
\uparrow & \uparrow & \uparrow \\
lt & i & gt \\
\hline
\end{array}\]
Dijkstra 3-way partitioning

- Let \( v \) be partitioning element \( a[lo] \).
- Scan \( i \) from left to right.
  - \( (a[i] < v) \): exchange \( a[lt] \) with \( a[i] \) and increment both \( lt \) and \( i \)
  - \( (a[i] > v) \): exchange \( a[gt] \) with \( a[i] \) and decrement \( gt \)
  - \( (a[i] == v) \): increment \( i \)

Invariant

- \( <V \)
- \( =V \)
- \( >V \)
- \( lt \)
- \( i \)
- \( gt \)
Dijkstra 3-way partitioning

- Let \( v \) be partitioning element \( a[lo] \).
- Scan \( i \) from left to right.
  - \((a[i] < v)\): exchange \( a[lt] \) with \( a[i] \) and increment both \( lt \) and \( i \)
  - \((a[i] > v)\): exchange \( a[gt] \) with \( a[i] \) and decrement \( gt \)
  - \((a[i] == v)\): increment \( i \)
Dijkstra 3-way partitioning

- Let \( v \) be partitioning element \( a[lo] \).
- Scan \( i \) from left to right.
  - \((a[i] < v)\): exchange \( a[lt] \) with \( a[i] \) and increment both \( lt \) and \( i \)
  - \((a[i] > v)\): exchange \( a[gt] \) with \( a[i] \) and decrement \( gt \)
  - \((a[i] == v)\): increment \( i \)
Let \( v \) be partitioning element \( a[10] \).

Scan \( i \) from left to right.

- \((a[i] < v)\): exchange \( a[lt] \) with \( a[i] \) and increment both \( lt \) and \( i \)
- \((a[i] > v)\): exchange \( a[gt] \) with \( a[i] \) and decrement \( gt \)
- \((a[i] == v)\): increment \( i \)
Dijkstra 3-way partitioning

- Let $v$ be partitioning element $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$

\[
\begin{array}{cccccccccccc}
\end{array}
\]

\[
\begin{array}{ccc}
\text{lt} & i & \text{gt} \\
\downarrow & \downarrow & \downarrow \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{lt} & i & \text{gt} \\
\uparrow & \uparrow & \uparrow \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
< V & = V & \text{gray} & > V \\
\uparrow & \uparrow & \uparrow \\
\text{lt} & i & \text{gt} \\
\end{array}
\]

\text{invariant}
Dijkstra 3-way partitioning

- Let \( v \) be partitioning element \( a[lo] \).
- Scan \( i \) from left to right.
  - \((a[i] < v)\): exchange \( a[lt] \) with \( a[i] \) and increment both \( lt \) and \( i \)
  - \((a[i] > v)\): exchange \( a[gt] \) with \( a[i] \) and decrement \( gt \)
  - \((a[i] == v)\): increment \( i \)

### Dijkstra 3-way partitioning

- \( \text{lt} \)
- \( i \)
- \( \text{gt} \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>V</th>
<th>P</th>
<th>D</th>
<th>W</th>
<th>Y</th>
<th>Z</th>
<th>X</th>
</tr>
</thead>
</table>

**Invariant**

\[<v \quad =v \quad \boxed{\text{partition}} \quad >v\]

\[\downarrow \quad \uparrow \quad \uparrow \quad \downarrow\]

\[\text{lt} \quad i \quad \text{gt}\]
Dijkstra 3-way partitioning

- Let $v$ be partitioning element $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$

**Invariant**

\[
\begin{array}{cccccccccccc}
\lt & i & \gt \\
\end{array}
\]
Dijkstra 3-way partitioning

• Let $v$ be partitioning element $a[lo]$.
• Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$

---

**Invariant**

<table>
<thead>
<tr>
<th>&lt;V</th>
<th>=V</th>
<th>&gt;V</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>lt</td>
<td>i</td>
<td>gt</td>
</tr>
</tbody>
</table>
Dijkstra 3-way partitioning

- Let \( v \) be partitioning element \( a[\text{lo}] \).
- Scan \( i \) from left to right.
  - \((a[i] < v)\): exchange \( a[\text{lt}] \) with \( a[i] \) and increment both \( \text{lt} \) and \( i \)
  - \((a[i] > v)\): exchange \( a[\text{gt}] \) with \( a[i] \) and decrement \( \text{gt} \)
  - \((a[i] == v)\): increment \( i \)
Dijkstra 3-way partitioning

- Let $v$ be partitioning element $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$

invariant

\[
\begin{array}{ccccccc}
\LT & = & \vdots & = & \v \LT & = & \v \LT & = & \v \LT & = & \v \LT & = & \v \\
\LT & = & \v & \LT & = & \v & \LT & = & \v & \LT & = & \v & \LT & = & \v \\
\end{array}
\]
Dijkstra 3-way partitioning

- Let $v$ be partitioning element $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$ and increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$ and decrement $gt$
  - $(a[i] == v)$: increment $i$
Bentley-McIlroy 3-Way Partitioning
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until $i$ and $j$ pointers cross.

- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$. 
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.
- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until $i$ and $j$ pointers cross.

- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$. 

exchange $a[i]$ with $a[j]$
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.
- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.
• Scan i from left to right so long as $a[i] < a[lo]$.
• Scan j from right to left so long as $a[j] > a[lo]$.
• Exchange $a[i]$ with $a[j]$.
• If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
• If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$. 
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as $a[i] < a[lo]$.
- Scan j from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$.

exchange $a[i]$ with $a[j]$
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

• Scan i from left to right so long as \( a[i] < a[lo] \).
• Scan j from right to left so long as \( a[j] > a[lo] \).
• Exchange \( a[i] \) with \( a[j] \).
• If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
• If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \(a[i] < a[lo]\).
- Scan j from right to left so long as \(a[j] > a[lo]\).
- Exchange \(a[i]\) with \(a[j]\).
- If \(a[i] == a[lo]\), exchange \(a[i]\) with \(a[p]\) and increment \(p\).
- If \(a[j] == a[lo]\), exchange \(a[j]\) with \(a[q]\) and decrement \(q\).
Bentley-McIlroy 3-way partitioning

Repeat until $i$ and $j$ pointers cross.
- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$. 

```plaintext
P P B C A P P V P D W X Y Z
```

$lo$ $i$ $j$ $hi$
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as a[i] < a[lo].
- Scan j from right to left so long as a[j] > a[lo].
- Exchange a[i] with a[j].
- If a[i] == a[lo], exchange a[i] with a[p] and increment p.
- If a[j] == a[lo], exchange a[j] with a[q] and decrement q.

exchange a[i] with a[j]
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as $a[i] < a[lo]$.
- Scan j from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$.

exchange $a[j]$ with $a[q]$ and decrement $q$
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \( a[i] < a[lo] \).
- Scan j from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.
- Scan i from left to right so long as $a[i] < a[lo]$.
- Scan j from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$. 

![Diagram of partitioning process]

Variables:
- $p$ and $q$ for pointers
- $lo$, $i$, $j$, and $hi$ for partitioning boundaries
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until $i$ and $j$ pointers cross.

- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$.

exchange $a[i]$ with $a[p]$ and increment $p$
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.
- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until $i$ and $j$ pointers cross.
- Scan $i$ from left to right so long as $a[i] < a[lo]$.
- Scan $j$ from right to left so long as $a[j] > a[lo]$.
- Exchange $a[i]$ with $a[j]$.
- If $a[i] == a[lo]$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $a[j] == a[lo]$, exchange $a[j]$ with $a[q]$ and decrement $q$. 
Bentley-McIlroy 3-way partitioning

Repeat until i and j pointers cross.

- Scan i from left to right so long as \( a[i] < a[lo] \).
- Scan j from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Repeat until \( i \) and \( j \) pointers cross.

- Scan \( i \) from left to right so long as \( a[i] < a[lo] \).
- Scan \( j \) from right to left so long as \( a[j] > a[lo] \).
- Exchange \( a[i] \) with \( a[j] \).
- If \( a[i] == a[lo] \), exchange \( a[i] \) with \( a[p] \) and increment \( p \).
- If \( a[j] == a[lo] \), exchange \( a[j] \) with \( a[q] \) and decrement \( q \).
Bentley-McIlroy 3-way partitioning

Afterwards, swap equal keys to the center.

- Scan \(j\) and \(p\) from right to left and exchange \(a[j]\) with \(a[p]\).
- Scan \(i\) and \(q\) from left to right and exchange \(a[i]\) with \(a[q]\).
Bentley-McIlroy 3-way partitioning

Afterwards, swap equal keys to the center.

- Scan $j$ and $p$ from right to left and exchange $a[j]$ with $a[p]$.
- Scan $i$ and $q$ from left to right and exchange $a[i]$ with $a[q]$.

exchange $a[j]$ with $a[p]$
Bentley-McIlroy 3-way partitioning

Afterwards, swap equal keys to the center.

- Scan $j$ and $p$ from right to left and exchange $a[j]$ with $a[p]$.
- Scan $i$ and $q$ from left to right and exchange $a[i]$ with $a[q]$.
Bentley-McIlroy 3-way partitioning

Afterwards, swap equal keys to the center.

- Scan \( j \) and \( p \) from right to left and exchange \( a[j] \) with \( a[p] \).
- Scan \( i \) and \( q \) from left to right and exchange \( a[i] \) with \( a[q] \).
Bentley-McIlroy 3-way partitioning

Afterwards, swap equal keys to the center.

- Scan \( j \) and \( p \) from right to left and exchange \( a[j] \) with \( a[p] \).
- Scan \( i \) and \( q \) from left to right and exchange \( a[i] \) with \( a[q] \).

exchange \( a[i] \) with \( a[q] \)
Bentley-McIlroy 3-way partitioning

Afterwards, swap equal keys to the center.

• Scan \( j \) and \( p \) from right to left and exchange \( a[j] \) with \( a[p] \).
• Scan \( i \) and \( q \) from left to right and exchange \( a[i] \) with \( a[q] \).