2.3 Partitioning Demos

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- Dual-pivot partitioning
2.3 Partitioning Demos

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
Quicksort partitioning demo

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 \(a[j]\).

\[
\begin{array}{cccccccccccccc}
K & R & A & T & E & L & E & P & U & I & M & Q & C & X & O & S \\
\uparrow & \uparrow & \uparrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
lo & i & j \\
\end{array}
\]

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

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 demo

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 \(a[j]\).

\[
\begin{array}{cccccccccccccccccccc}
K & C & A & I & E & E & L & P & U & T & M & Q & R & X & O & S \\
\uparrow & & & \uparrow & \uparrow & \uparrow & l_o & i & j
\end{array}
\]
Quicksort partitioning demo

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 demo

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 demo

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 \(a[j]\).
Quicksort partitioning demo

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!
2.3 Partitioning Demos

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
Dijkstra 3-way partitioning demo

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

- Let v be partitioning item $a[10]$.
- Scan i from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - $(a[i] == v)$: increment $i$
Dijkstra 3-way partitioning demo

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

- Let \( v \) be partitioning item \( a[10] \).
- Scan \( i \) from left to right.
  - \((a[i] < v)\): exchange \( a[lt] \) with \( a[i] \); increment both \( lt \) and \( i \)
  - \((a[i] > v)\): exchange \( a[gt] \) with \( a[i] \); decrement \( gt \)
  - \((a[i] == v)\): increment \( i \)
Dijkstra 3-way partitioning demo

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

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

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

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

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

- Let $v$ be partitioning item $a[l0]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$, decrement $gt$
  - $(a[i] == v)$: increment $i$

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

- less
- equal
- unknown
- greater
Let $v$ be partitioning item $a[lo]$.

Scan $i$ from left to right.

- $(a[i] < v)$: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
- $(a[i] > v)$: exchange $a[gt]$ with $a[i]$; decrement $gt$
- $(a[i] == v)$: increment $i$

### Dijkstra 3-way partitioning demo

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>P</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>
<tbody>
<tr>
<td>less</td>
<td>equal</td>
<td>unknown</td>
<td>greater</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $lt$: left
- $i$: index
- $gt$: greater
Let v be partitioning item a[lo].

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

- Let $v$ be partitioning item $a[10]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[\text{lt}]$ with $a[i]$; increment both $\text{lt}$ and $i$
  - $(a[i] > v)$: exchange $a[\text{gt}]$ with $a[i]$; decrement $\text{gt}$
  - $(a[i] == v)$: increment $i$
Let $v$ be partitioning item $a[10]$.
Scan $i$ from left to right.
- $(a[i] < v)$: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
- $(a[i] > v)$: exchange $a[gt]$ with $a[i]$; decrement $gt$
- $(a[i] == v)$: increment $i$

Dijkstra 3-way partitioning demo

\[\begin{array}{cccccccccccc}
\hline
less & equal & greater & unknown
\end{array}\]
Let $v$ be partitioning item $a[10]$.

Scan $i$ from left to right.

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

- Let $v$ be partitioning item $a[lo]$.
- Scan $i$ from left to right.
  - $(a[i] < v)$: exchange $a[lit]$ with $a[i]$; increment both $lit$ and $i$
  - $(a[i] > v)$: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - $(a[i] == v)$: increment $i$
2.3 Partitioning Demos

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- Dual-pivot partitioning
Phase I. 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 demo

**Phase I.** 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 demo

**Phase I.** 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 \( q \).
Phase I. 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 demo

**Phase I.** 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 \( q \).

\[
\begin{array}{cccccccccccccccc}
& & & & & p & & & & & & & & & & q \\
& & & & \downarrow & & & & \downarrow \\
\uparrow & & & & \uparrow & & & & \uparrow \\
lo & & & & i & & & & j & & & & hi \\
\end{array}
\]

exchange \( a[i] \) with \( a[j] \)
Phase I. 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 demo

**Phase I.** 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.
Phase I. 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.
Phase I. 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
Phase I. 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 the Bentley–McIlroy 3-way partitioning demo]

- $P$ and $P$ markers indicate the partitioning points.
- $i$ and $j$ are the pointers moving towards each other.
- $lo$, $hi$, $p$, and $q$ are the control variables.
Bentley–McIlroy 3-way partitioning demo

**Phase I.** 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 \( q \).
**Phase I.** 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:**

```
<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>P</td>
<td>P</td>
<td>V</td>
<td>P</td>
<td>D</td>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

*exchange \( a[i] \) with \( a[j] \)*
Bentley–McIlroy 3-way partitioning demo

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

- Scan i from left to right so long as \(a[i] < a[\text{lo}]\).
- Scan j from right to left so long as \(a[j] > a[\text{lo}]\).
- Exchange \(a[i]\) with \(a[j]\).
- If \((a[i] == a[\text{lo}])\), exchange \(a[i]\) with \(a[p]\) and increment \(p\).
- If \((a[j] == a[\text{lo}])\), exchange \(a[j]\) with \(a[q]\) and decrement \(q\).
Phase I. 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 demo

**Phase 1.** 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.
Phase I. 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 demo

Phase I. 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 demo

Phase 1. 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 \(q\).
Phase II. 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 $a[p]$
Phase II. 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 $a[p]$
Phase II. 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 \( a[q] \).
Phase II. 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 \( a[q] \)
Phase II. 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 \( a[q] \).
Phase II. 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 $a[q]$.
2.3 Partitioning Demos

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
Dual-pivot partitioning demo

Initialization.

- Choose $a[lo]$ and $a[hi]$ as partitioning items.
- Exchange if necessary to ensure $a[lo] \leq a[hi]$. 

exchange $a[lo]$ and $a[hi]$
Initialization.

- Choose $a[lo]$ and $a[hi]$ as partitioning items.
- Exchange if necessary to ensure $a[lo] \leq a[hi]$. 
Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.
- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[lt]\) and increment \(lt\) and \(i\).
- Else if \((a[i] > a[hi])\), exchange \(a[i]\) with \(a[gt]\) and decrement \(gt\).
- Else, increment \(i\).

\[
\begin{array}{cccccc}
\text{p}_1 & \text{< p}_1 & \text{p}_1 \leq \text{and} \leq \text{p}_2 & \text{?} & \text{> p}_2 & \text{p}_2 \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\text{lo} & \text{lt} & \text{i} & \text{gt} & \text{hi} & \\
\end{array}
\]

\[
\begin{array}{cccccccccccccccc}
\text{K} & \text{E} & \text{A} & \text{Y} & \text{R} & \text{L} & \text{F} & \text{V} & \text{Z} & \text{Q} & \text{T} & \text{C} & \text{M} & \text{S} \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & & \\
\text{lo} & \text{lt} & \text{i} & & & & & & & & & & \text{gt} & \text{hi} \\
\end{array}
\]

exchange \(a[i]\) and \(a[lt]\); increment \(lt\) and \(i\)
Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.
- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[lt]\) and increment \(lt\) and \(i\).
- Else if \((a[i] > a[hi])\), exchange \(a[i]\) with \(a[gt]\) and decrement \(gt\).
- Else, increment \(i\).

```
exchange a[i] and a[lt]; increment lt and i
```
Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[lt]\) and increment \(lt\) and \(i\).
- Else if \((a[i] > a[hi])\), exchange \(a[i]\) with \(a[gt]\) and decrement \(gt\).
- Else, increment \(i\).

\[
\begin{array}{ccccccc}
\text{p}_1 & \text{< p}_1 & \text{p}_1 \leq \text{and} \leq \text{p}_2 & \text{?} & \text{> p}_2 & \text{p}_2 \\
\uparrow \downarrow \text{lo} & \uparrow \downarrow \text{lt} & \uparrow \downarrow \text{i} & \uparrow \downarrow \text{gt} & \uparrow \downarrow \text{hi} \\
\end{array}
\]

exchange \(a[i]\) and \(a[gt]\); decrement \(gt\)
**Dual-pivot partitioning demo**

**Main loop.** Repeat until $i$ and $gt$ pointers cross.
- If $(a[i] < a[lo])$, exchange $a[i]$ with $a[lt]$ and increment $lt$ and $i$.
- Else if $(a[i] > a[hi])$, exchange $a[i]$ with $a[gt]$ and decrement $gt$.
- Else, increment $i$.
Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[lt]\) and increment \(lt\) and \(i\).
- Else if \((a[i] > a[hi])\), exchange \(a[i]\) with \(a[gt]\) and decrement \(gt\).
- Else, increment \(i\).

<table>
<thead>
<tr>
<th>(p_1)</th>
<th>&lt; (p_1)</th>
<th>(p_1 \leq) and (\leq) (p_2)</th>
<th>?</th>
<th>&gt; (p_2)</th>
<th>(p_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↓</td>
<td>↑</td>
<td>←</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>lo</td>
<td>lt</td>
<td>i</td>
<td></td>
<td>gt</td>
<td>hi</td>
</tr>
</tbody>
</table>

increment \(i\)
Main loop. Repeat until i and gt pointers cross.

- If (a[i] < a[lo]), exchange a[i] with a[lt] and increment lt and i.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.
Main loop. Repeat until $i$ and $gt$ pointers cross.

- If $(a[i] < a[lo])$, exchange $a[i]$ with $a[lt]$ and increment $lt$ and $i$.
- Else if $(a[i] > a[hi])$, exchange $a[i]$ with $a[gt]$ and decrement $gt$.
- Else, increment $i$.

```
\begin{array}{|c|c|c|c|c|c|c|}
\hline
p_1 & < p_1 & p_1 \leq \text{ and } \leq p_2 & ? & > p_2 & p_2 \\
\hline
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
lo & lt & i & gt & hi \\
\hline
\end{array}
```

exchange $a[i]$ and $a[lt]$; increment $lt$ and $i$
Dual-pivot partitioning demo

**Main loop.** Repeat until i and gt pointers cross.
- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[lt]\) and increment \(lt\) and \(i\).
- Else if \((a[i] > a[hi])\), exchange \(a[i]\) with \(a[gt]\) and decrement \(gt\).
- Else, increment \(i\).

\[
\begin{array}{ccccccc}
\text{p}_1 & < \text{p}_1 & \text{p}_1 \leq \text{and} \leq \text{p}_2 & \text{?} & > \text{p}_2 & \text{p}_2 \\
\hline
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\text{lo} & \text{lt} & \text{i} & \text{gt} & \text{hi} \\
\end{array}
\]

**Example:**

\[
\begin{array}{ccccccccccccccc}
K & E & A & F & R & L & M & V & Z & Q & T & C & Y & S \\
\hline
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\text{lo} & \text{lt} & \text{i} & \text{gt} & \text{hi} \\
\end{array}
\]

exchange \(a[i]\) and \(a[gt]\); decrement \(gt\)
**Main loop.** Repeat until $i$ and $gt$ pointers cross.

- If $(a[i] < a[lo])$, exchange $a[i]$ with $a[lt]$ and increment $lt$ and $i$.
- Else if $(a[i] > a[hi])$, exchange $a[i]$ with $a[gt]$ and decrement $gt$.
- Else, increment $i$.

<table>
<thead>
<tr>
<th>$p_1$</th>
<th>$&lt; p_1$</th>
<th>$p_1 \leq$ and $\leq p_2$</th>
<th>$?$</th>
<th>$&gt; p_2$</th>
<th>$p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑ lo</td>
<td>↑ i</td>
<td>↑</td>
<td>↑ gt</td>
<td>↑ hi</td>
</tr>
</tbody>
</table>

**exchange $a[i]$ and $a[lt]$; increment $lt$ and $i**
Dual-pivot partitioning demo

**Main loop.** Repeat until $i$ and $gt$ pointers cross.
- If $(a[i] < a[lo])$, exchange $a[i]$ with $a[lt]$ and increment $lt$ and $i$.
- Else if $(a[i] > a[hi])$, exchange $a[i]$ with $a[gt]$ and decrement $gt$.
- Else, increment $i$.

<table>
<thead>
<tr>
<th>$p_1$</th>
<th>$&lt; p_1$</th>
<th>$p_1 \leq$ and $\leq p_2$</th>
<th>$?$</th>
<th>$&gt; p_2$</th>
<th>$p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>↑ lo</td>
<td>↑ lt</td>
<td>↑ i</td>
<td>↑ gt</td>
<td>↑ hi</td>
</tr>
</tbody>
</table>

**exchange $a[i]$ and $a[gt]$; decrement $gt$**
Dual-pivot partitioning demo

**Main loop.** Repeat until i and gt pointers cross.
- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[lt]\) and increment \(lt\) and \(i\).
- Else if \((a[i] > a[hi])\), exchange \(a[i]\) with \(a[gt]\) and decrement \(gt\).
- Else, increment \(i\).

<table>
<thead>
<tr>
<th>(p_1)</th>
<th>(&lt; p_1)</th>
<th>(p_1 \leq ) and (\leq p_2)</th>
<th>(?)</th>
<th>(&gt; p_2)</th>
<th>(p_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>lo</td>
<td>lt</td>
<td>i</td>
<td>gt</td>
<td>hi</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

```
K E A F C L M R T Q Z V Y S
↑ lo ↑ lt ↑ i ↑ gt ↑ hi
```

**Notes:**
- Exchange \(a[i]\) and \(a[gt]\); decrement \(gt\).
Dual-pivot partitioning demo

Main loop. Repeat until $i$ and $gt$ pointers cross.
- If $(a[i] < a[lo])$, exchange $a[i]$ with $a[l]t$ and increment $lt$ and $i$.
- Else if $(a[i] > a[hi])$, exchange $a[i]$ with $a[gt]$ and decrement $gt$.
- Else, increment $i$.

<table>
<thead>
<tr>
<th>$p_1$</th>
<th>$&lt; p_1$</th>
<th>$p_1 \leq$ and $\leq p_2$</th>
<th>$\ ? \ $</th>
<th>$&gt; p_2$</th>
<th>$p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
</tr>
<tr>
<td>lo</td>
<td>lt</td>
<td>i</td>
<td>gt</td>
<td>hi</td>
<td></td>
</tr>
</tbody>
</table>

increment $i$
Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.
- If $a[i] < a[lo]$, exchange $a[i]$ with $a[lt]$ and increment $lt$ and $i$.
- Else if $a[i] > a[hi]$, exchange $a[i]$ with $a[gt]$ and decrement $gt$.
- Else, increment $i$.

stop when pointers cross
Finalize.

- Exchange $a[hi]$ with $a[+gt]$.

<table>
<thead>
<tr>
<th>$p_1$</th>
<th>$&lt; p_1$</th>
<th>$p_1 \leq and \leq p_2$</th>
<th>$&gt; p_2$</th>
<th>$p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>hi</td>
</tr>
<tr>
<td>lt</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td></td>
</tr>
<tr>
<td>gt</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K</th>
<th>E</th>
<th>A</th>
<th>F</th>
<th>C</th>
<th>L</th>
<th>M</th>
<th>R</th>
<th>Q</th>
<th>T</th>
<th>Z</th>
<th>V</th>
<th>Y</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>hi</td>
</tr>
</tbody>
</table>
Dual-pivot partitioning demo

**Finalize.**
- Exchange $a[hi]$ with $a[++]gt$.

<table>
<thead>
<tr>
<th></th>
<th>$\lt p_1$</th>
<th>$p_1$</th>
<th>$p_1 \leq \text{ and } \leq p_2$</th>
<th>$p_2$</th>
<th>$\gt p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$lo$</td>
<td>$lt$</td>
<td>$gt$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$hi$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3-way partitioned