2.3 Partitioning Demos

- Hoare 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
2.3 Partitioning Demos

- Hoare 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 \( a[j] \).

\[
\begin{array}{ccccccccccccccc}
\text{K} & \text{R} & \text{A} & \text{T} & \text{E} & \text{L} & \text{E} & \text{P} & \text{U} & \text{I} & \text{M} & \text{Q} & \text{C} & \text{X} & \text{O} & \text{S} \\
\text{lo} & \rightarrow & \rightarrow & i & & & & \text{lo} & \quad & \rightarrow & \rightarrow & j
\end{array}
\]

\text{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]
Repeats 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 $a[i] \geq a[lo]$
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 \( a[j] \).

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

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 \(a[j]\).

stop j scan because \(a[j] \leq a[lo]\)
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].

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

- Hoare 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
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 \( a[l_0] \).
- 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 $a[lo]$.

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 $a[lo]$. 

- 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 $a[l_0]$.
- 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 $a[10]$.
- 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 $i$
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

<table>
<thead>
<tr>
<th>D</th>
<th>B</th>
<th>C</th>
<th>P₁</th>
<th>P₅</th>
<th>P₂</th>
<th>P₃</th>
<th>V</th>
<th>P₄</th>
<th>A</th>
<th>W</th>
<th>Y</th>
<th>Z</th>
<th>X</th>
</tr>
</thead>
</table>

--- less --- | --- equal --- | --- unknown --- | --- greater ---
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 \( i \)

```
D  B  C    P1  P5  P2  P3  V  P4  A  W  Y  Z  X

less  equal  unknown  greater
```
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 $i$
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 \( i \)

\[
\begin{array}{cccccccccccc}
D & B & C & P_1 & P_5 & P_2 & P_3 & A & P_4 & V & W & Y & Z & X \\
\hline
\lt & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
less & equal & unknown & greater
\end{array}
\]
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 $i$
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 \( i \)
2.3 Partitioning Demos

- Hoare 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
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.
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 $q$.

```
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.
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[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 \(q\).
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 \(q\).
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 \( q \).

\[
\begin{array}{cccccccccccccc}
\uparrow lo & \uparrow i & \uparrow & \uparrow j & \uparrow hi
\end{array}
\]

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 \( q \).
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[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 $q$.

```
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.
**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–Mcllroy 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 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 \( a[q] \).
Bentley–McIlroy 3-way partitioning demo

**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 \( a[q] \).
**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

- Hoare 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] ≤ a[hi].
Dual-pivot partitioning demo

**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 \(i\).

<table>
<thead>
<tr>
<th>p1</th>
<th>&lt; p1</th>
<th>p1 ≤ and ≤ p2</th>
<th>?</th>
<th>&gt; p2</th>
<th>p2</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>

- **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 \( i \).

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

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>(\text{?})</th>
<th>(&gt; p_2)</th>
<th>(p_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑ (lo)</td>
<td>↑ (lt)</td>
<td>↑ (i)</td>
<td>↑ (gt)</td>
<td>↑ (hi)</td>
<td></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>p1</th>
<th>&lt; p1</th>
<th>p1 ≤ and ≤ p2</th>
<th>?</th>
<th>&gt; p2</th>
<th>p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>lo</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>hi</td>
</tr>
<tr>
<td>↑</td>
<td>lt</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td></td>
</tr>
<tr>
<td></td>
<td>i</td>
<td></td>
<td>↑</td>
<td>↑</td>
<td></td>
</tr>
</tbody>
</table>
```

\[< p_1] \quad [p_1 \leq \text{ and } p_2 \leq ?] \quad > p_2 \quad p_2

\[\text{increment } i\]
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></th>
<th>&lt; p1</th>
<th>p1 ≤ and ≤ p2</th>
<th></th>
<th>&gt; p2</th>
<th>p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lo</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lt</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gt</td>
<td></td>
<td></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>
```
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>( \uparrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>lo ( \uparrow )</td>
</tr>
<tr>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>lt ( \uparrow )</td>
</tr>
<tr>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>i ( \uparrow )</td>
</tr>
<tr>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>gt ( \uparrow )</td>
</tr>
<tr>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>( \uparrow )</td>
<td>hi ( \uparrow )</td>
</tr>
</tbody>
</table>

**Array:** [K, E, A, M, R, L, F, V, Z, Q, T, C, Y, S]

**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 \(i\).

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

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 \(i\).

<table>
<thead>
<tr>
<th>p₁</th>
<th>&lt; p₁</th>
<th>(p₁ \leq ) and (\leq p₂)</th>
<th>?</th>
<th>&gt; p₂</th>
<th>p₂</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>

```
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 \( i \).

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>p₁</th>
<th>&lt; p₁</th>
<th>p₁ ≤ and ≤ p₂</th>
<th>?</th>
<th>&gt; p₂</th>
<th>p₂</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>

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>p1</th>
<th>&lt; p1</th>
<th>p1 ≤ and ≤ p2</th>
<th>?</th>
<th>&gt; p2</th>
<th>p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>lo</td>
<td>↑</td>
<td>↑</td>
<td></td>
<td>hi</td>
</tr>
<tr>
<td>↑</td>
<td>lt</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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 \(i\).

```
K E A F C L M R Q T Z V Y S
```
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>$&gt;$</th>
<th>$p_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\uparrow$</td>
<td>lo</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
</tr>
<tr>
<td></td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>$\uparrow$</td>
<td>hi</td>
</tr>
</tbody>
</table>

---

stop when pointers cross
**Finalize.**
- Exchange $a[hi]$ with $a[++gt]$.
Dual-pivot partitioning demo

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

### 3-way partitioned