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 \(a[j]\).

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

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 \(a[j]\).
Quick sort 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 \(a[j]\)
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 \(a[j]\).

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

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 \(a[j]\).
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 \( a[j] \).

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

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 \(a[j]\).

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

\text{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 \( a[j] \).

\[
\begin{array}{c}
\text{stop i scan because } a[i] \geq a[lo]
\end{array}
\]
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 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 \( a[j] \).

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 \( i \)

\[
\begin{array}{cccccccccccccc}
P_1 & D & B & X & W & P_2 & P_3 & V & P_4 & A & P_5 & C & Y & Z \\
\end{array}
\]
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 $i$
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 \( a[\text{lt}] \) 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 \( i \)

<table>
<thead>
<tr>
<th>D</th>
<th>B</th>
<th>P_1</th>
<th>C</th>
<th>W</th>
<th>P_2</th>
<th>P_3</th>
<th>V</th>
<th>P_4</th>
<th>A</th>
<th>P_5</th>
<th>Y</th>
<th>Z</th>
<th>X</th>
</tr>
</thead>
</table>

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

less \hspace{1cm} equal \hspace{1cm} unknown \hspace{1cm} greater
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

```
    | D | B | C | P1 | W | P2 | P3 | V | P4 | A | P5 | Y | Z | X |
---|---|---|---|----|---|----|----|---|----|---|----|---|---|---|
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 $i$
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
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 \( i \)
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 \( a[10] \).
- 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 $a[l]$ with $a[i]$; increment both $l$ and $i$
  - ($a[i] > v$): exchange $a[g]$ with $a[i]$; decrement $g$
  - ($a[i] == v$): increment $i$

```
lo        |    |    |    |
          ↓  ↓  ↓  ↓
D B C A   | P₅ | P₄ | P₃ | P₁ |
           |    |    |    |
```

---

<table>
<thead>
<tr>
<th>less</th>
<th>equal</th>
<th>greater</th>
</tr>
</thead>
</table>
2.3 Partitioning Demos

- Sedgewick 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 \( q \).
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.
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.

```plaintext
  lo i j hi
P P B C A P P V P D W X Y Z
```
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[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 $q$. 

**Diagram:**

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

- $lo$ to $i$
- $j$ to $hi$

**Text:**

`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 $q$.

![Diagram of partitioning process]

exchange $a[i]$ with $a[p]$ and increment $p$
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.
**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 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 $a[q]$.
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 $a[q]$.

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 $i$.

```plaintext

<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>$p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</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></th>
<th>&lt; p₁</th>
<th>p₁ ≤ and ≤ p₂</th>
<th>?</th>
<th>&gt; p₂</th>
<th>p₂</th>
</tr>
</thead>
<tbody>
<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>

exchange a[i] and a[lt]; increment lt and 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 $i$.

```plaintext

<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>
</tbody>
</table>

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 \(i\).
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>(\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 \(i\).

<table>
<thead>
<tr>
<th>(p_1)</th>
<th>&lt; (p_1)</th>
<th>(p_1 \leq \text{ 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>

increment \(i\)

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

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

**K E A F R L M C Z Q T V Y S**

<table>
<thead>
<tr>
<th>(\uparrow)</th>
<th>(\uparrow)</th>
<th>(\uparrow)</th>
<th>(\uparrow)</th>
<th>(\uparrow)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(lo)</td>
<td>(lt)</td>
<td>(i)</td>
<td>(gt)</td>
<td>(hi)</td>
</tr>
</tbody>
</table>

**exchange** \(a[i]\) **and** \(a[lt]\); **increment** \(lt\) **and** \(i\)**
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 $i$.

```
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:**

<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>

---

**Diagram:**

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

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></td>
<td>gt</td>
<td>hi</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 $i$.

<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>$j$</td>
<td>$hi$</td>
</tr>
</tbody>
</table>

stop when pointers cross
Dual-pivot partitioning demo

Finalize.

- Exchange $a[hi]$ with $a[+gt]$.
Dual-pivot partitioning demo

Finalize.
- Exchange $a[\text{lo}]$ with $a[\text{--lt}]$.
- Exchange $a[\text{hi}]$ with $a[\text{++gt}]$.

3-way partitioned