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 \( a[j] \).

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 \(a[j]\).

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

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

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

stop 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 \(a[j]\).

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

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 \( a[j] \).

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

- Sedgewick 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 $a[lo]$.
- 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
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 $a[10]$.
- 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 $a[10]$.

- 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 $a[l]$. 
- 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 P V P D W Y Z X

<table>
<thead>
<tr>
<th>less</th>
<th>equal</th>
<th>unknown</th>
<th>greater</th>
</tr>
</thead>
</table>
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 \( \text{lt} \) and \( i \)
  - \((a[i] > v)\): exchange \( a[gt] \) 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 \( i \)
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 $a[lo]$.

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

- 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 \( q \).
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.

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.

```
exchangec a[i]c with c a[p]c and c incrementc 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 \(q\).
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 \( q \).

```
\[ a[i] < a[lo] \]
\[ a[j] > a[lo] \]
```

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 \(q\).
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 \(q\).

![Diagram showing the partitioning process](image-url)
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
```
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 \(q\).

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

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 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 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 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 $a[q]$.

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

  lo  j  i  hi
```

*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 $a[j]$ with $a[p]$
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 $a[q]$.
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 \( a[q] \).
2.3 PARTITIONING DEMOS

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley-McIlroy 3-way partitioning
- dual-pivot partitioning
Initialization.

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

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

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

```
   p1 < p1   p1 ≤ and ≤ p2   ?   > p2   p2
   ↑  lo  ↑  lt  ↑  i  ↑  gt  ↑  hi
```

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

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 \(i\).

```
<table>
<thead>
<tr>
<th>(p_1)</th>
<th>&lt; (p_1)</th>
<th>(p_1) ≤ and ≤ (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 \(i\)
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 \(i\).

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

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

exchange \(a[i]\) and \(a[lt]\); 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 \text{ 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>

**Diagram:**

- 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>$\uparrow$ lo</td>
<td>$\uparrow$ lt</td>
<td>$\uparrow$ i</td>
<td>$\uparrow$ gt</td>
<td>$\uparrow$ hi</td>
</tr>
</tbody>
</table>

**Example:**

- 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 \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></td>
<td>gt</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 \(i\).

\[
\begin{array}{ccccccc}
p_1 & < p_1 & p_1 \leq \text{and} \leq p_2 & ? & > p_2 & p_2 \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
lo & lt & i & gt & hi \\
\end{array}
\]

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 \(i\).
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
Dual-pivot partitioning demo

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>
</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></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></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>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>lo</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>hi</th>
</tr>
</thead>
<tbody>
<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>
<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>
<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>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dual-pivot partitioning demo

Finalize.

- Exchange \( a[lo] \) with \( a[--lt] \).
- Exchange \( a[hi] \) with \( a[++gt] \).

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

3-way partitioned