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 \(a[j]\).

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

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

Repeat until i and j pointers cross.

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

Repeat until i and j pointers cross.

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

Repeat until i and j pointers cross.

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

stop j scan and exchange a[i] with a[j]
Quicksort partitioning demo

Repeat until i and j pointers cross.

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

Repeat until i and j pointers cross.

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

Repeat until i and j pointers cross.

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

\[ \]

\[ \]

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

Repeat until i and j pointers cross.

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

Repeat until i and j pointers cross.

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

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

Repeat until i and j pointers cross.

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

Repeat until i and j pointers cross.

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

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

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

\[ \text{stop j scan because } a[j] \leq a[lo] \]
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].
2.3 Partitioning Demos

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

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[\text{lt}]$ with $a[i]$; increment both $\text{lt}$ and $i$
  - greater: exchange $a[\text{gt}]$ with $a[i]$; decrement $\text{gt}$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Let $v = a[10]$ be pivot.

Scan $i$ from left to right and compare $a[i]$ to $v$.
- less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
- greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
- equal: increment $i$
Dijkstra 3-way partitioning demo

- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

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

- Let \( v = a[10] \) be pivot.
- Scan \( i \) from left to right and compare \( a[i] \) to \( v \).
  - less: exchange \( a[lt] \) with \( a[i] \); increment both \( lt \) and \( i \)
  - greater: exchange \( a[gt] \) with \( a[i] \); decrement \( gt \)
  - equal: increment \( i \)

<table>
<thead>
<tr>
<th>D</th>
<th>B</th>
<th>C</th>
<th>P_1</th>
<th>P_5</th>
<th>P_2</th>
<th>P_3</th>
<th>A</th>
<th>P_4</th>
<th>V</th>
<th>W</th>
<th>Y</th>
<th>Z</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>lt</td>
<td></td>
<td></td>
<td></td>
<td>i</td>
<td></td>
<td>gt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>less</td>
<td>equal</td>
<td>unknown</td>
<td>greater</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let $v = a[10]$ be pivot.

Scan $i$ from left to right and compare $a[i]$ to $v$.
- less: exchange $a[\lt]$ with $a[i]$; increment both $\lt$ and $i$
- greater: exchange $a[\gt]$ with $a[i]$; decrement $\gt$
- equal: increment $i$
Dijkstra 3-way partitioning demo

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
Dijkstra 3-way partitioning demo

- Let $v = a[lo]$ be pivot.
- Scan $i$ from left to right and compare $a[i]$ to $v$.
  - less: exchange $a[lt]$ with $a[i]$; increment both $lt$ and $i$
  - greater: exchange $a[gt]$ with $a[i]$; decrement $gt$
  - equal: increment $i$
2.3 Partitioning Demos

- Hoare 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley–McIlroy 3-way partitioning
- dual-pivot partitioning
Phase I. Repeat until i and j pointers cross.

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

**Phase I.** Repeat until i and j pointers cross.

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

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

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

---

**Diagram:**

- $p$ moves from left to right.
- $q$ moves from right to left.
- $i$, $j$, and $lo$ are pointers moving along the array.
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 \( q \).
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–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 $q$.

---

**Diagram Description:**

- $p$ points to the left.
- $q$ points to the right.
- $lo$ points to the left.
- $hi$ points to the right.
- $i$ points to the left.
- $j$ points to the right.

**Action:**

- Exchange $a[i]$ with $a[j]$. 

---

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

**Notes:**

- 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 $q$. 
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.

```
  P  P  B  C  A  P  P  V  P  D  W  X  Y  Z
  \( \uparrow \)  \( \uparrow \)  \( \uparrow \)  \( \uparrow \)  \( \uparrow \)  \( \uparrow \)  \( \uparrow \)  \( \uparrow \)
  lo  i  j  hi
```
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 \( q \).

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.
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–McIllroy 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 $q$. 

---

<table>
<thead>
<tr>
<th>$P$</th>
<th>$P$</th>
<th>$P$</th>
<th>$C$</th>
<th>$A$</th>
<th>$D$</th>
<th>$B$</th>
<th>$V$</th>
<th>$Y$</th>
<th>$Z$</th>
<th>$W$</th>
<th>$X$</th>
<th>$P$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$lo$</td>
<td>$i$</td>
<td>$j$</td>
<td>$hi$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 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 $a[p]$
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 \( a[q] \).

\[ \text{exchange } a[i] \text{ with } a[q] \]
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 $a[q]$.

3-way partitioned
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 $a[lo] \leq a[hi]$.

exchange $a[lo]$ and $a[hi]$
Dual-pivot partitioning demo

**Initialization.**

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

![Diagram of partitioning process](image)
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>

交换 \( a[i] \) 与 \( a[lt] \); 增加 \( lt \) 和 \( i \)。
**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>(&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>
</tr>
<tr>
<td>(lo)</td>
<td>(lt)</td>
<td>(i)</td>
<td>(gt)</td>
<td>(hi)</td>
</tr>
</tbody>
</table>

**Example:**

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

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

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

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

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>↑ lo</td>
<td>↑ lt</td>
<td>↑ i</td>
<td>↑ gt</td>
<td>↑ hi</td>
</tr>
</tbody>
</table>

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

↑ lo

↑ ↑ lt i

↑ gt

↑ hi

increment $i$
Dual-pivot partitioning demo

**Main loop.** Repeat until i and gt pointers cross.
- If \((a[i] < a[lo])\), exchange \(a[i]\) with \(a[l\ell]\) and increment \(l\ell\) 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>(\downarrow)</td>
<td>(\uparrow)</td>
<td>(\uparrow)</td>
<td>(\downarrow)</td>
<td>(\uparrow)</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 $i$.

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 \( i \).

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

\begin{align*}
exchanging a[i] \text{ and } a[lt] \text{; increment } lt \text{ and } i
\end{align*}
Dual-pivot partitioning demo

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

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

exchange \( a[i] \) and \( a[gt] \); decrement \( gt \)
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 \( i \).

---

**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 \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>(\downarrow lo)</td>
<td>(\downarrow lt)</td>
<td>(\uparrow i)</td>
<td>(\uparrow gt)</td>
<td>(\uparrow 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>$p_1$</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 \(i\).

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

stop when pointers cross
Finalize.

- Exchange \( a[lo] \) with \( a[--lt] \).
- Exchange \( a[hi] \) with \( a[++gt] \).
Dual-pivot partitioning demo

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

3-way partitioned
2.3 PARTITIONING DEMOS

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

For \( i = \text{lo}+1 \) to \( \text{hi} \):
- If \( a[i] < a[\text{lo}] \):
  - increment \( j \)
  - exchange \( a[i] \) with \( a[j] \).
Lomuto partitioning demo

For \( i = \text{lo} + 1 \) to \( \text{hi} \):

- If \( a[i] < a[\text{lo}] \):
  - increment \( j \)
  - exchange \( a[i] \) with \( a[j] \).
Lomuto partitioning demo

For $i = lo + 1$ to $hi$:

- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 

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

↑  ↑  ↑
lo  j  i
```
Lomuto partitioning demo

For $i = lo+1$ to $hi$:
  - If $a[i] < a[lo]$:
    - increment $j$
    - exchange $a[i]$ with $a[j]$.
Lomuto partitioning demo

For $i = lo+1$ to $hi$:
- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$.
Lomuto partitioning demo

For $i = lo+1$ to $hi$:
- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$.
Lomuto partitioning demo

For \( i = \text{lo} + 1 \) to \( \text{hi} \):

- If \( a[i] < a[\text{lo}] \):
  - increment \( j \)
  - exchange \( a[i] \) with \( a[j] \).
Lomuto partitioning demo

For $i = lo+1$ to $hi$:

- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 
Lomuto partitioning demo

For $i = \text{lo+1}$ to hi:

- If $a[i] < a[\text{lo}]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$.
Lomuto partitioning demo

For $i = lo+1$ to $hi$:

- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 

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

\[
\uparrow \quad \uparrow \quad \uparrow \\
lo \quad j \quad i
\]
Lomuto partitioning demo

For $i = lo+1$ to $hi$:
- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 
Lomuto partitioning demo

For $i = lo+1$ to $hi$:
- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 

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

  lo   j   i
```
Lomuto partitioning demo

For $i = \text{lo}+1$ to $\text{hi}$:
- If $a[i] < a[\text{lo}]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 
Lomuto partitioning demo

For \( i = \text{lo} + 1 \) to \( \text{hi} \):

- If \( a[i] < a[\text{lo}] \):
  - increment \( j \)
  - exchange \( a[i] \) with \( a[j] \).
Lomuto partitioning demo

For $i = lo+1$ to $hi$:

- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 

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

↑ lo  ↑ j  ↑ i
```
Lomuto partitioning demo

For \( i = 1o+1 \) to \( hi \):
- If \( a[i] < a[1o] \):
  - increment \( j \)
  - exchange \( a[i] \) with \( a[j] \).
Lomuto partitioning demo

For $i = lo+1$ to $hi$:

- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 
Lomuto partitioning demo

For $i = \text{lo}+1$ to $\text{hi}$:

- If $a[i] < a[\text{lo}]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 
Lomuto partitioning demo

For $i = lo+1$ to $hi$:

- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$. 
Lomuto partitioning demo

For i = lo+1 to hi:

- If a[i] < a[lo]:
  - increment j
  - exchange a[i] with a[j].
Lomuto partitioning demo

For $i = lo+1$ to $hi$:
- If $a[i] < a[lo]$:
  - increment $j$
  - exchange $a[i]$ with $a[j]$.

Lomuto partitioning demo

For \( i = \text{lo} + 1 \) to \( \text{hi} \):

- If \( a[i] < a[\text{lo}] \):
  - increment \( j \)
  - exchange \( a[i] \) with \( a[j] \).

Exchange \( a[\text{lo}] \) with \( a[j] \).