



<http://algs4.cs.princeton.edu>

2.3 PARTITIONING DEMOS

- ▶ *Sedgwick 2-way partitioning*
- ▶ *Dijkstra 3-way partitioning*
- ▶ *Bentley-McIlroy 3-way partitioning*
- ▶ *dual-pivot partitioning*



<http://algs4.cs.princeton.edu>

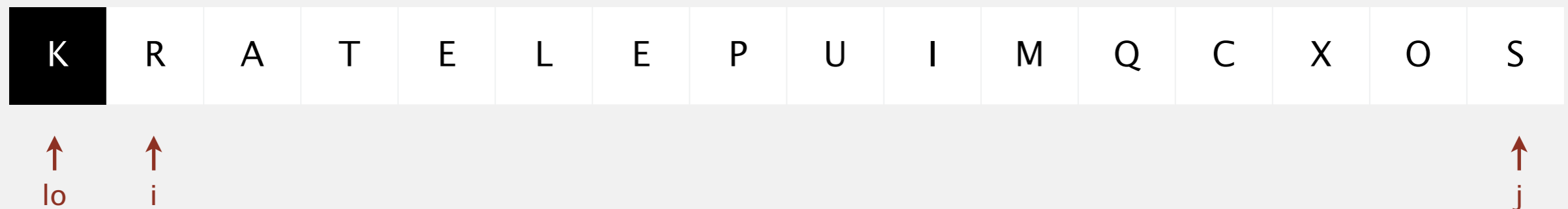
2.3 PARTITIONING DEMOS

- ▶ *Sedgwick 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 $a[j]$.

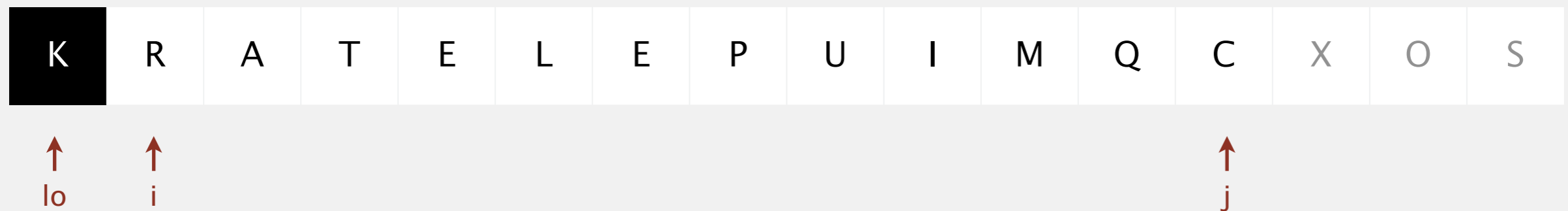


stop i scan because $a[i] \geq a[lo]$

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 $a[j]$.

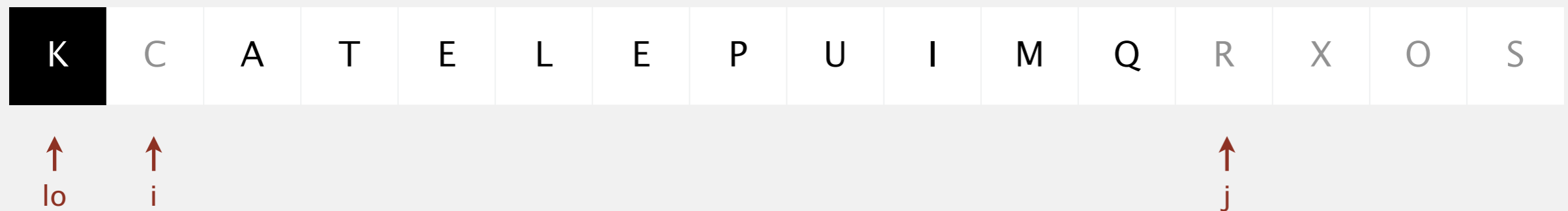


stop j scan and exchange $a[i]$ with $a[j]$

Quicksort partitioning demo

Repeat until i and j pointers cross.

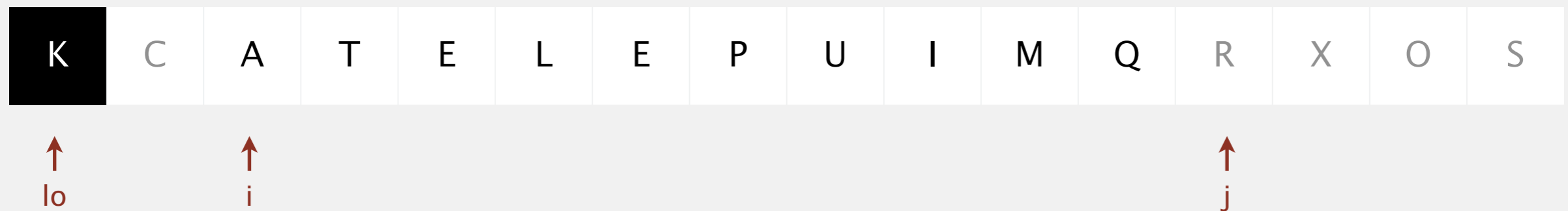
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



Quicksort partitioning demo

Repeat until i and j pointers cross.

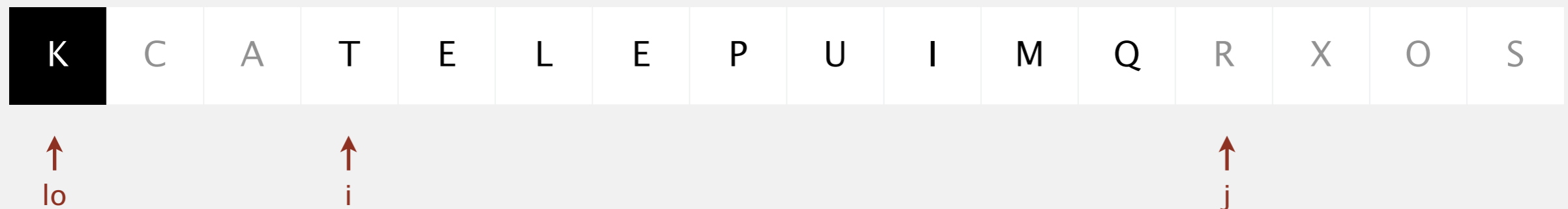
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



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 $a[j]$.

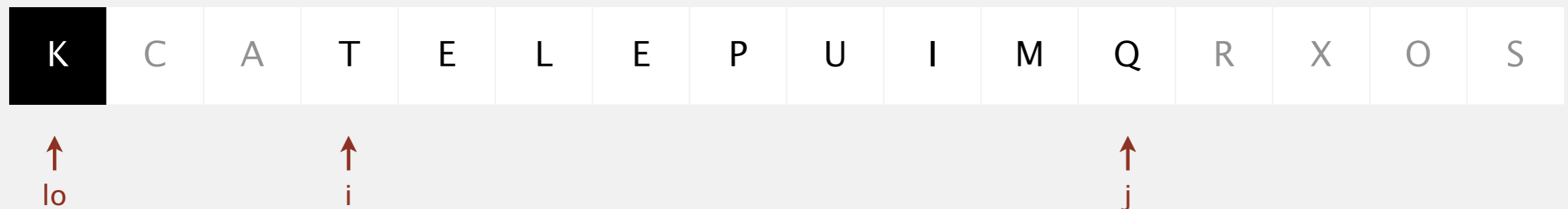


stop i scan because $a[i] \geq a[lo]$

Quicksort partitioning demo

Repeat until i and j pointers cross.

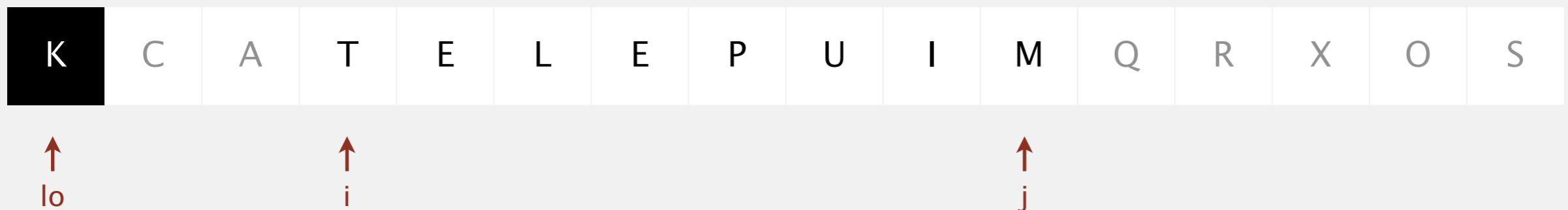
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



Quicksort partitioning demo

Repeat until i and j pointers cross.

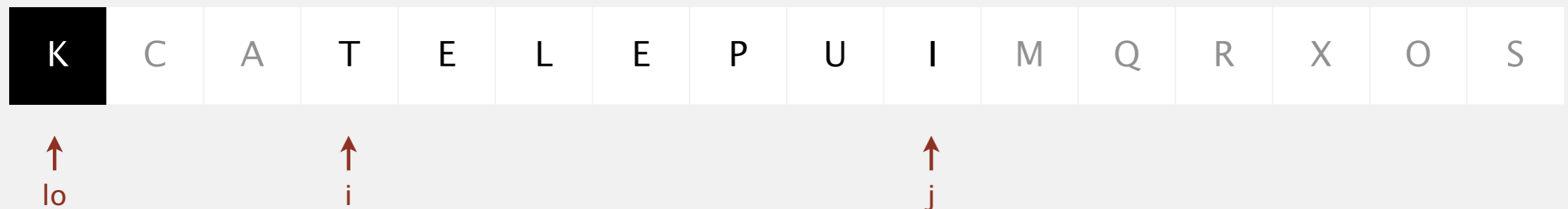
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



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 $a[j]$.

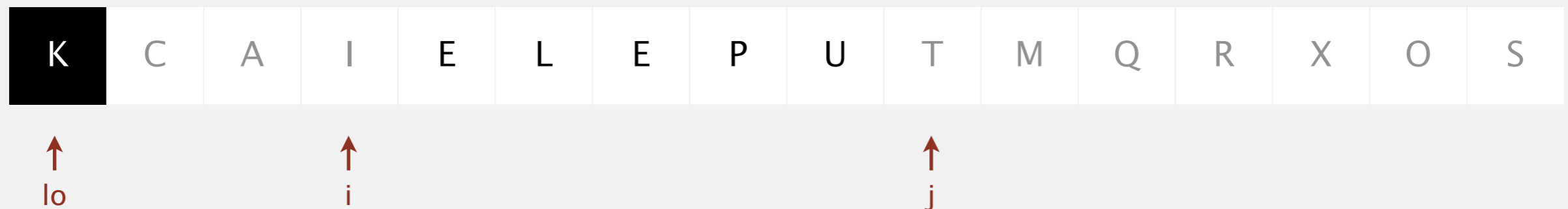


stop j scan and exchange $a[i]$ with $a[j]$

Quicksort partitioning demo

Repeat until i and j pointers cross.

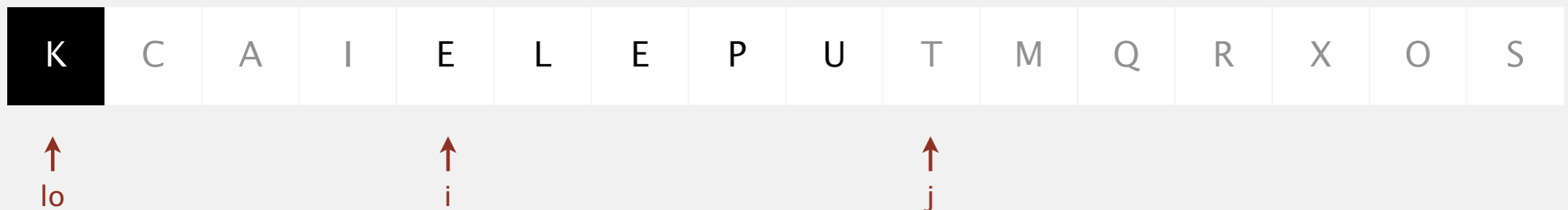
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



Quicksort partitioning demo

Repeat until i and j pointers cross.

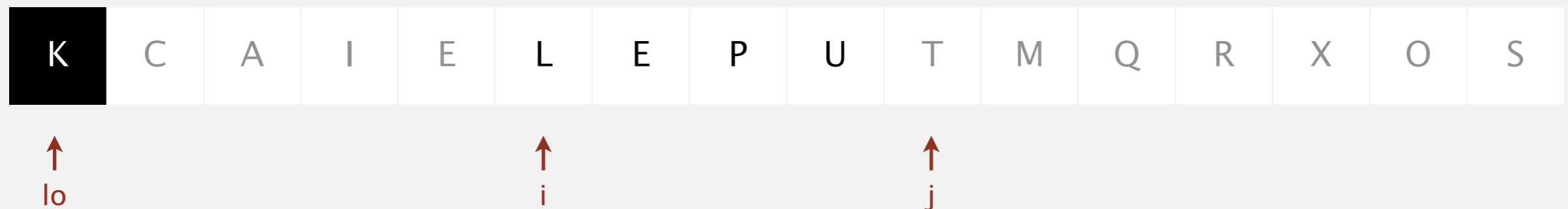
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



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.

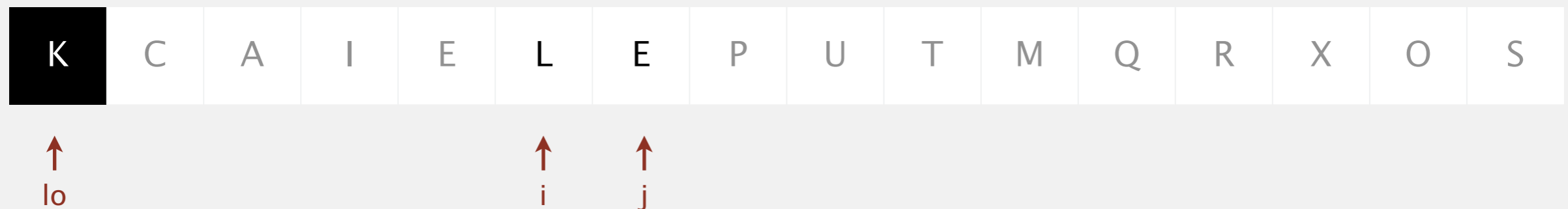
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.



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 $a[j]$.

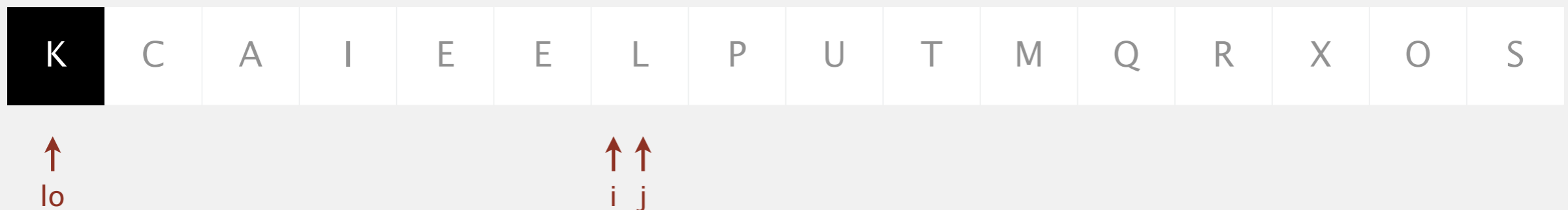


stop j scan and exchange $a[i]$ with $a[j]$

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 $a[j]$.

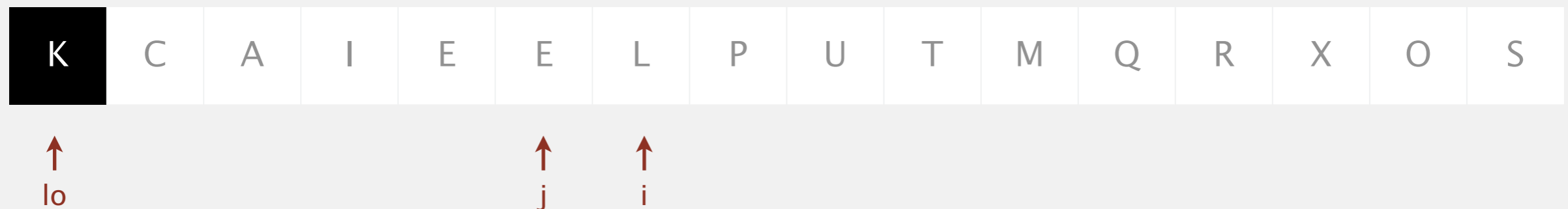


stop i scan because $a[i] \geq a[lo]$

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 $a[j] \leq a[lo]$

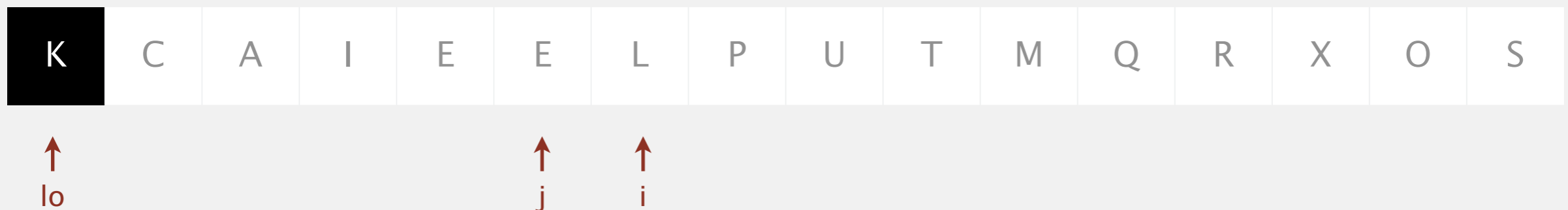
Quicksort partitioning demo

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 $a[j]$

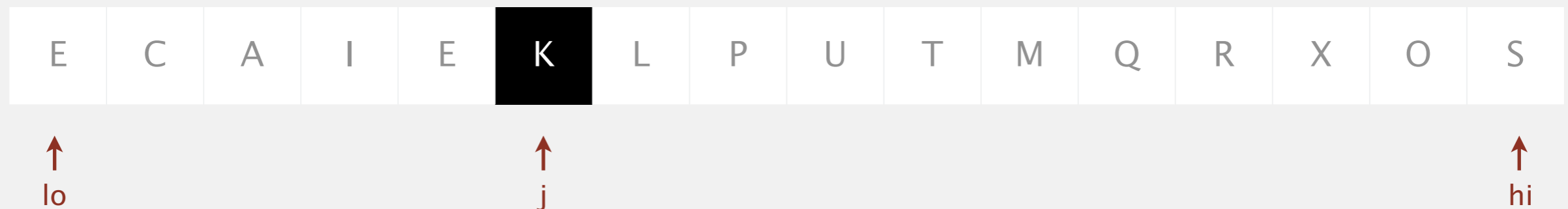
Quicksort partitioning demo

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!



<http://algs4.cs.princeton.edu>

2.3 PARTITIONING DEMOS

- ▶ *Sedgwick 2-way partitioning*
- ▶ *Dijkstra 3-way partitioning*
- ▶ *Bentley-McIlroy 3-way partitioning*
- ▶ *dual-pivot partitioning*

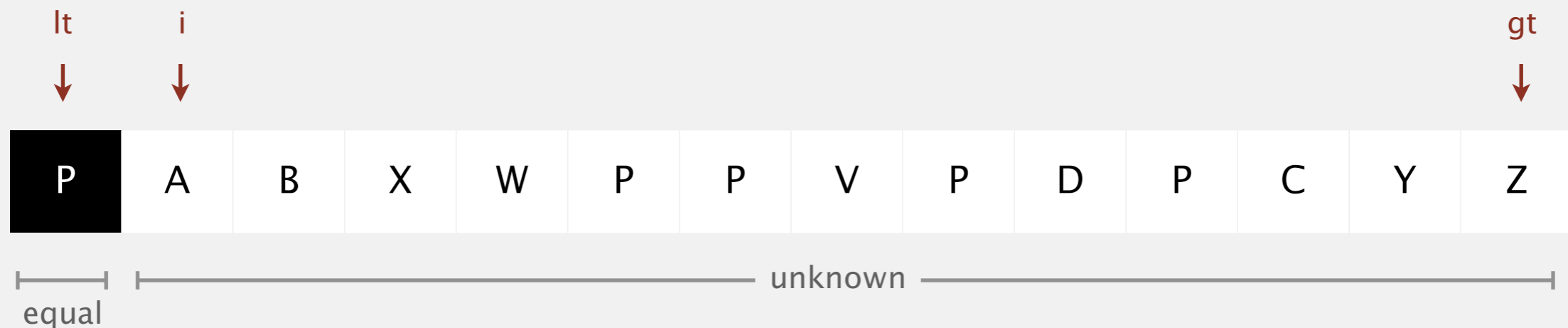
Dijkstra 3-way partitioning demo

- 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



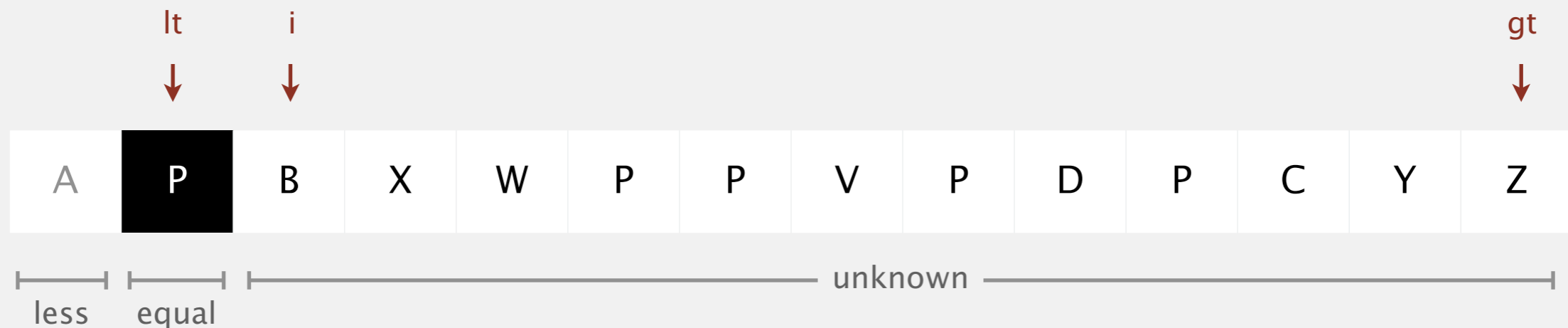
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



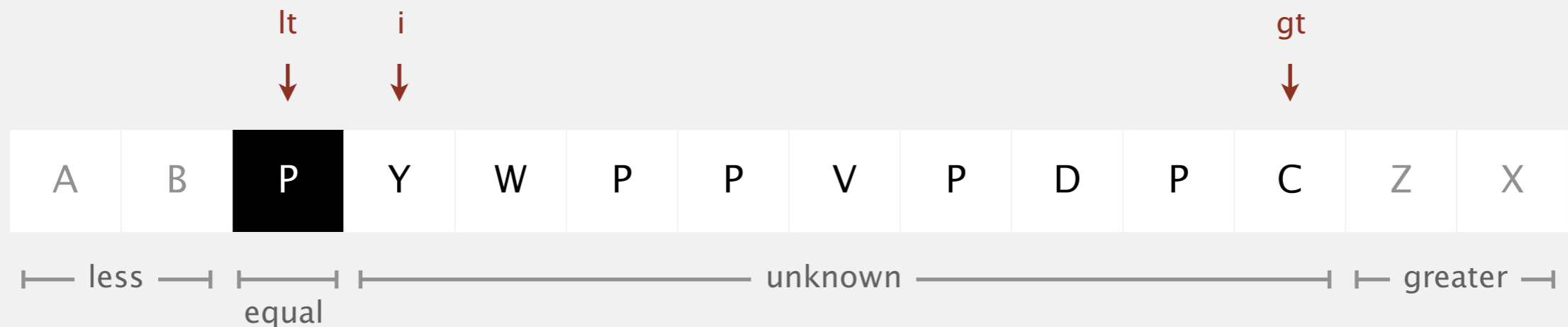
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



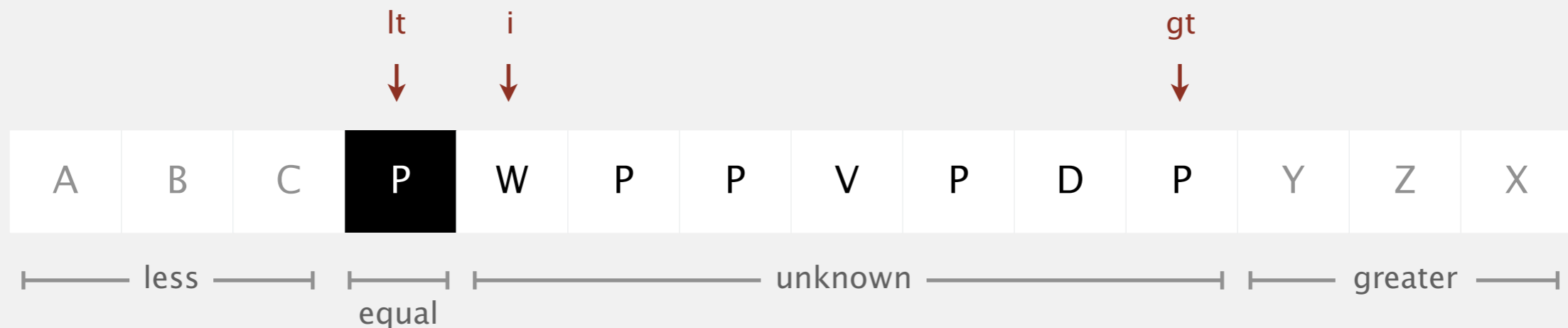
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



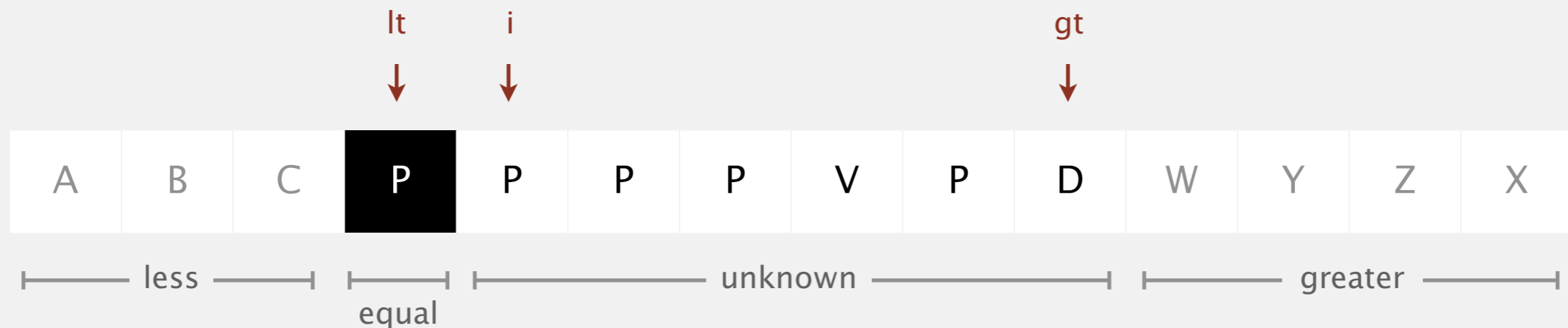
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



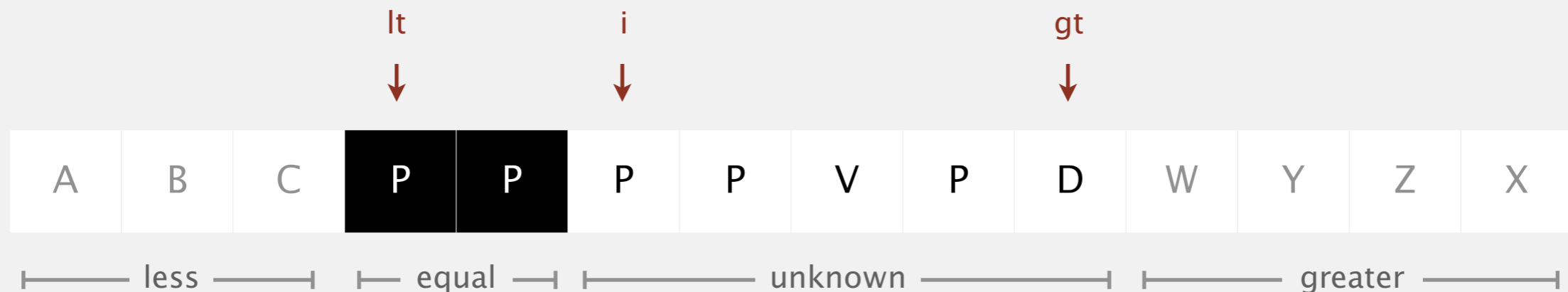
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



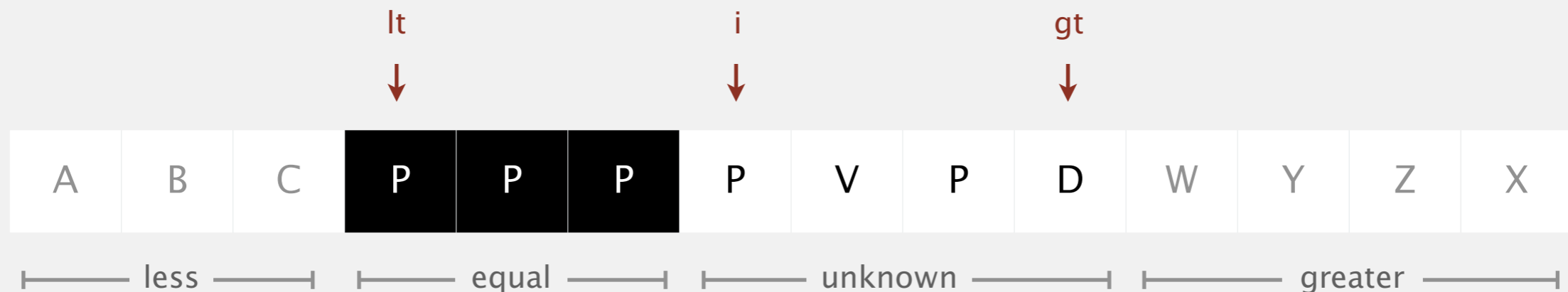
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



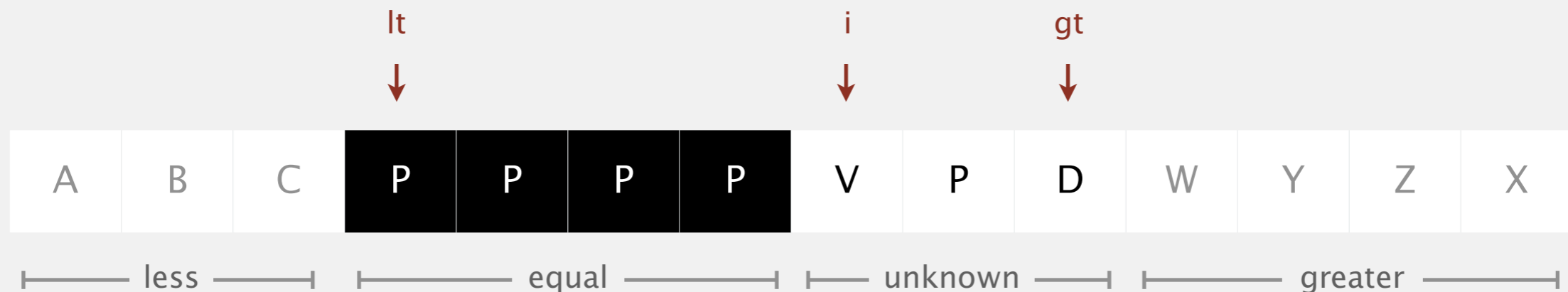
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



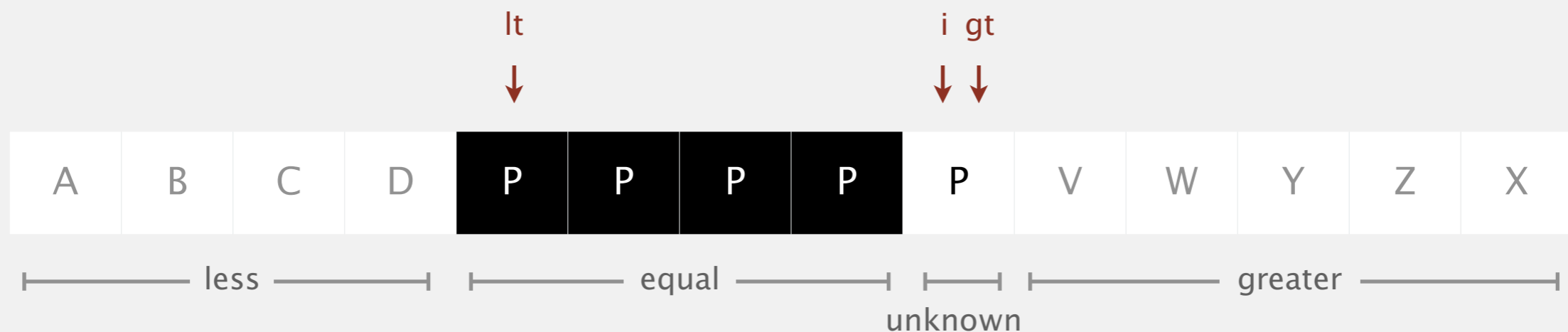
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



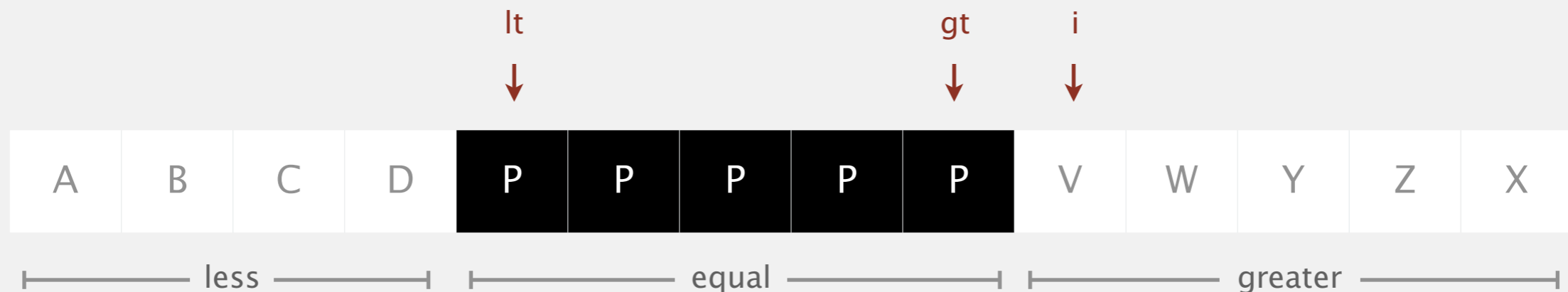
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t and i
 - ($a[i] > v$): exchange $a[gt]$ with $a[i]$; decrement gt
 - ($a[i] == v$): increment i



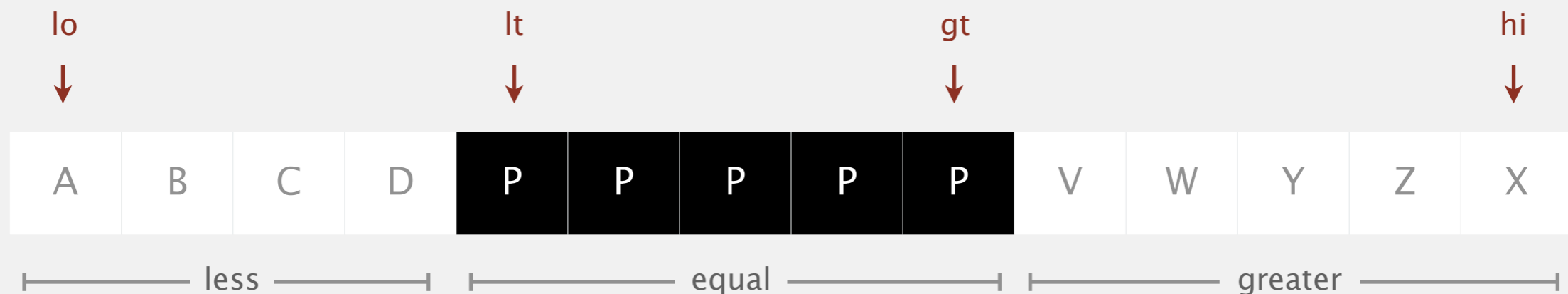
Dijkstra 3-way partitioning demo

- Let v be partitioning item $a[l_0]$.
- Scan i from left to right.
 - ($a[i] < v$): exchange $a[l_t]$ with $a[i]$; increment both l_t 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





<http://algs4.cs.princeton.edu>

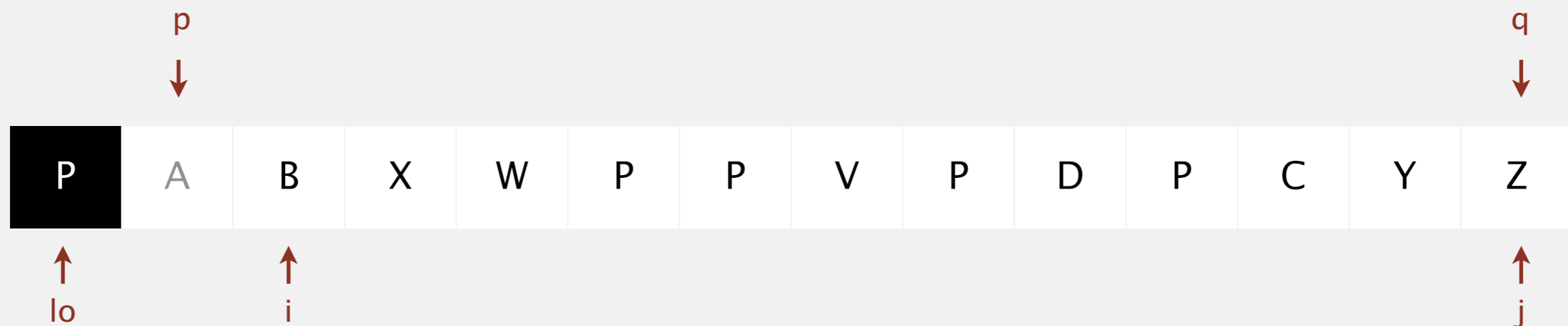
2.3 PARTITIONING DEMOS

- ▶ *Sedgwick 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.

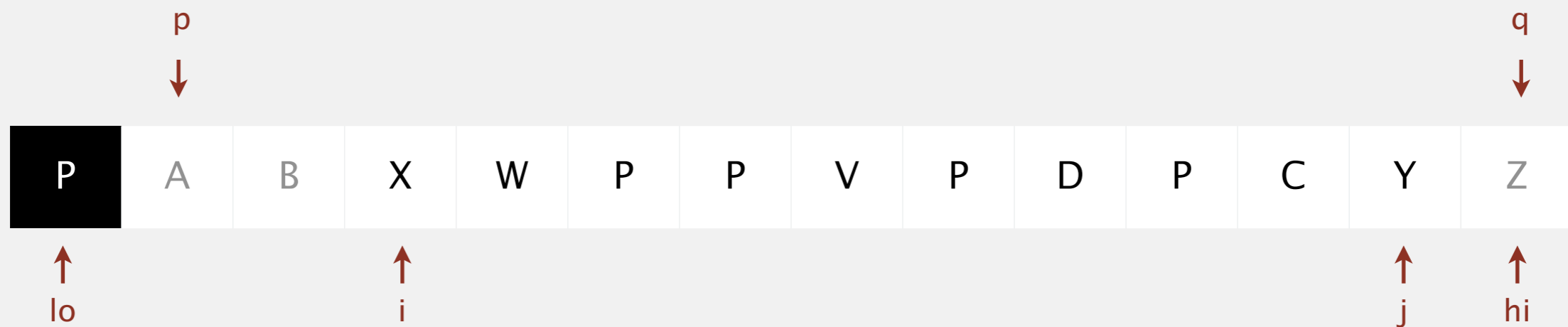
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

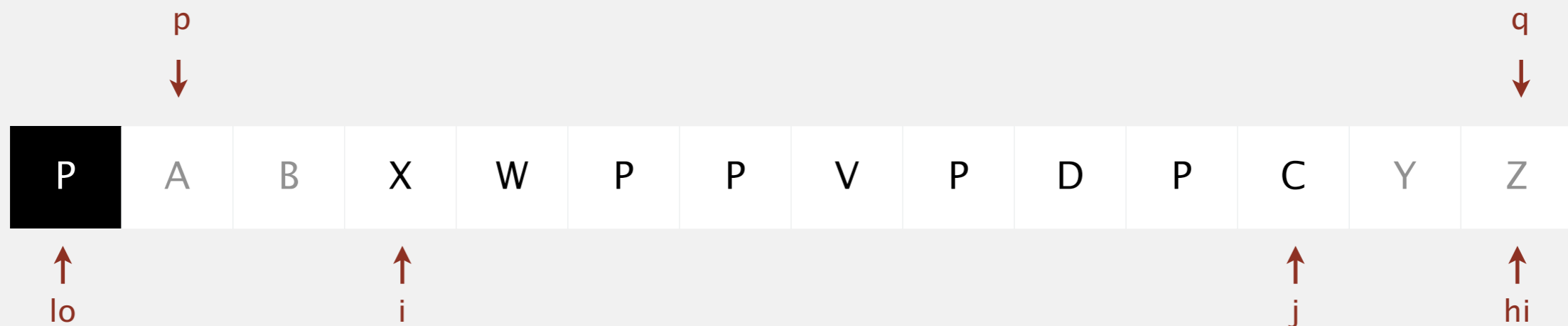
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

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

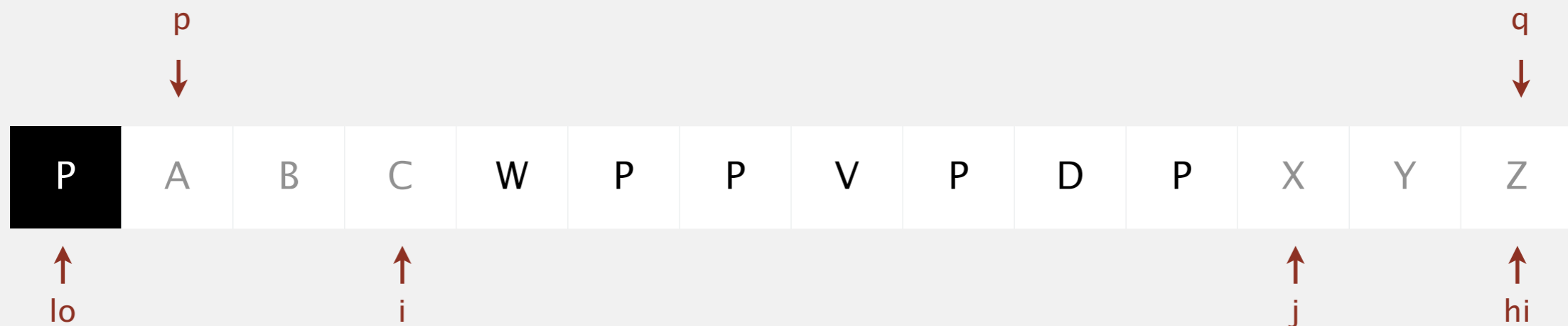


exchange $a[i]$ with $a[j]$

Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

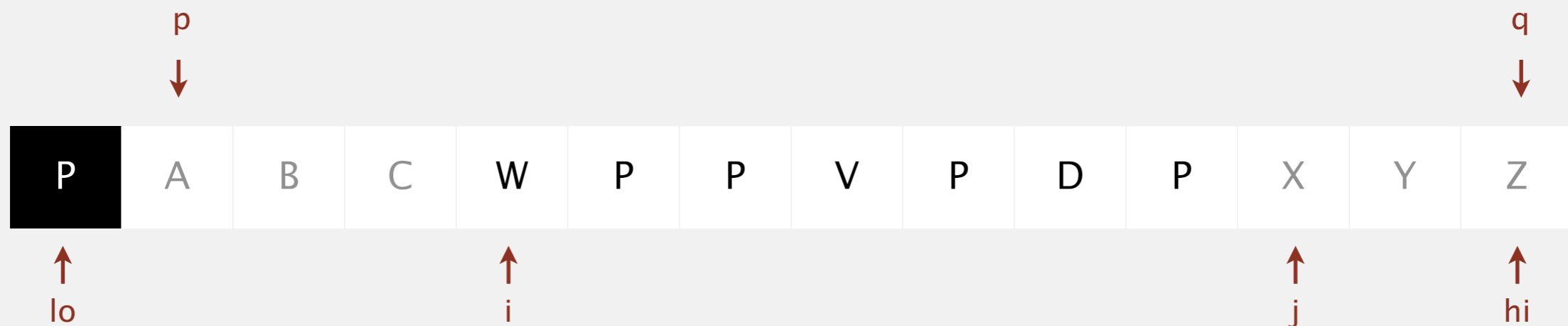
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

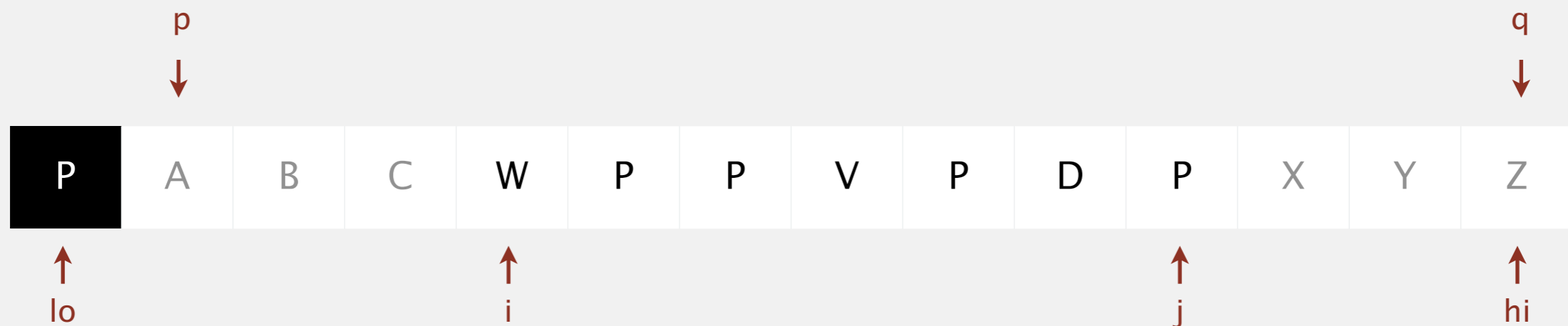
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

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

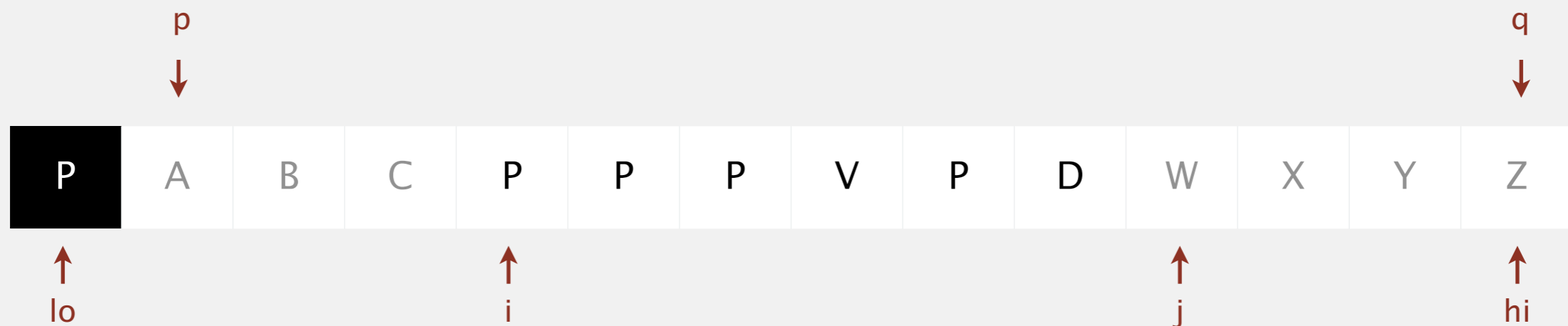


exchange $a[i]$ with $a[j]$

Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

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

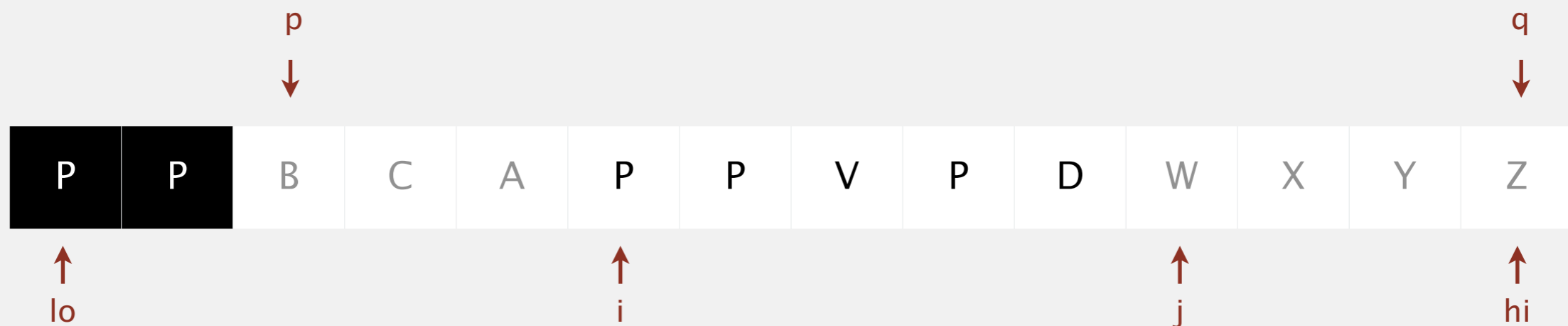


exchange $a[i]$ with $a[p]$ and increment p

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 .



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 .

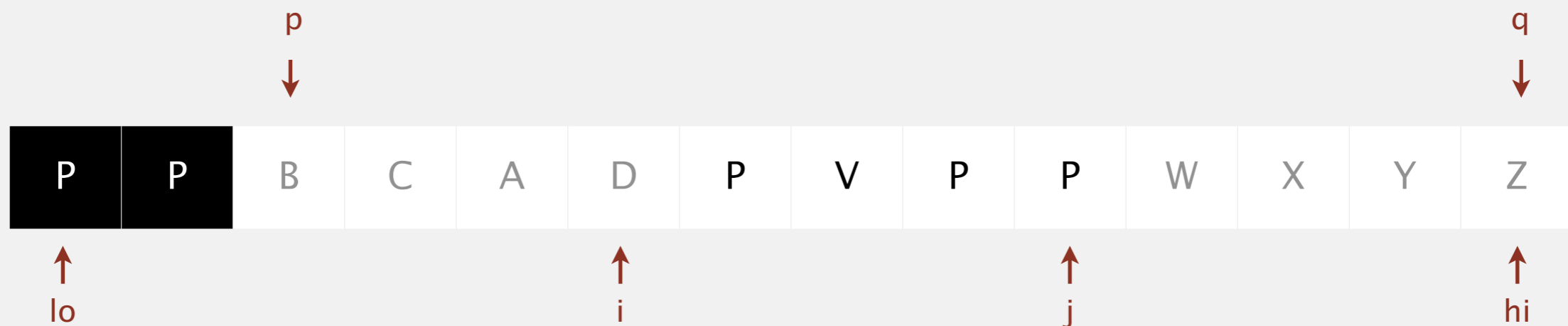


exchange $a[i]$ with $a[j]$

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 .

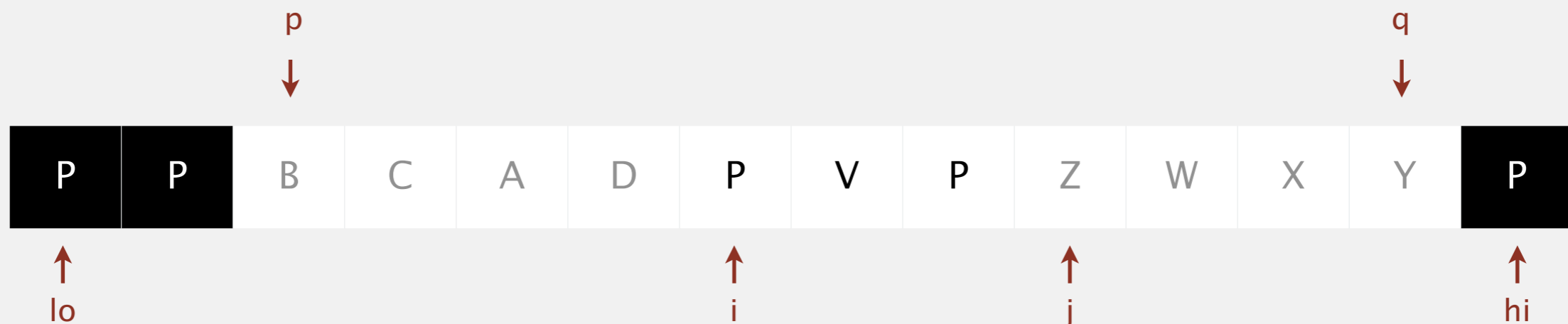


exchange $a[j]$ with $a[q]$ and decrement q

Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

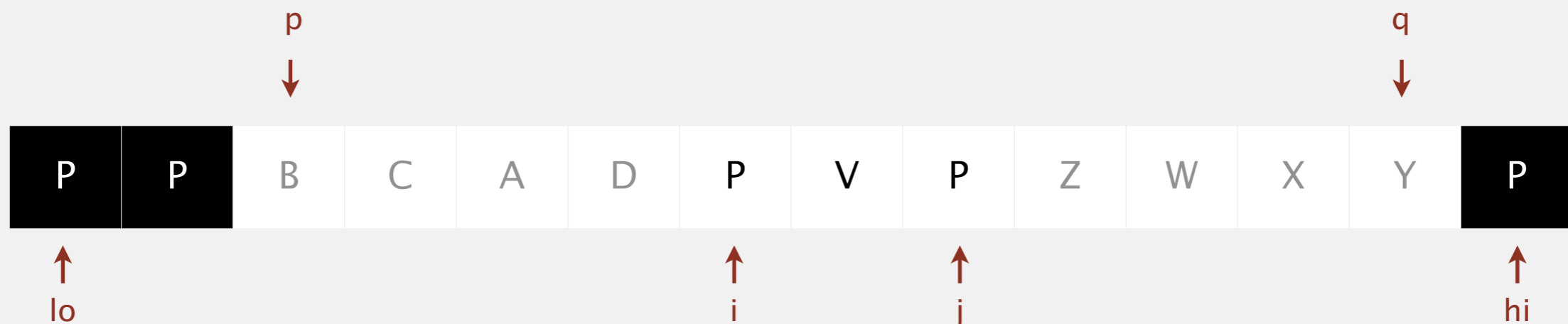
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



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 .

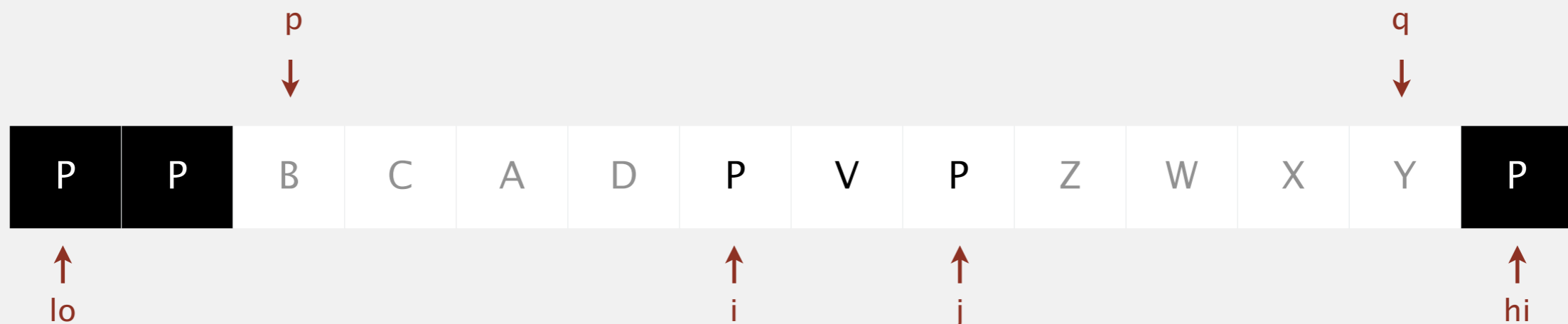


exchange $a[i]$ with $a[j]$

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 .

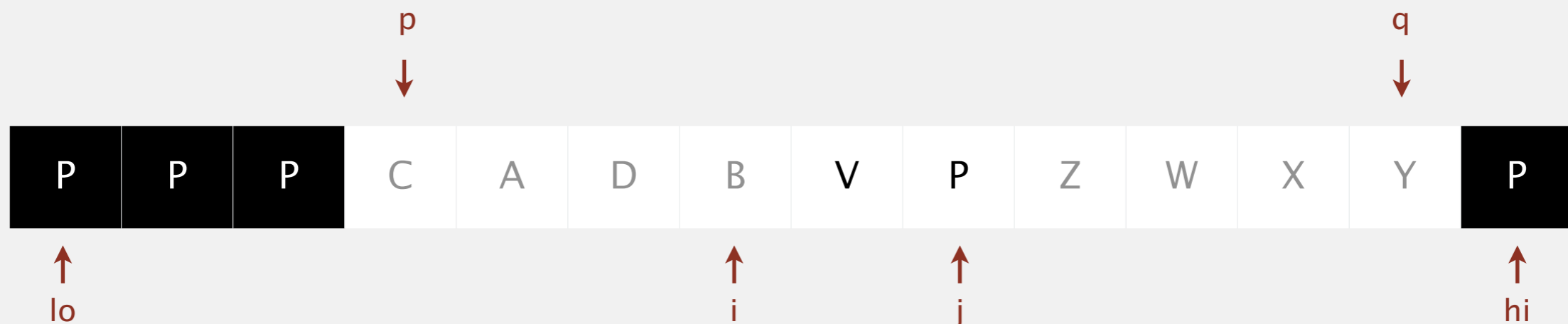


exchange $a[i]$ with $a[p]$ and increment p

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 .

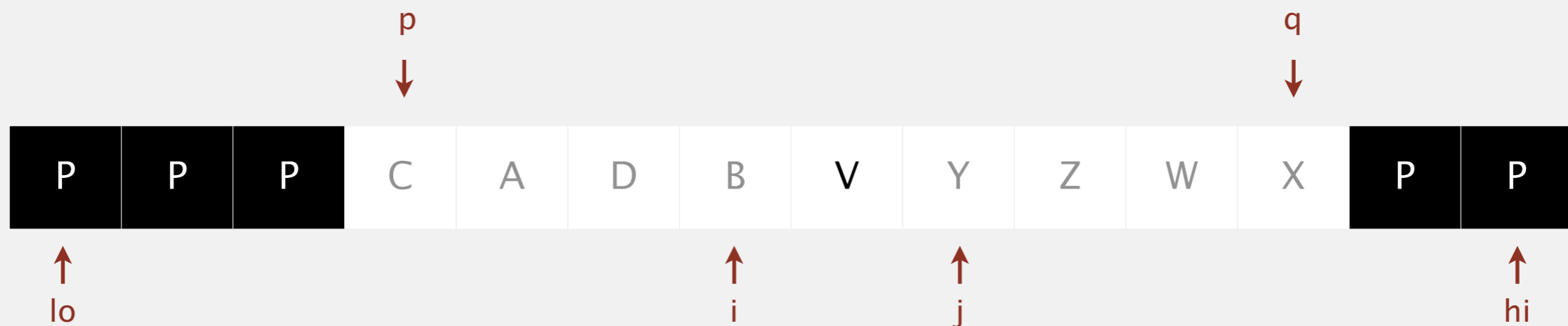


exchange $a[j]$ with $a[q]$ and decrement q

Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

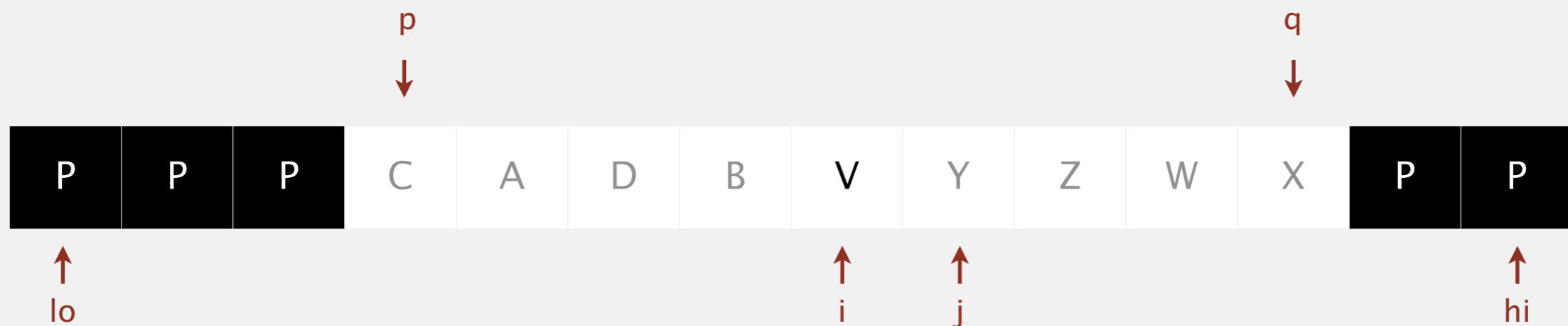
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

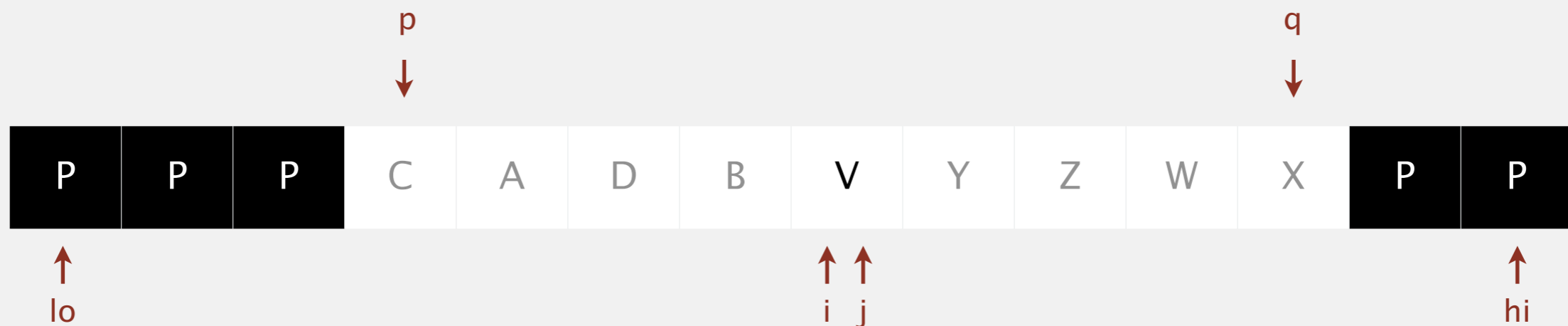
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



Bentley-McIlroy 3-way partitioning demo

Phase I. Repeat until i and j pointers cross.

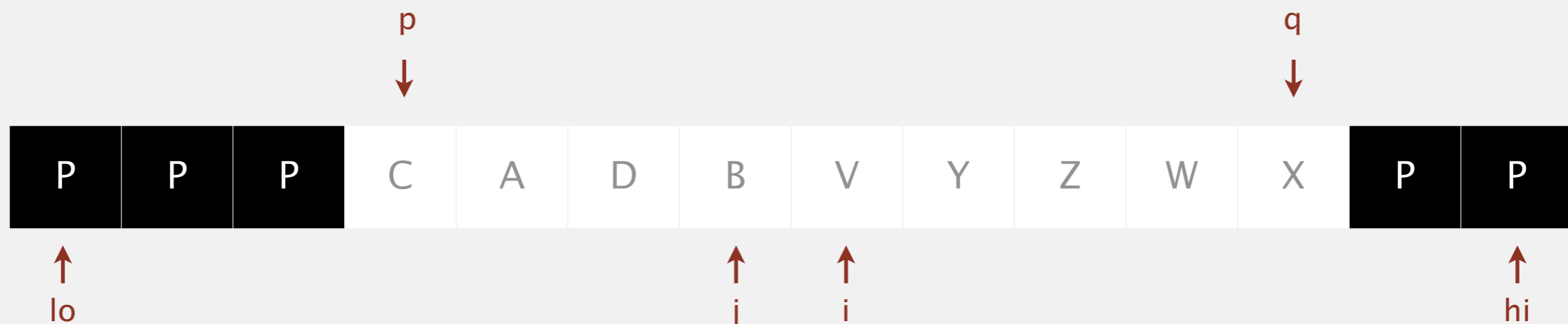
- Scan i from left to right so long as $(a[i] < a[lo])$.
- Scan j from right to left so long as $(a[j] > a[lo])$.
- Exchange $a[i]$ with $a[j]$.
- If $(a[i] == a[lo])$, exchange $a[i]$ with $a[p]$ and increment p .
- If $(a[j] == a[lo])$, exchange $a[j]$ with $a[q]$ and decrement q .



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 .

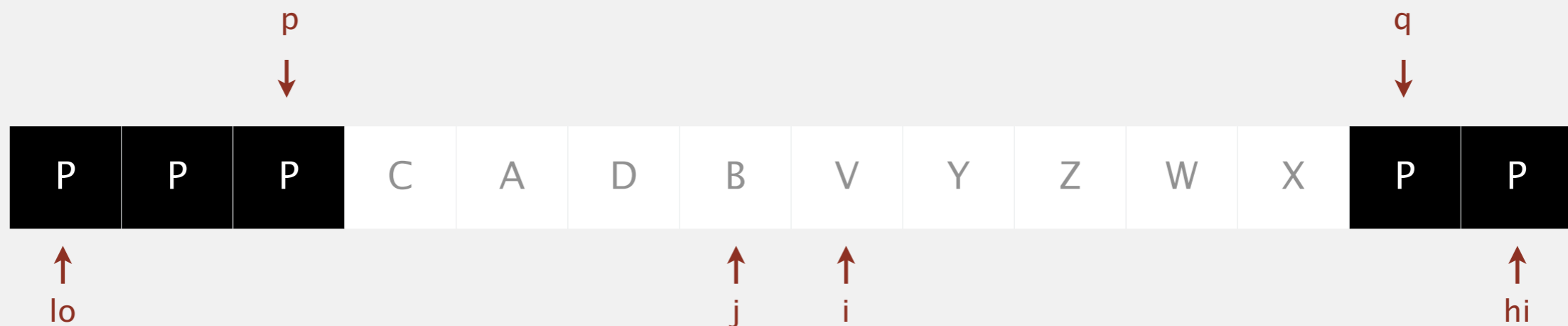


pointers cross

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 $a[q]$.

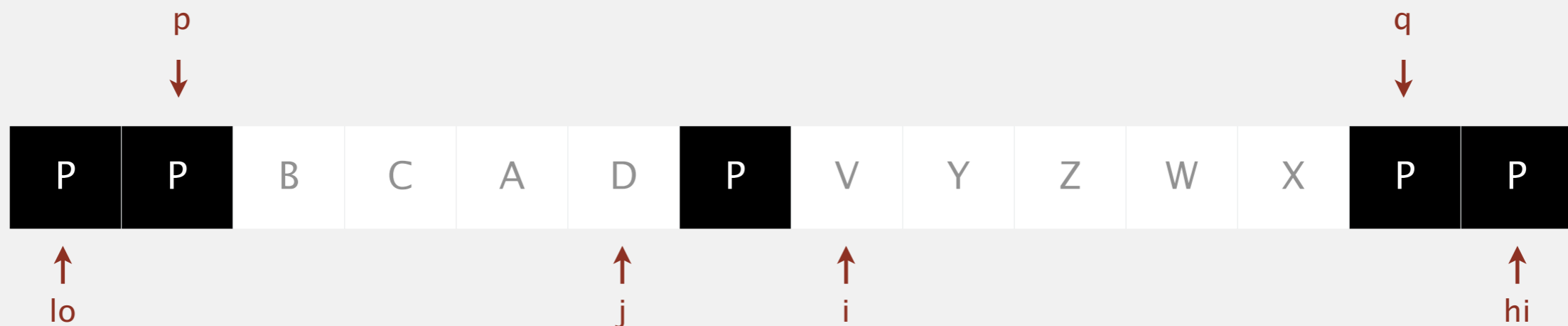


exchange $a[j]$ with $a[p]$

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 $a[q]$.

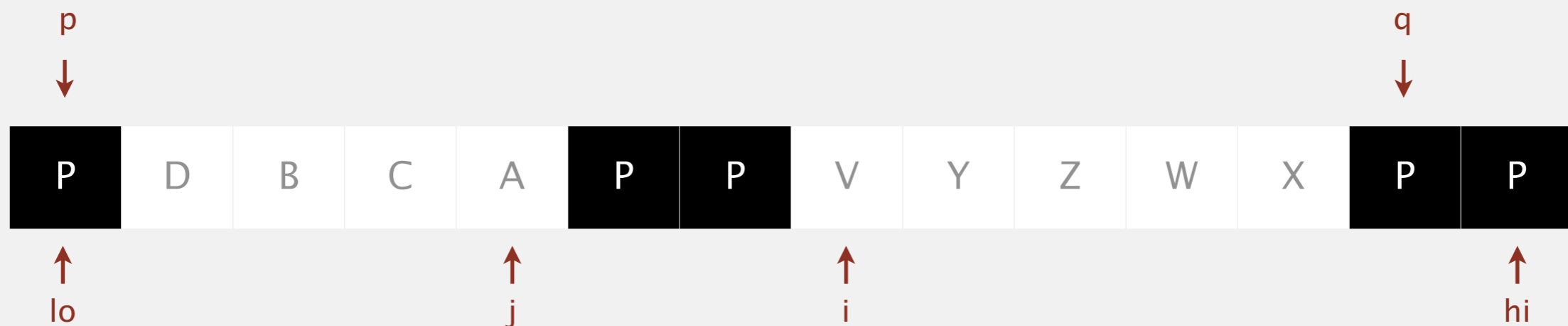


exchange $a[j]$ with $a[p]$

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 $a[p]$

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 $a[q]$.

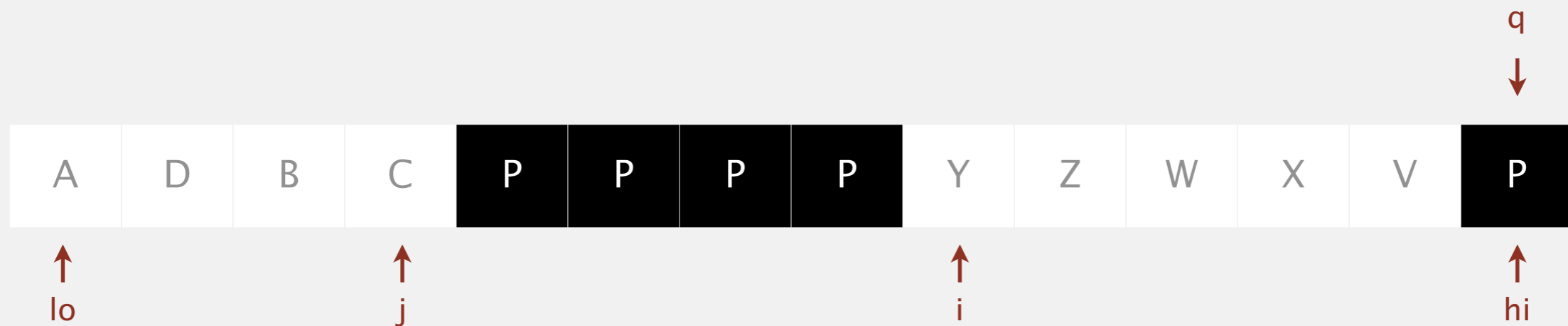


exchange $a[i]$ with $a[q]$

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 $a[q]$.

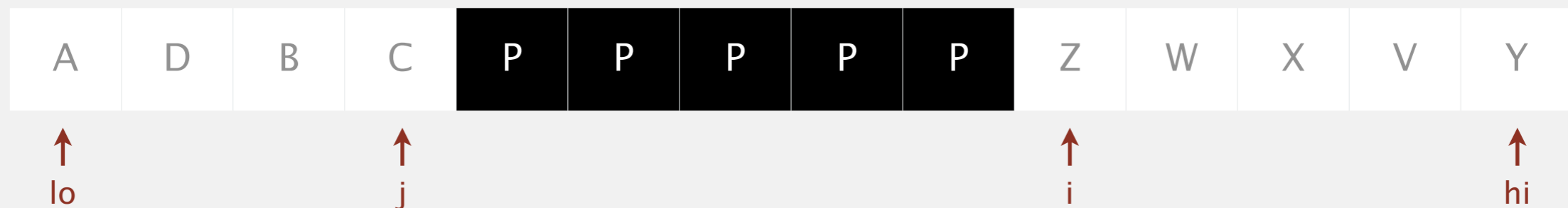


exchange $a[i]$ with $a[q]$

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



<http://algs4.cs.princeton.edu>

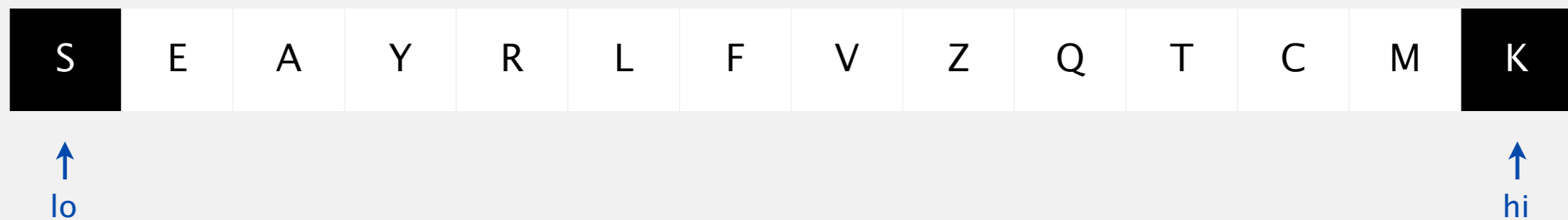
2.3 PARTITIONING DEMOS

- ▶ *Sedgwick 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 $a[lo] \leq a[hi]$.

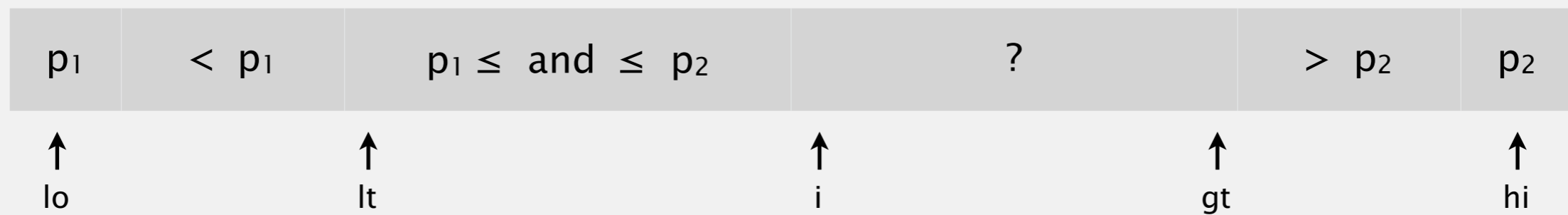


exchange $a[lo]$ and $a[hi]$

Dual-pivot partitioning demo

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 .

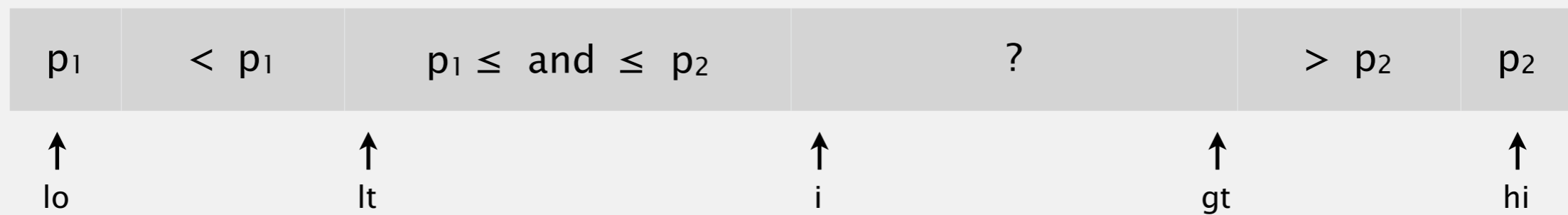


exchange $a[i]$ and $a[lt]$; increment lt and i

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 .

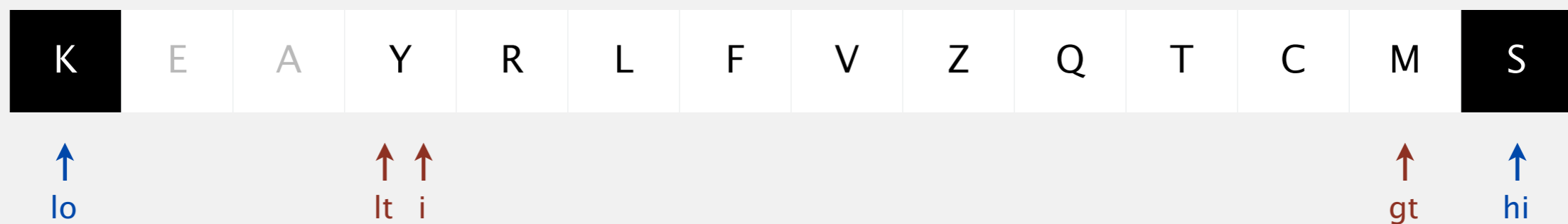
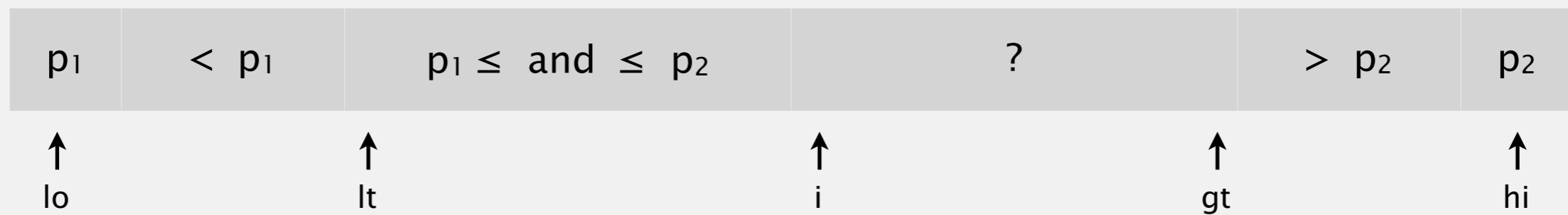


exchange $a[i]$ and $a[lt]$; increment lt and i

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 .

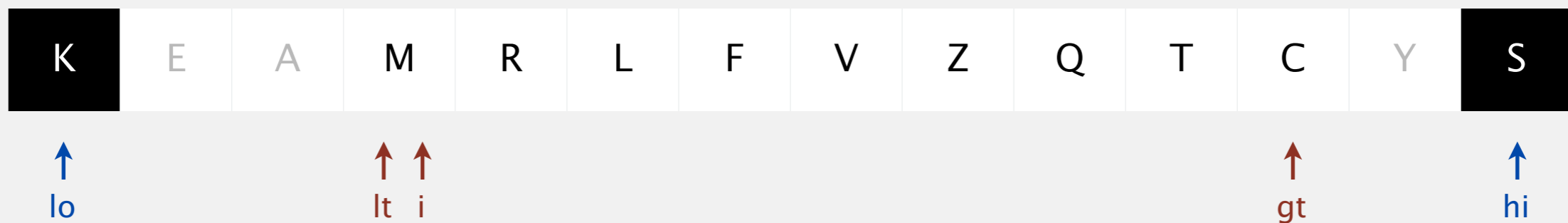
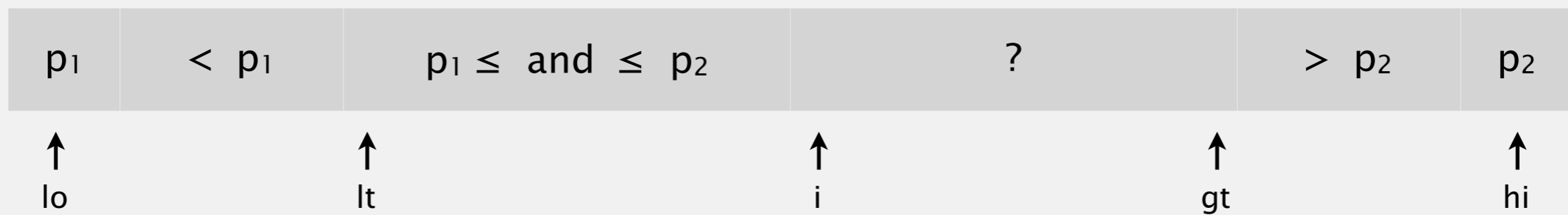


exchange $a[i]$ and $a[gt]$; decrement gt

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 .

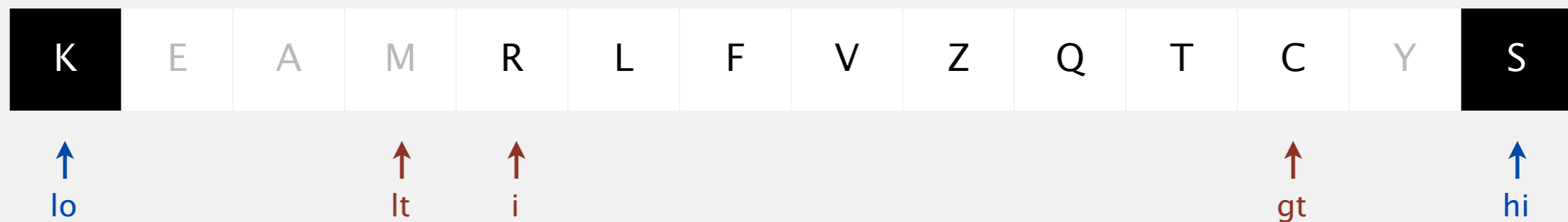
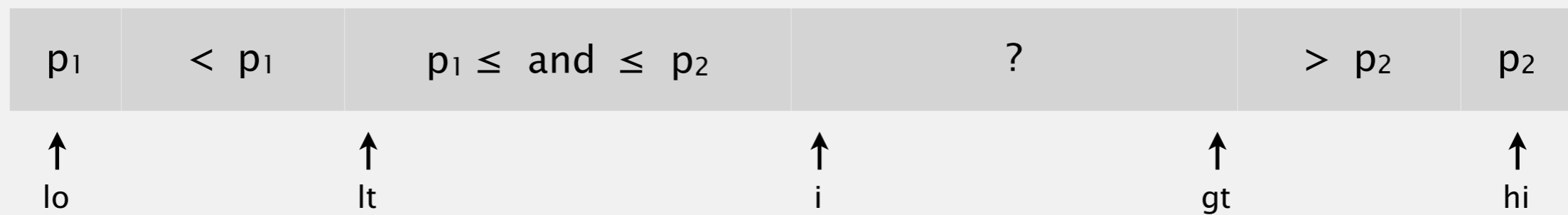


increment i

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 .

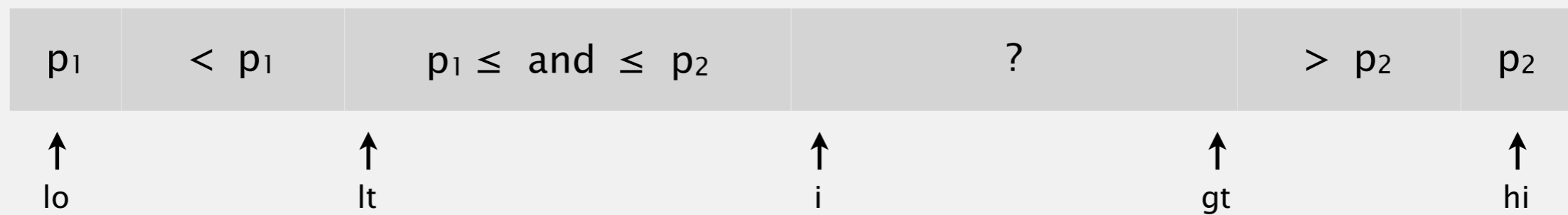


increment i

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 .

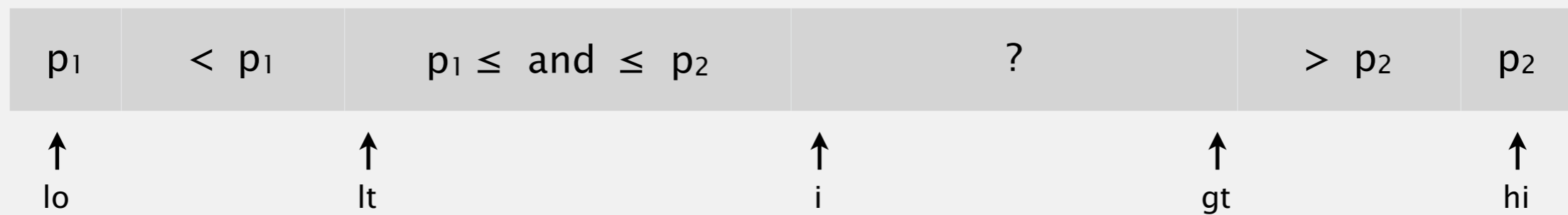


increment i

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 .

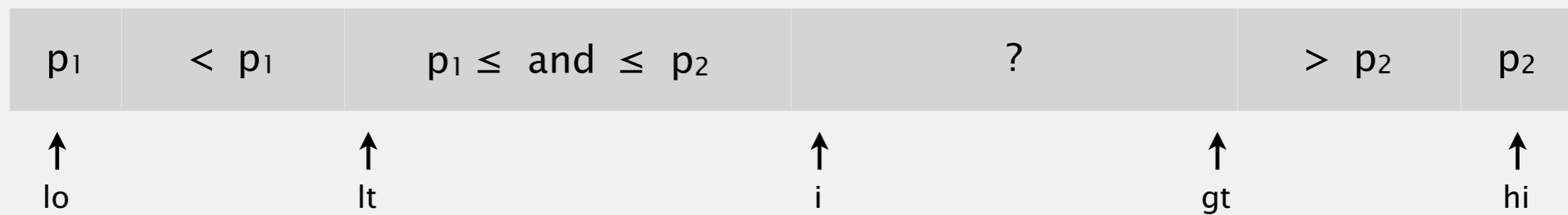


exchange $a[i]$ and $a[lt]$; increment lt and i

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 .

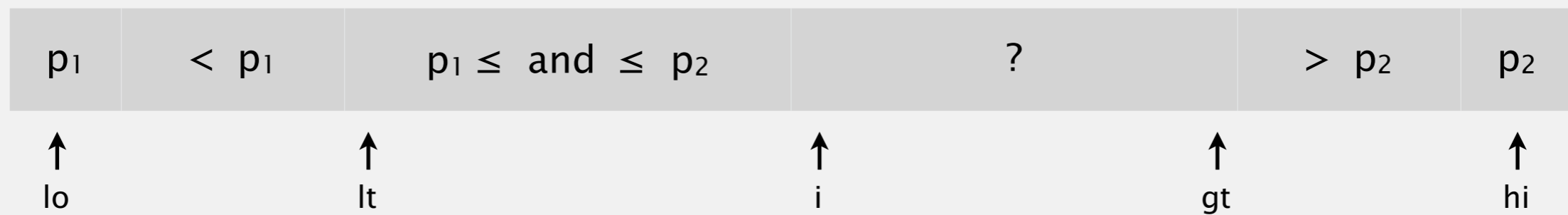


exchange $a[i]$ and $a[gt]$; decrement gt

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 .

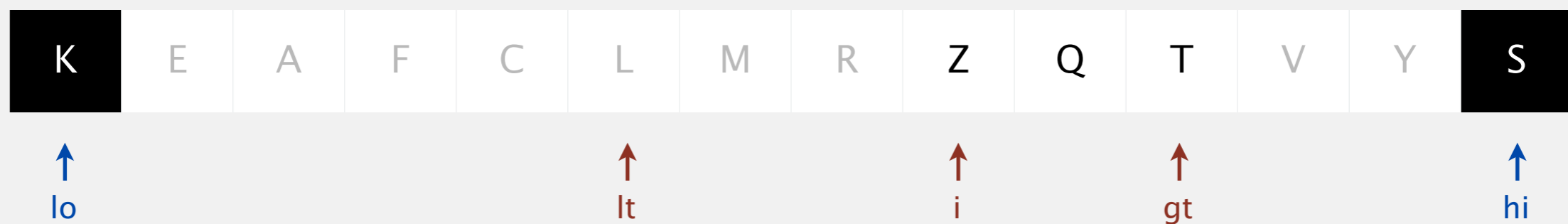
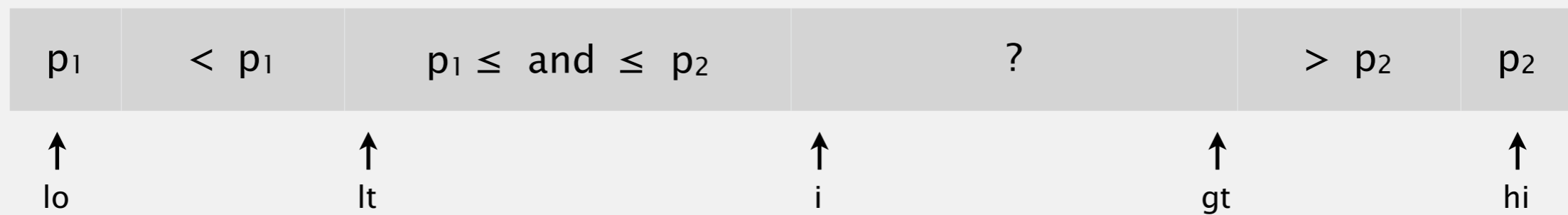


exchange $a[i]$ and $a[lt]$; increment lt and i

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 .

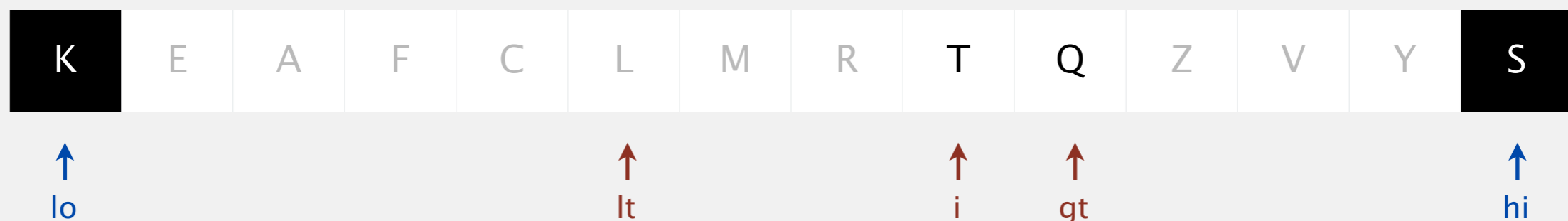
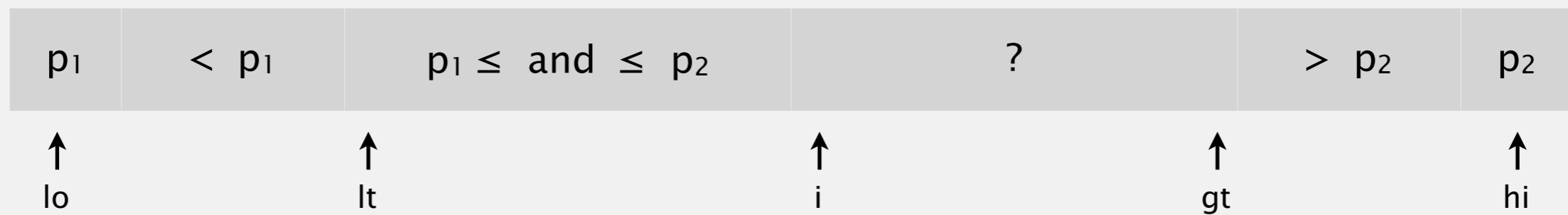


exchange $a[i]$ and $a[gt]$; decrement gt

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 .

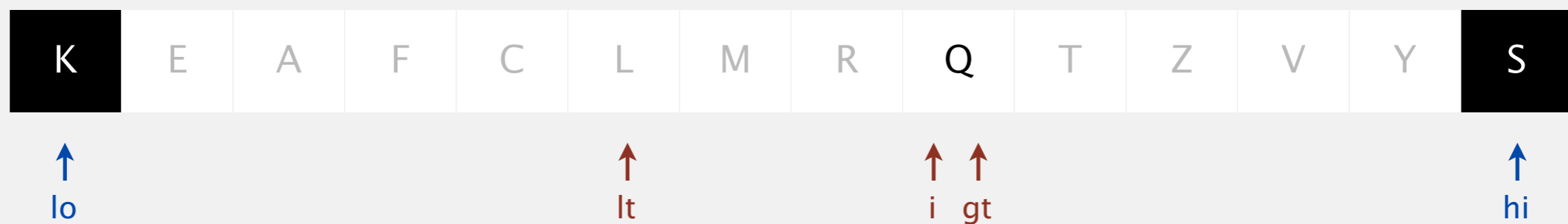
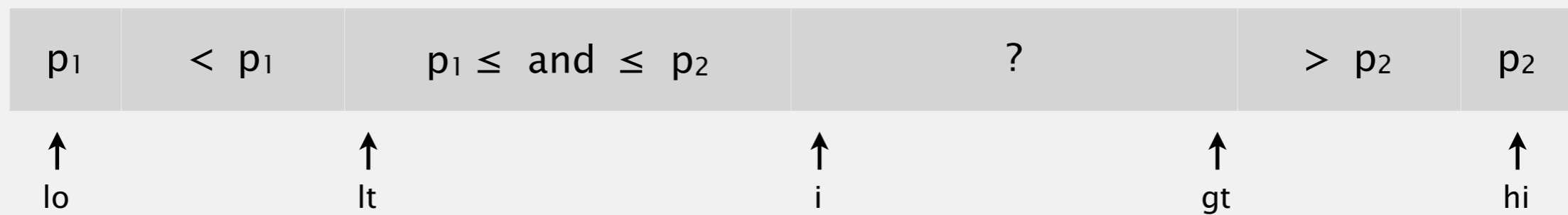


exchange $a[i]$ and $a[gt]$; decrement gt

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 .

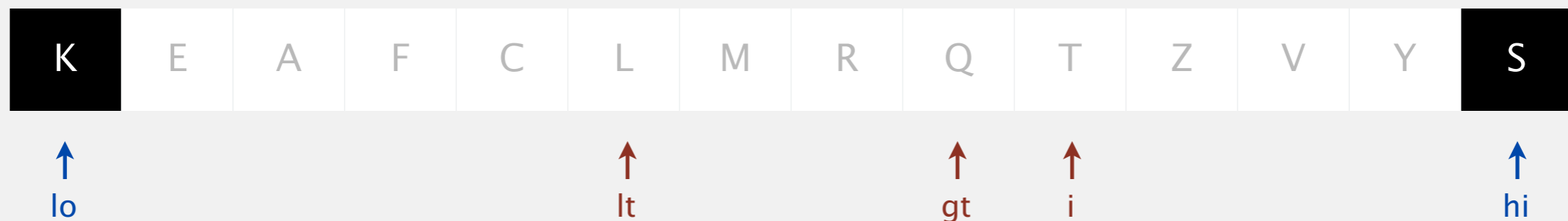
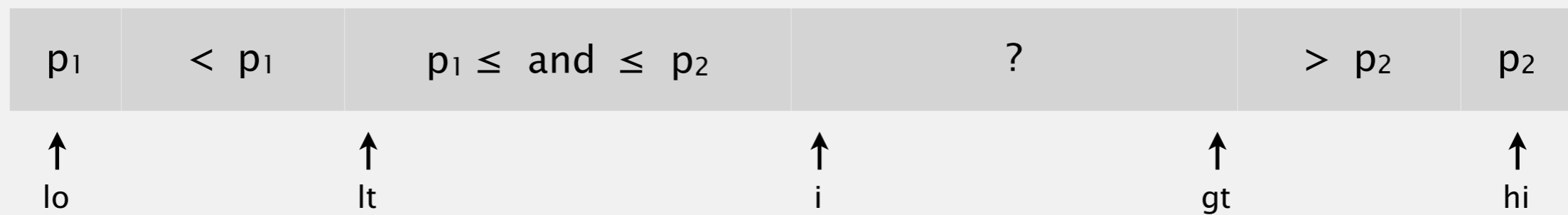


increment i

Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

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

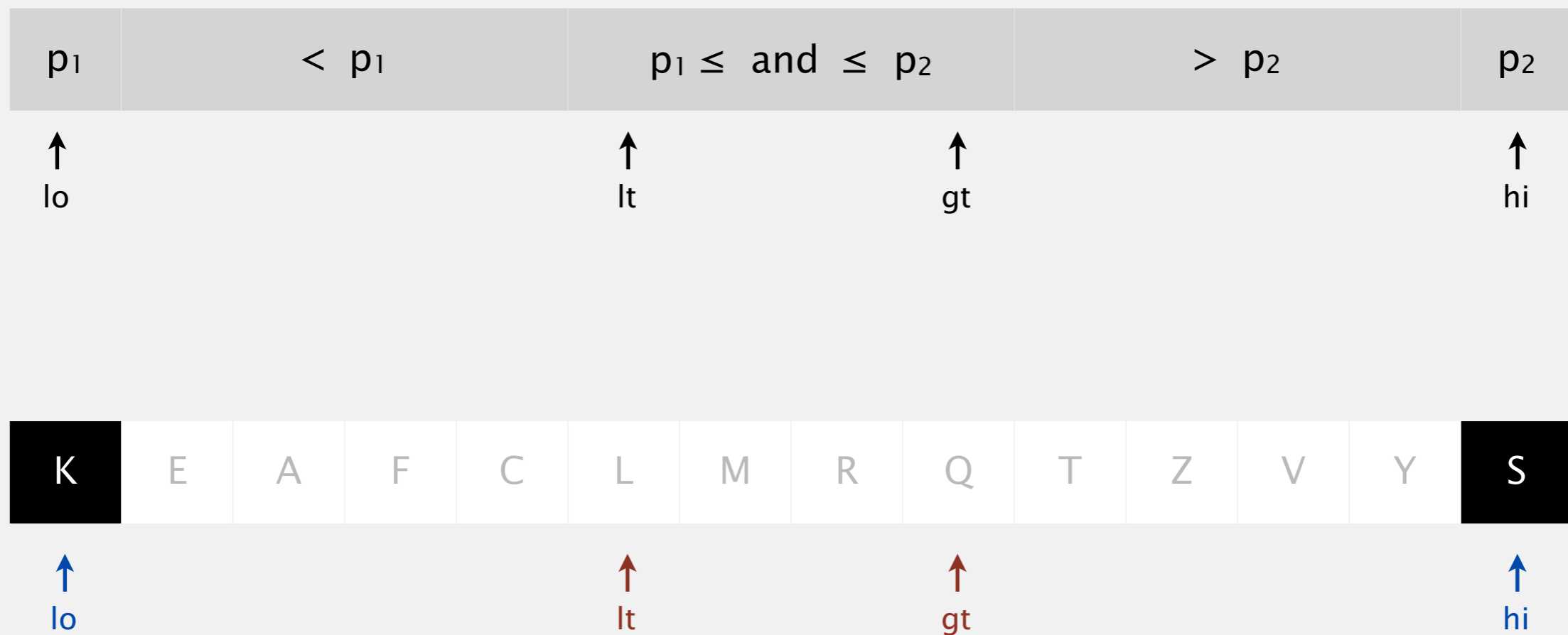


stop when pointers cross

Dual-pivot partitioning demo

Finalize.

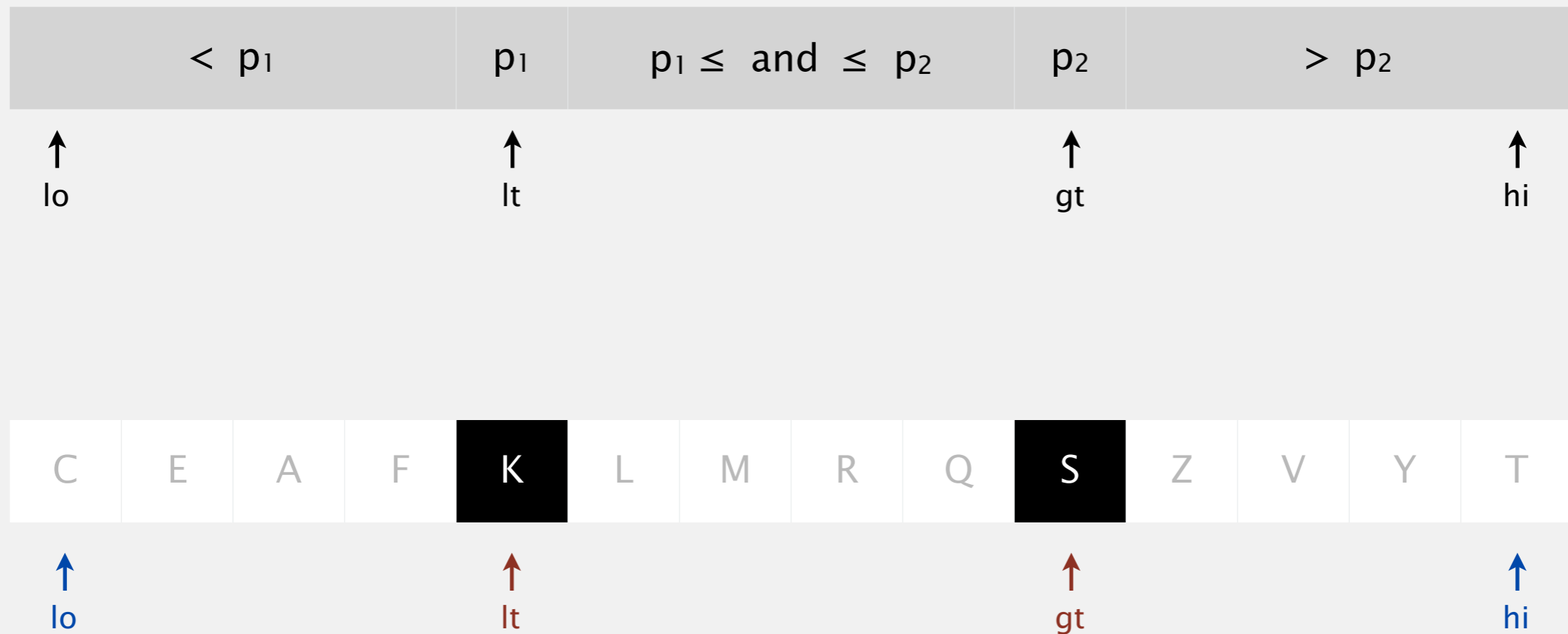
- Exchange $a[lo]$ with $a[--lt]$.
- Exchange $a[hi]$ with $a[++gt]$.



Dual-pivot partitioning demo

Finalize.

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



3-way partitioned