http://algs4.cs.princeton.edu

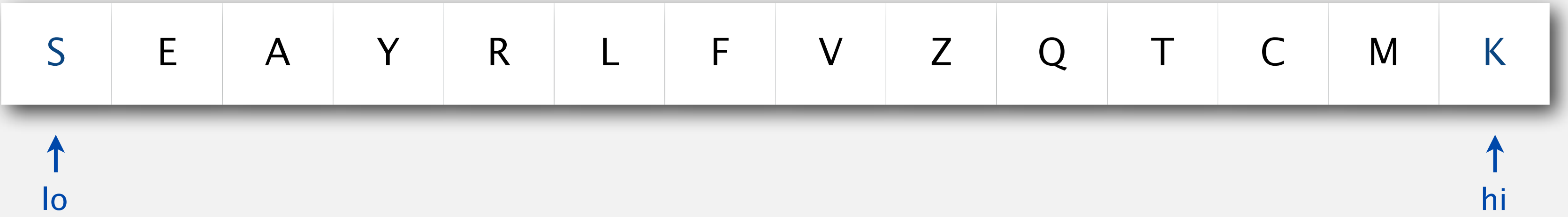
2.3 Quicksort Demos

- Sedgewick 2-way partitioning
- Dijkstra 3-way partitioning
- Dijkstra 3-way partitioning
- Dual-pivot partitioning

Dual-pivot partitioning demo

Initialization.

- Choose `a[lo]` and `a[hi]` as partitioning items.
- Exchange if necessary to ensure `a[lo]` ≤ `a[hi]`.



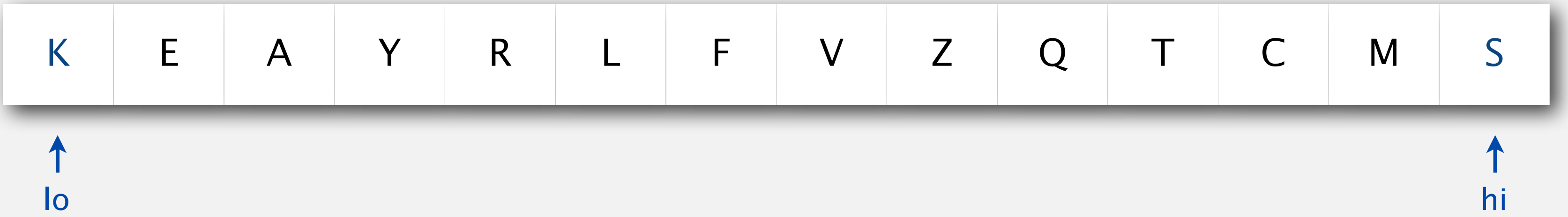
S	E	A	Y	R	L	F	V	Z	Q	T	C	M	K
lo													hi

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]` ≤ `a[hi]`.

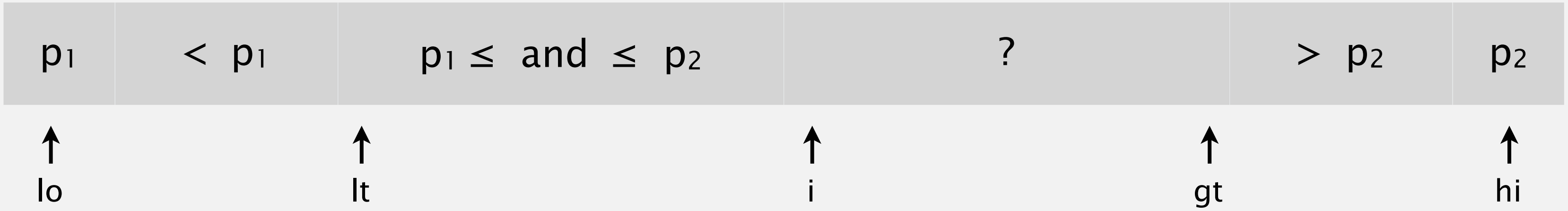


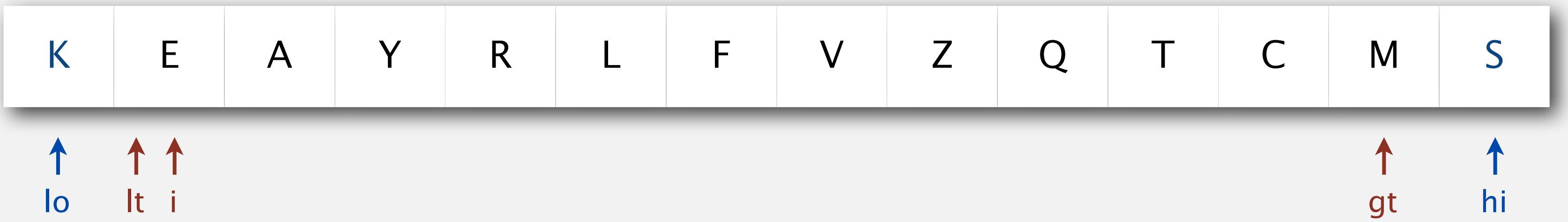
K	E	A	Y	R	L	F	V	Z	Q	T	C	M	S
↑ lo													↑ hi

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`.



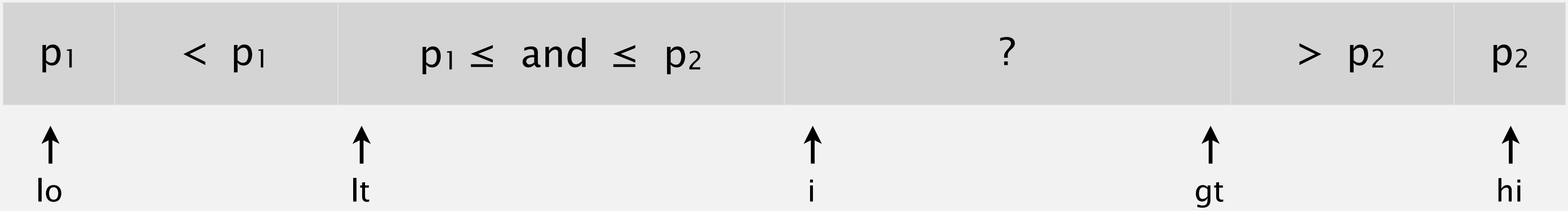


exchange `a[i]` and `a[lt]`; increment `lt` and `i`

Dual-pivot partitioning demo

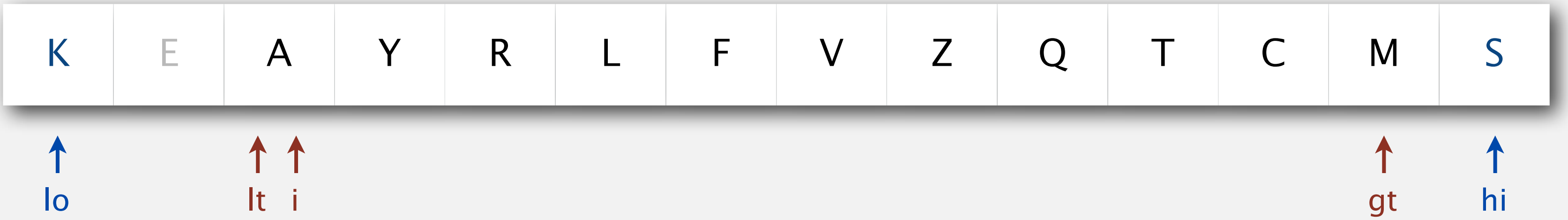
Main loop. Repeat until i and gt pointers cross.

- 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.



p_1	$< p_1$	$p_1 \le$ and $\le p_2$	?	$> p_2$	p_2

lo lt i gt hi



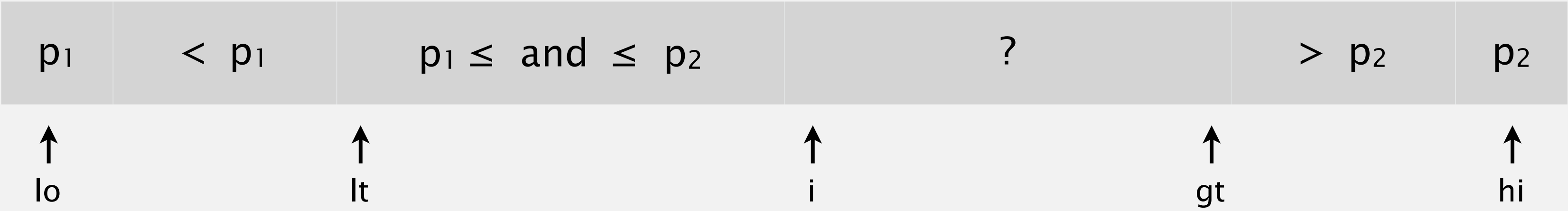
K	E	A	Y	R	L	F	V	Z	Q	T	C	M	S
lo		lt i										gt	hi

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

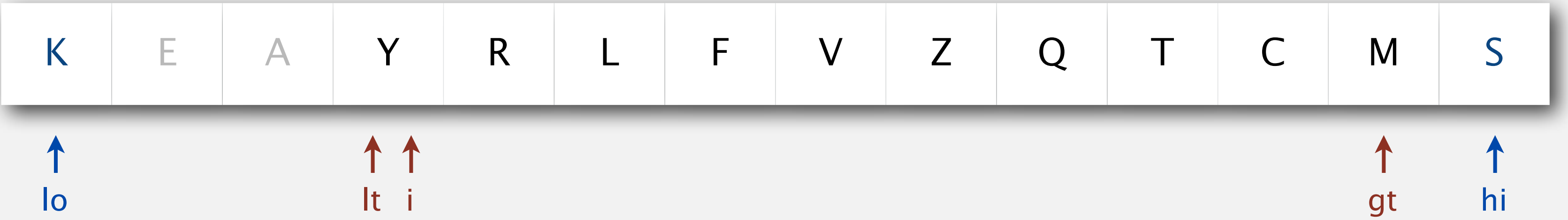
Dual-pivot partitioning demo

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`.



p_1	$< p_1$	$p_1 \leq$ and $\leq p_2$	?	$> p_2$	p_2



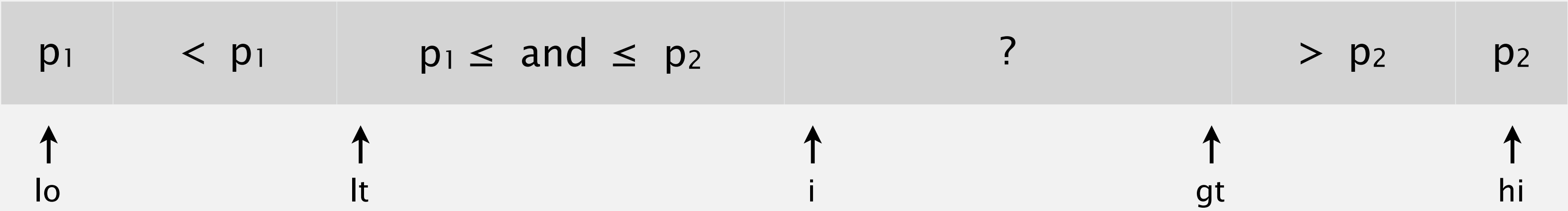
K	E	A	Y	R	L	F	V	Z	Q	T	C	M	S

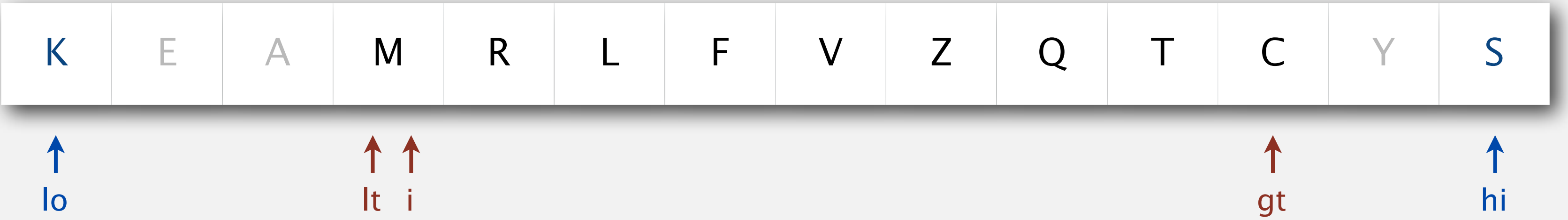
exchange `a[i]` and `a[gt]`; decrement `gt`

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`.



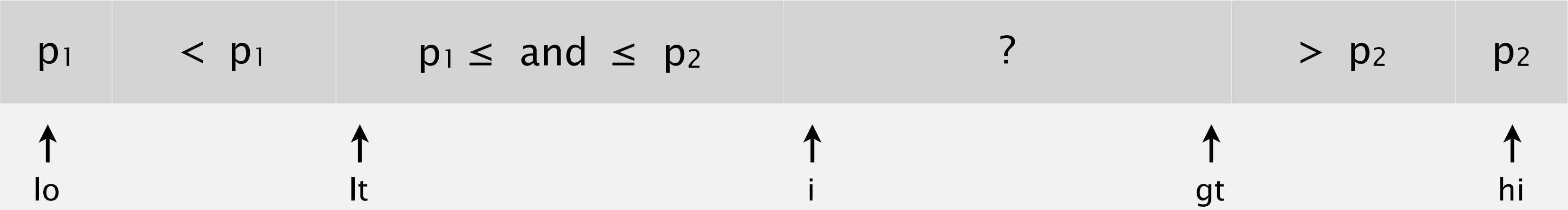


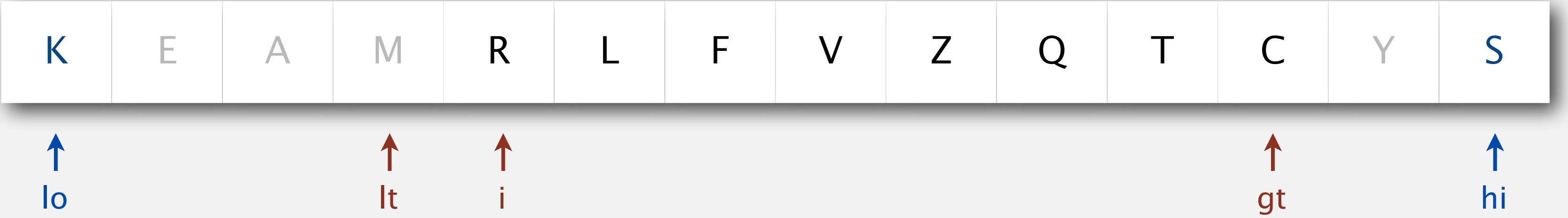
increment `i`

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`.



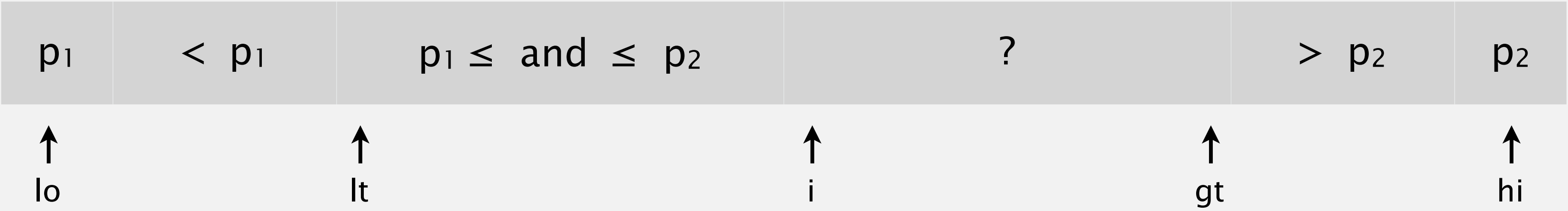


increment i

Dual-pivot partitioning demo

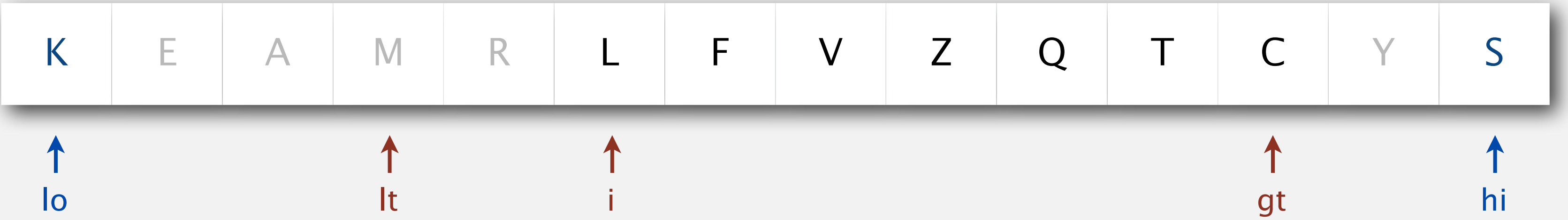
Main loop. Repeat until `i` and `gt` pointers cross.

- 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`.



p_1	$< p_1$	$p_1 \leq$ and $\leq p_2$	?	$> p_2$	p_2

lo lt i gt hi



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

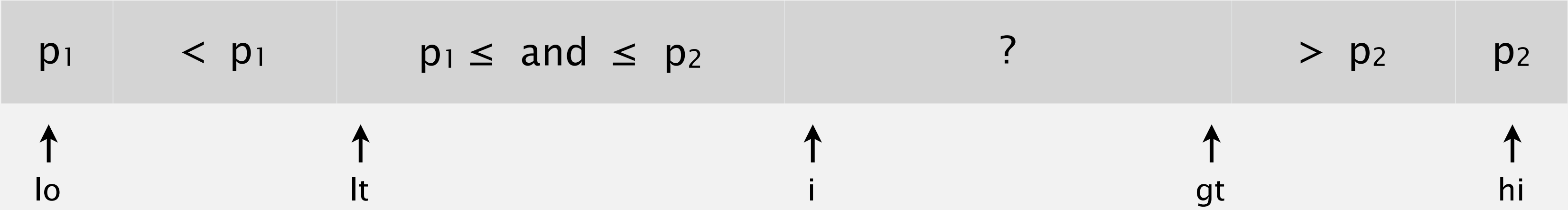
lo lt i gt hi

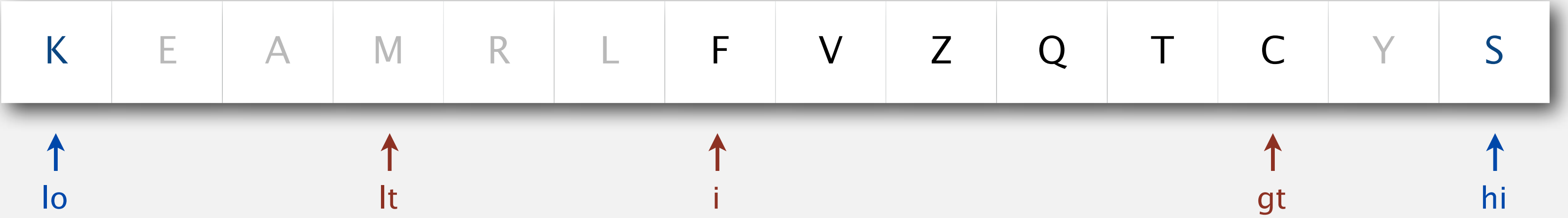
increment `i`

Dual-pivot partitioning demo

Main loop. Repeat until `i` and `gt` pointers cross.

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



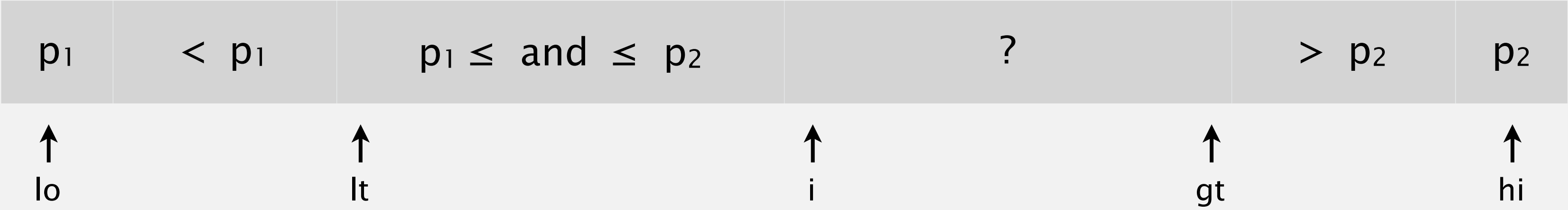


exchange `a[i]` and `a[lt]`; increment `lt` and `i`

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`.



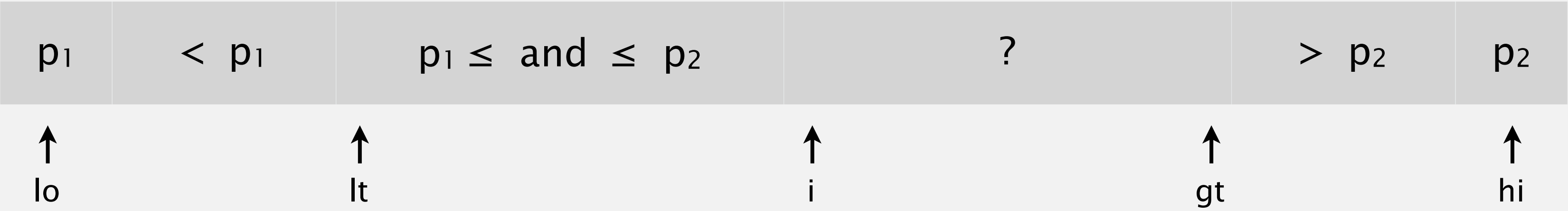
K	E	A	F	R	L	M	V	Z	Q	T	C	Y	S
lo				lt			i				gt		hi

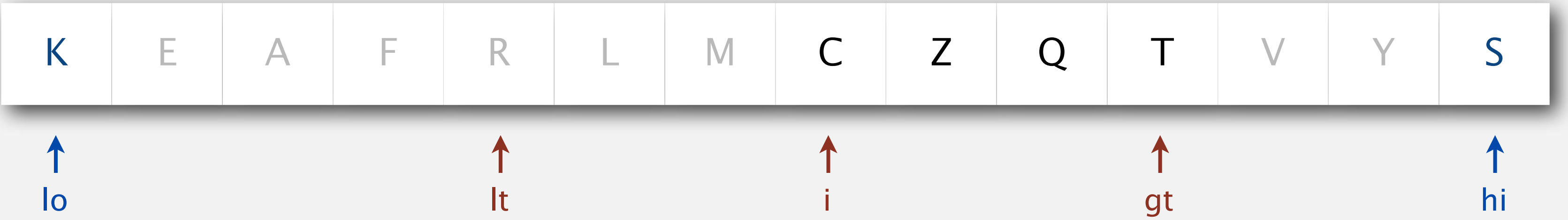
exchange `a[i]` and `a[gt]`; decrement `gt`

Dual-pivot partitioning demo

Main loop. Repeat until `i` and `gt` pointers cross.

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



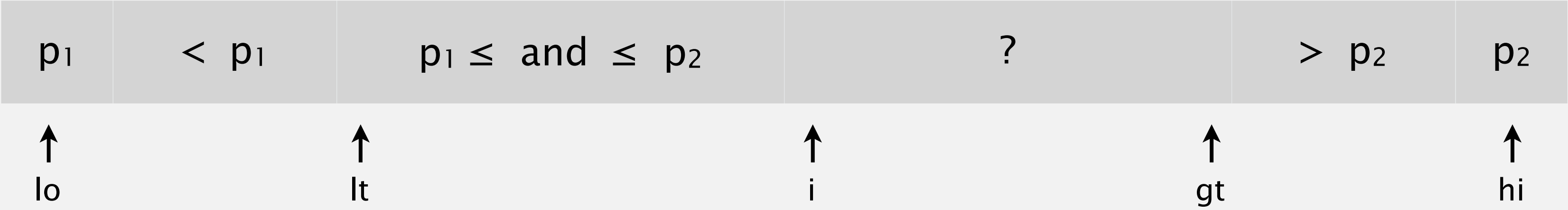


exchange `a[i]` and `a[lt]`; increment `lt` and `i`

Dual-pivot partitioning demo

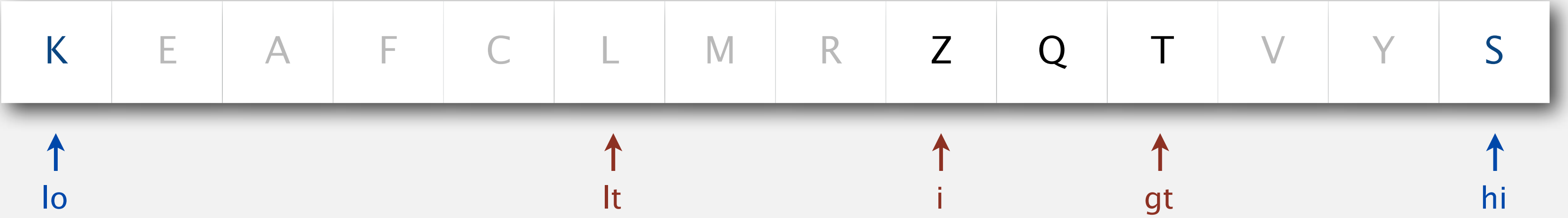
Main loop. Repeat until `i` and `gt` pointers cross.

- 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`.



p_1	$< p_1$	$p_1 \leq$ and $\leq p_2$	?	$> p_2$	p_2

lo lt i gt hi



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

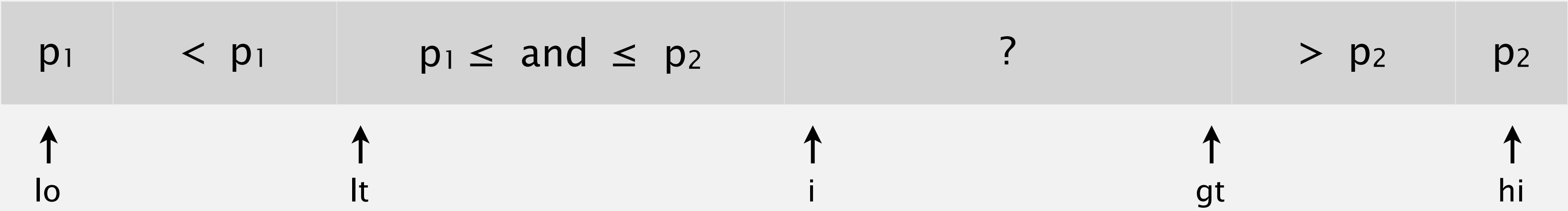
lo lt i gt hi

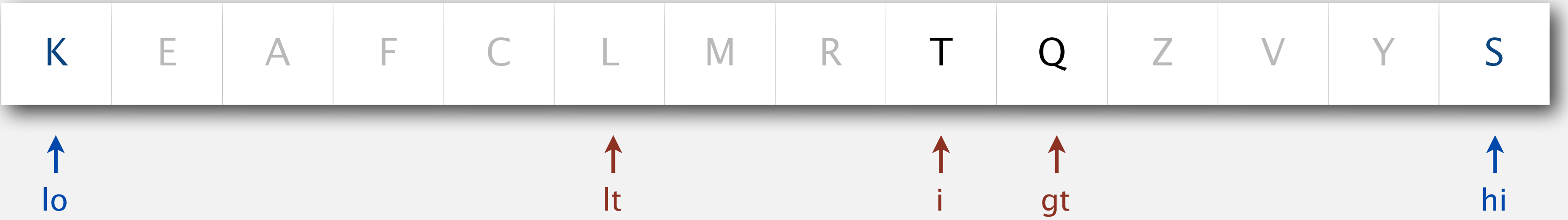
exchange `a[i]` and `a[gt]`; decrement `gt`

Dual-pivot partitioning demo

Main loop. Repeat until `i` and `gt` pointers cross.

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



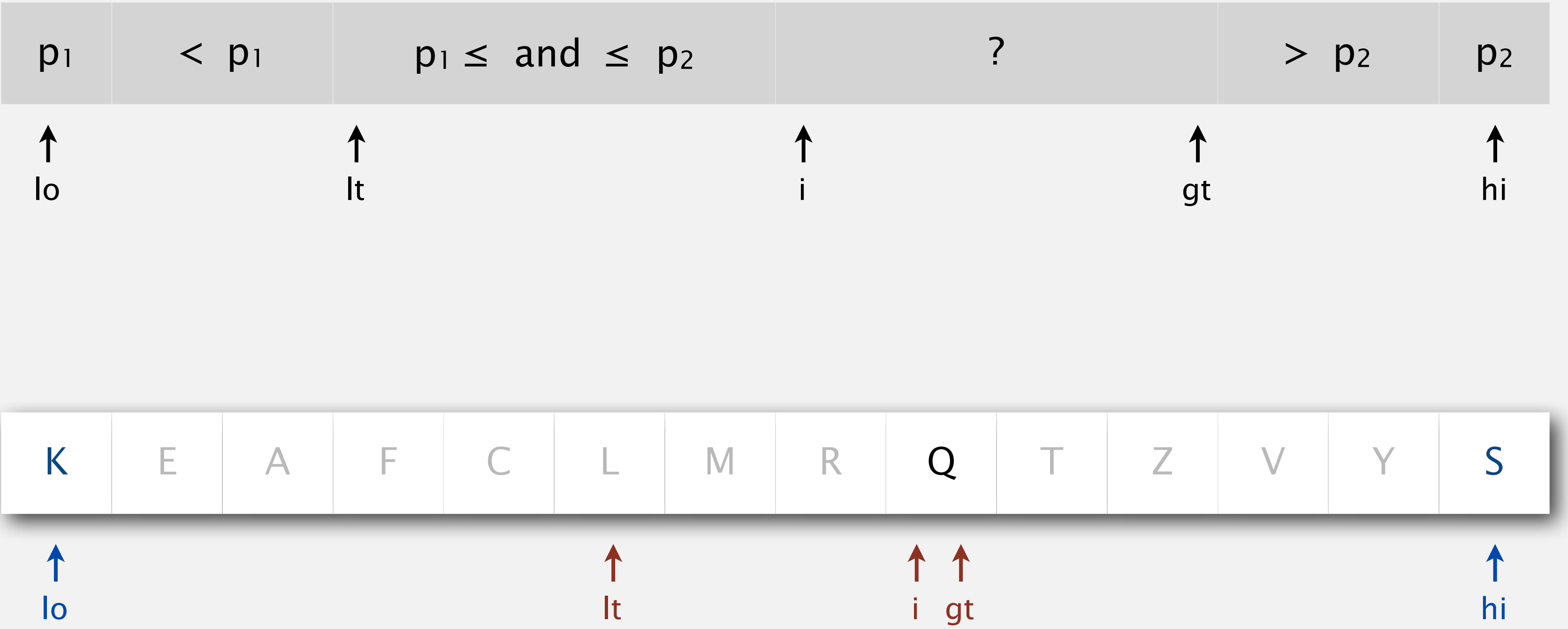


exchange `a[i]` and `a[gt]`; decrement `gt`

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`.



p_1	$< p_1$	$p_1 \le$ and $\le p_2$	?	$> p_2$	p_2

lo lt i gt hi

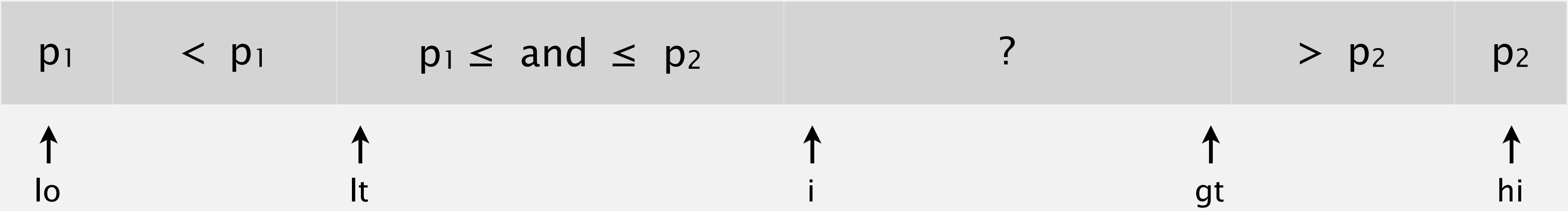
K	E	A	F	C	L	M	R	Q	T	Z	V	Y	S
lo					lt			i gt					hi

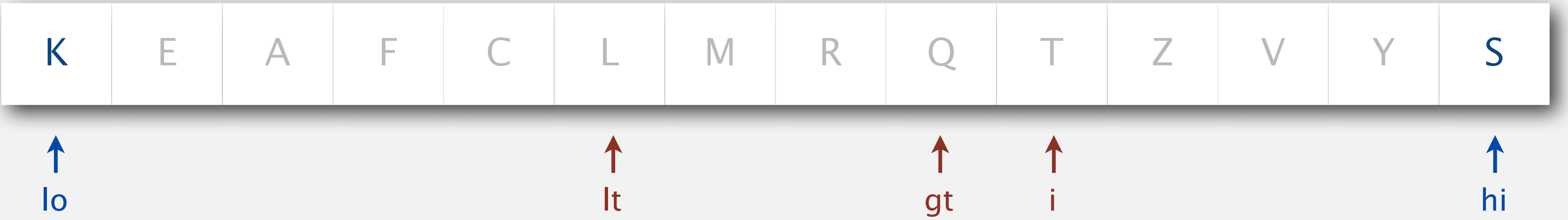
increment `i`

Dual-pivot partitioning demo

Main loop. Repeat until `i` and `gt` pointers cross.

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



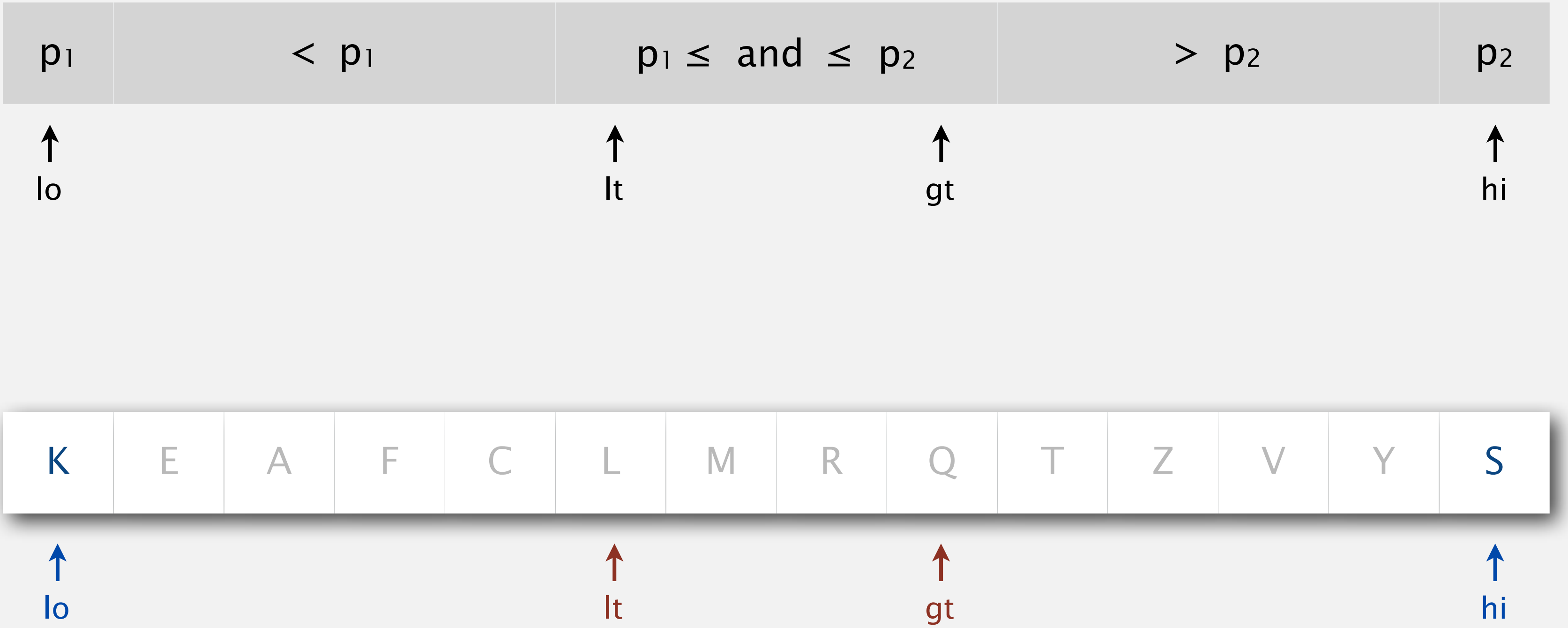


stop when pointers cross

Dual-pivot partitioning demo

Finalize.

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



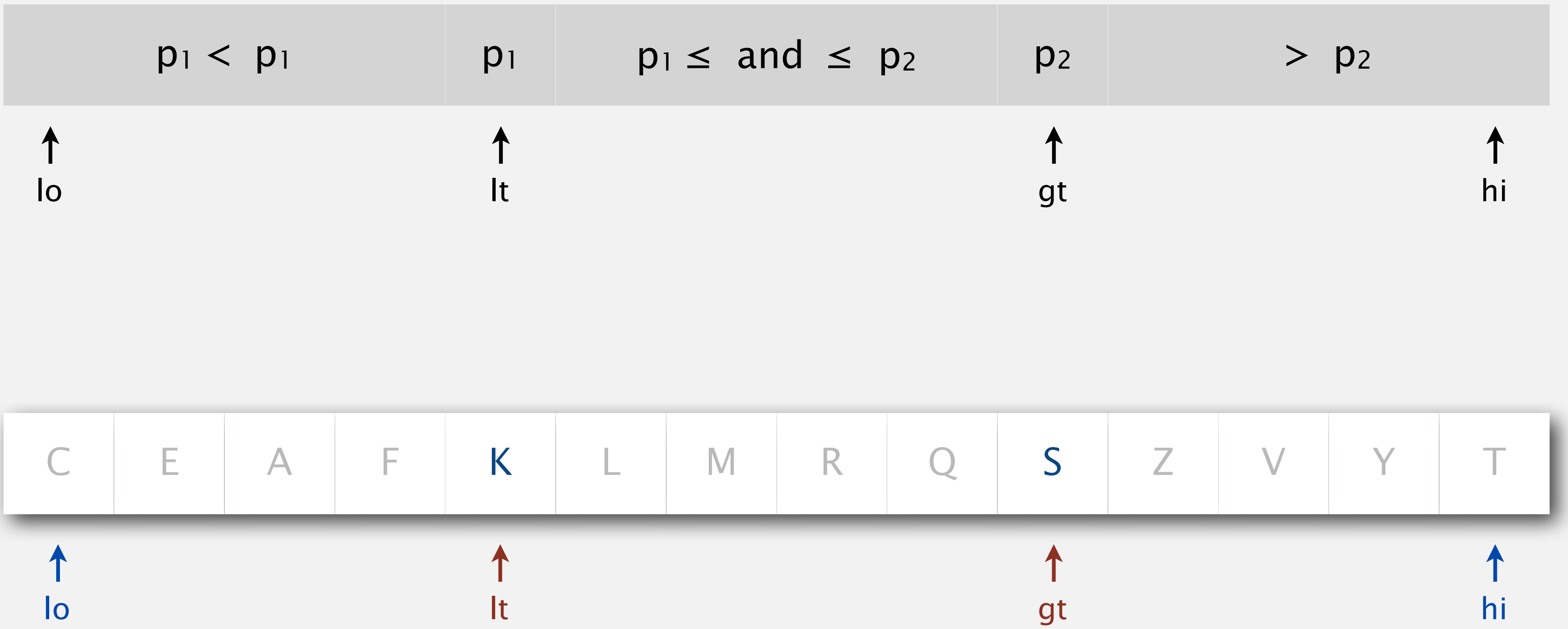
p_1	$< p_1$	$p_1 \le$ and $\le p_2$	$> p_2$	p_2

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

Dual-pivot partitioning demo

Finalize.

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



$p_1 < p_1$	p_1	$p_1 \le$ and $\le p_2$	p_2	$> p_2$

lo lt gt hi

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

lo lt gt hi

3-way partitioned