### 2.3 Partitioning Demos

- Hoare 2-way partitioning
- Dijkstra 3-way partitioning
- Bentley-Mcllroy 3-way partitioning
- dual-pivot partitioning

```
Robert Sedgewick I Kevin Wayne
```

https://algs4.cs.princeton.edu

### 2.3 Partitioning Demos

- Hoare 2-way partitioning
- Dijkstra 3-way partitioning

Algorithms

Robert Sedgewick | Kevin Wayne

- Bentley-Mcllroy 3-way pártitioning
- dual-pivot partitioning
https://algs4.cs.princeton.edu


## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| $K$ | $R$ | $A$ | $T$ | $E$ | $L$ | $E$ | $P$ | $U$ | $I$ | $M$ | $Q$ | $C$ | $X$ | $O$ | $S$ |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | $\uparrow$ | $\uparrow$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| lo | $i$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  | i |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | R | A | T | E | L | E | P | U | I | M | Q | C | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ | $\uparrow$ |  |  |  |  |  |  |  |  |  |  |  |  | $\uparrow$ |  |
| Io | i |  |  |  |  |  |  |  |  |  |  |  |  | j |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | R | A | T | E | L | E | P | U | I | M | Q | C | X | O | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ | $\uparrow$ |  |  |  |  |  |  |  |  |  |  |  | $\uparrow$ |  |  |
| lo | i |  |  |  |  |  |  |  |  |  |  |  | j |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | R | A | T | E | L | E | P | U | I | M | Q | C | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ | $\uparrow$ |  |  |  |  |  |  |  |  |  |  | $\uparrow$ |  |  |  |
| lo | i |  |  |  |  |  |  |  |  |  |  | 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | C | A | T | E | L | E | P | U | I | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ | $\uparrow$ |  |  |  |  |  |  |  |  |  |  | $\uparrow$ |  |  |  |
| lo | i |  |  |  |  |  |  |  |  |  |  | j |  |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | C | A | T | E | L | E | P | U | I | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  | $\uparrow$ |  |  |  |  |  |  |  |  |  | $\uparrow$ |  |  |  |
| Io |  | i |  |  |  |  |  |  |  |  |  | j |  |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | C | A | T | E | L | E | P | U | I | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  |  | $\uparrow$ |  |  |  |  |  |  |  |  | $\uparrow$ |  |  |  |
| Io |  |  | i |  |  |  |  |  |  |  |  | j |  |  |  |

stop $\mathbf{i}$ scan because $\mathrm{a}[\mathrm{i}]>=\mathrm{a}[\mathrm{lo}]$

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | C | A | T | E | L | E | P | U | I | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  |  | $\uparrow$ |  |  |  |  |  |  |  | $\uparrow$ |  |  |  |  |
| lo |  |  | i |  |  |  |  |  |  |  | j |  |  |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | C | A | T | E | L | E | P | U | I | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  |  | $\uparrow$ |  |  |  |  |  |  | $\uparrow$ |  |  |  |  |  |
| lo |  |  | i |  |  |  |  |  |  | j |  |  |  |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].

| K | C | A | T | E | L | E | P | U | I | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  |  | $\uparrow$ |  |  |  |  |  | $\uparrow$ |  |  |  |  |  |  |
| Io |  |  | i |  |  |  |  |  | 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].

| K | C | A | 1 | E | L | E | P | U | T | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ 10 |  |  | $\uparrow$ |  |  |  |  |  | $\uparrow$ $j$ |  |  |  |  |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

| K | C | A | I | E | L | E | P | U | T | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  |  |  | $\uparrow$ |  |  |  |  | $\uparrow$ |  |  |  |  |  |  |
| Io |  |  |  | i |  |  |  |  | j |  |  |  |  |  |  |

## Quicksort partitioning demo

Repeat until i and j pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].

| K | C | A | I | E | L | E | P | U | T | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ lo |  |  |  |  | $\uparrow$ |  |  |  | $\uparrow$ $j$ |  |  |  |  |  |  |

stop i scan because $\mathrm{a}[\mathrm{i}]>=\mathrm{a}[\mathrm{lo}]$

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].

| K | C | A | I | E | L | E | P | U | T | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ¢ 10 |  |  |  |  | $\uparrow$ $i$ |  |  | $\uparrow$ j |  |  |  |  |  |  |  |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].

| K | C | A | I | E | L | E | P | U | T | M | Q | R | X | O | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ lo |  |  |  |  | $\uparrow$ | $\uparrow$ 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] < $a[1 o]$ ).
- 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] < $a[1 o]$ ).
- 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].

| $K$ | $C$ | $A$ | $I$ | $E$ | $E$ | $L$ | $P$ | $U$ | $T$ | $M$ | $Q$ | $R$ | $X$ | $O$ | $S$ |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

stop j scan because $\mathrm{a}[\mathrm{j}]<=\mathrm{a}[\mathrm{lo}]$

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

When pointers cross.

- Exchange a[1o] with a[j].

| $K$ | $C$ | $A$ | $I$ | $E$ | $E$ | $L$ | $P$ | $U$ | $T$ | $M$ | $Q$ | $R$ | $X$ | $O$ | $S$ |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

## Quicksort partitioning demo

Repeat until i and $j$ pointers cross.

- Scan i from left to right so long as (a[i] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > $a[1 o]$ ).
- Exchange a[i] with a[j].

When pointers cross.

- Exchange a[1o] with a[j].

| E | C | A | I | E | K | L | P | U | T | M | Q | R | X | 0 | S |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ |  |  |  |  | $\uparrow$ |  |  |  |  |  |  |  |  |  | $\uparrow$ |
| Io |  |  |  |  | j |  |  |  |  |  |  |  |  |  | hi |

### 2.3 Partitioning Demos

- Hoare2-way partitioning
- Dijkstra 3-way partitioning

Algorithms

- Bentley-Mcllroy 3-way pártitioning
- dual-pivot partitioning

Robert Sedgewick I Kevin Wayne
https://algs4.cs.princeton.edu

## Dijkstra 3-way partitioning demo

- Let $v$ be partitioning item a[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[7t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t 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[1o].
- Scan i from left to right.
- (a[i] < v): exchange a[1t] with a[i]; increment both 1t and i
- (a[i] > v): exchange a[gt] with a[i]; decrement gt
- (a[i] == v): increment i



### 2.3 Partitioning Demos

- Hoare 2-way partitioning

Algorithms

- Dijkstra 3-way partitioning
- Bentley-Mcllroy 3-way partitioning
- dual-pivot partitioning

Robert Sedgewick I Kevin Wayne
https://algs4.cs.princeton.edu

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | A | B | X | w | P | P | V | P | D | P | C | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 10 |  |  | $\uparrow$ |  |  |  |  |  |  |  |  | ${ }_{\text {¢ }}{ }^{\text {d }}$ | $\xrightarrow{\uparrow} \mathrm{hi}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | A | B | X | w | P | P | v | P | D | P | C | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ 10 |  |  | $\uparrow$ |  |  |  |  |  |  |  | ${ }_{\text {¢ }}{ }^{\text {j }}$ |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | A | B | C | w | P | P | V | P | D | P | X | Y | z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 10 |  |  | $\uparrow$ |  |  |  |  |  |  |  | ${ }_{\text {¢ }}{ }^{\text {j }}$ |  | $\xrightarrow{\uparrow}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | A | B | C | w | P | P | V | P | D | P | x | Y | z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ 10 |  |  |  | $\uparrow$ |  |  |  |  |  |  | ${ }_{\text {¢ }}{ }^{\text {j }}$ |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | A | B | C | w | P | P | v | P | D | P | x | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ 10 |  |  |  | $\uparrow$ |  |  |  |  |  |  |  |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | A | B | C | P | P | P | v | P | D | w | X | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ 10 |  |  |  | $\uparrow$ |  |  |  |  |  | ${ }_{\text {¢ }}{ }^{\text {j }}$ |  |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | P | B | C | A | P | P | v | P | D | w | x | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ 10 |  |  |  | $\uparrow$ |  |  |  |  |  | ${ }_{\text {¢ }}{ }^{\text {j }}$ |  |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | P | B | C | A | P | P | V | P | D | W | X | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ¢ lo |  |  |  |  | $\uparrow$ |  |  |  |  | 个 j |  |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.

| P | P | B | C | A | P | P | v | P | D | w | X | Y | Z |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ¢ I |  |  |  |  | $\uparrow$ |  |  |  | j |  |  |  | $\stackrel{\uparrow}{\text { hi }}$ |

## 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, 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] < $a[1 o]$ ).
- Scan j from right to left so long as (a[j] > a[lo]).
- Exchange a[i] with a[j].
- If (a[i] == $a[1 o])$, exchange $a[i]$ with $a[p]$ and increment $p$.
- If $(a[j]==a[1 o])$, exchange $a[j]$ with $a[q]$ and decrement $q$.



## Bentley-Mcllroy 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].



## Bentley-Mcllroy 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].



## Bentley-Mcllroy 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].



## Bentley-Mcllroy 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].



## Bentley-Mcllroy 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].



## Bentley-Mcllroy 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-Mellroy 3-way pártitioning
- dual-pivot partitioning

Robert Sedgewick I Kevin Wayne
https://algs4.cs.princeton.edu

## Dual-pivot partitioning demo

Initialization.

- Choose a[1o] and a[hi] as partitioning items.
- Exchange if necessary to ensure $a[1 o] \leq a[h i]$.

exchange a[lo] and a[hi]


## Dual-pivot partitioning demo

Initialization.

- Choose a[1o] and a[hi] as partitioning items.
- Exchange if necessary to ensure $a[1 o] \leq a[h i]$.



## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

exchange $a[i]$ and $a[I t]$; increment It and $i$


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

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


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

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


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ 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 $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ 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 $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

| $\mathrm{p}_{1}$ |  |  | $\mathrm{p}_{1} \leq$ and $\leq \mathrm{p}_{2}$ |  |  |  | $?$ |  |  |  |  |  | $\mathrm{p}_{2}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & \uparrow \\ & \text { lo } \end{aligned}$ | $\uparrow$ |  |  |  |  |  | $\uparrow$ |  |  |  |  |  | $\begin{gathered} \uparrow \\ \mathrm{hi} \end{gathered}$ |
| K | E | A | M | R | L | F | V | Z | Q | T | C | Y | S |
| $\uparrow$ | I |  |  |  | $\uparrow$ |  |  |  |  |  | $\begin{gathered} \uparrow \\ \mathrm{gt} \end{gathered}$ |  | $\uparrow$ |
| lo |  |  |  |  |  |  |  |  |  |  |  |  | hi |

increment i

## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

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


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

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


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

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


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ 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 $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

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


## Dual-pivot partitioning demo

Main loop. Repeat until i and gt pointers cross.

- If $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ 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 $\quad(a[i]<a[7 o])$, exchange $a[i]$ with $a[7 t]$ and increment $1 t$ and $i$.
- Else if (a[i] > a[hi]), exchange a[i] with a[gt] and decrement gt.
- Else, increment i.

| $\mathrm{p}_{1}$ |  |  | $\mathrm{p}_{1} \leq$ and $\leq \mathrm{p}_{2}$ |  |  |  | ? |  |  |  |  |  | $\mathrm{p}_{2}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & \uparrow \\ & \text { lo } \end{aligned}$ | $\begin{aligned} & \uparrow \\ & \text { It } \end{aligned}$ |  |  |  |  |  | $\uparrow$ |  |  |  |  |  | $\begin{gathered} \uparrow \\ \mathrm{hi} \end{gathered}$ |
| K | E | A | F | C | L | M | R | Q | T | Z | V | Y | S |
| $\begin{aligned} & \uparrow \\ & \text { lo } \end{aligned}$ |  |  | $\begin{aligned} & \uparrow \\ & \text { It } \end{aligned}$ |  |  |  | $\begin{gathered} \uparrow \\ \mathrm{gt} \end{gathered}$ |  | $\uparrow$ |  |  |  | ¢ ${ }_{\text {¢ }}^{\text {hi }}$ |

## Dual-pivot partitioning demo

Finalize.

- Exchange a[1o] with a[--7t].
- Exchange a[hi] with a[++gt].

| $\mathrm{p}_{1}<\mathrm{p}_{1}$ | $\mathrm{p}_{1} \leq$ and $\leq \mathrm{p}_{2}$ | $>\mathrm{p}_{2}$ | $\mathrm{p}_{2}$ |  |
| :---: | :---: | :---: | :---: | :---: |
| $\uparrow$ | $\uparrow$ | $\uparrow$ |  | 个 |
| lo | lt | gt | hi |  |


| K | $E$ | $A$ | $F$ | $C$ | $L$ | $M$ | $R$ | $Q$ | $T$ | $Z$ | $V$ | $Y$ | $S$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## Dual-pivot partitioning demo

Finalize.

- Exchange a[lo] with a[--7t].
- Exchange a[hi] with a[++gt].

|  | $<\mathrm{p}_{1}$ |  |  | $\mathrm{p}_{1}$ | $\mathrm{p}_{1} \leq$ and $\leq \mathrm{p}_{2}$ |  |  |  | $\mathrm{p}_{2}$ |  | $>\mathrm{p}_{2}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{gathered} \uparrow \\ \text { 10 } \end{gathered}$ |  |  |  | $\uparrow$ lt |  |  |  |  | $\begin{gathered} \uparrow \\ \mathrm{gt} \end{gathered}$ |  |  |  | $\stackrel{\uparrow}{\text { hi }}$ |
| c | E | A | F | K | L | M | R | Q | S | z | V | Y | T |
| ¢ lo |  |  |  | ¢ t |  |  |  |  | ¢ gt |  |  |  | $\uparrow{ }_{\text {hi }}$ |

