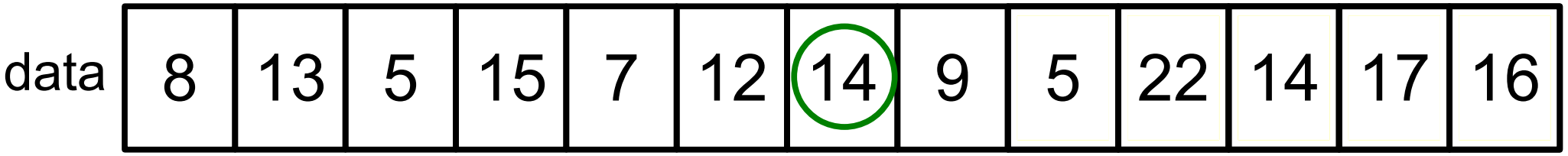
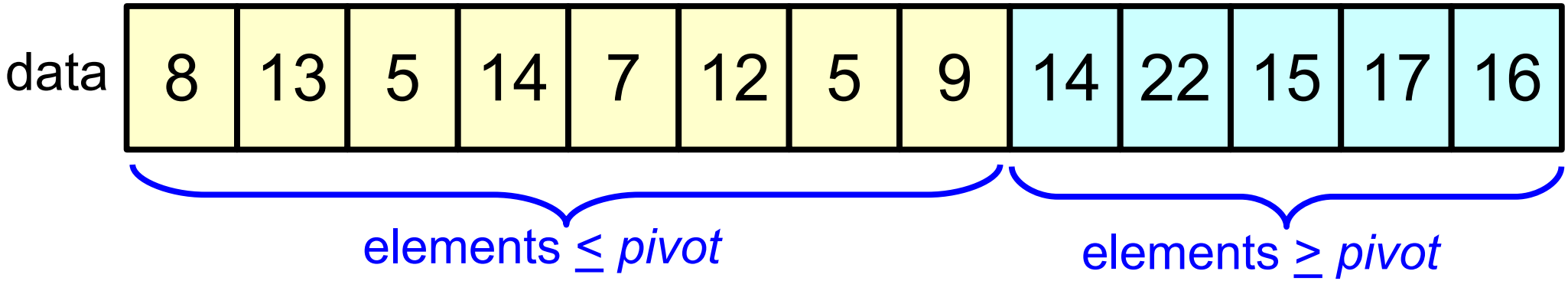
Quicksort

Choose a *pivot* element



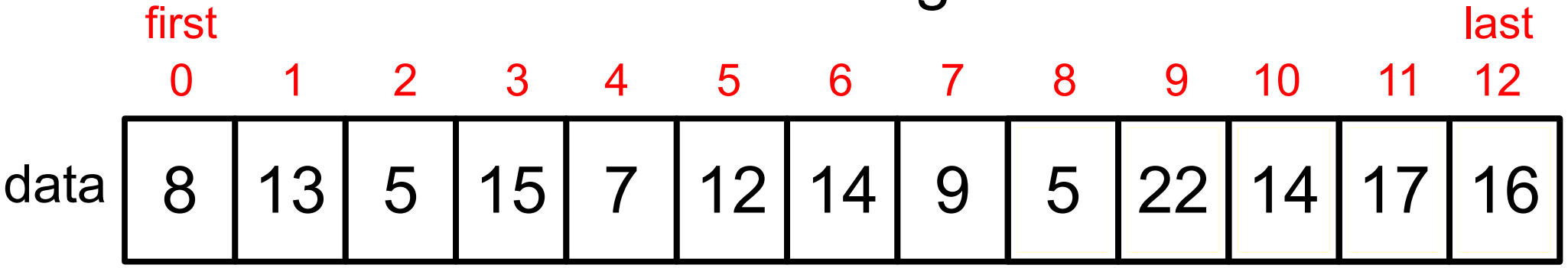
Rearrange the elements into two partitions



...and repeat for the partitions

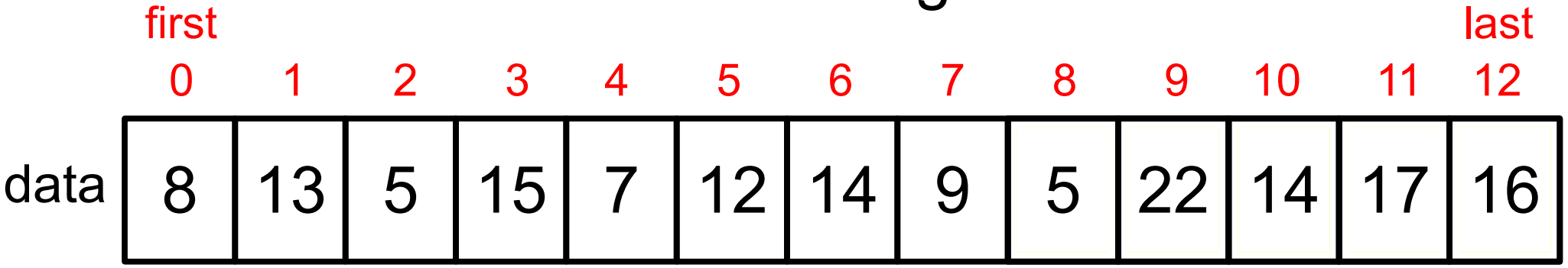
Partitioning Algorithm

Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

Partitioning



choose a *pivot* element

initialize left and right indices i and j

loop:

move left index i rightward until we encounter an element ≥ *pivot*

move right index j leftward until we encounter an element ≤ *pivot*

if indices have met or crossed, quit the loop

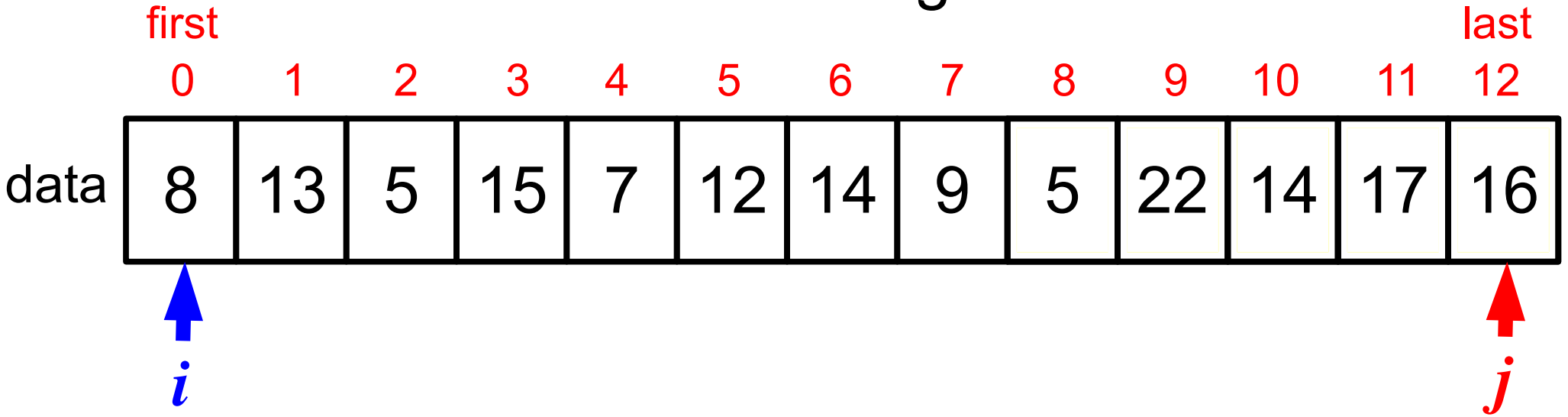
otherwise swap elements at positions i and j

move i to right and j to left one position, and continue loop

return j as the midpoint position of the partition

pivot element = 14

Partitioning



choose a *pivot* element

initialize left and right indices i and j

loop:

move left index i rightward until we encounter an element ≥ *pivot*

move right index j leftward until we encounter an element ≤ *pivot*

if indices have met or crossed, quit the loop

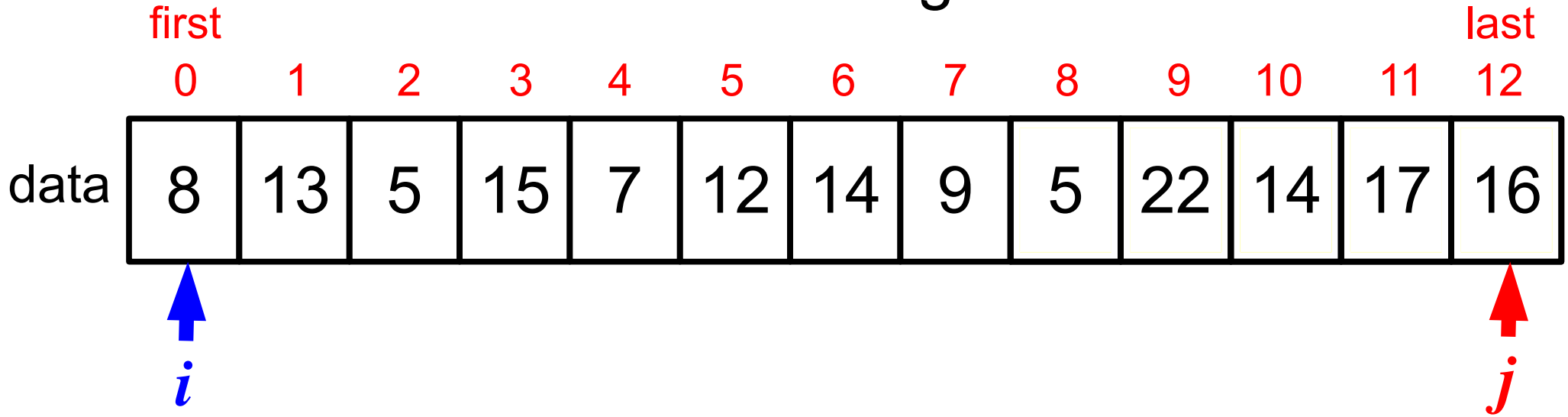
otherwise swap elements at positions i and j

move i to right and j to left one position, and continue loop

return j as the midpoint position of the partition

pivot element = 14

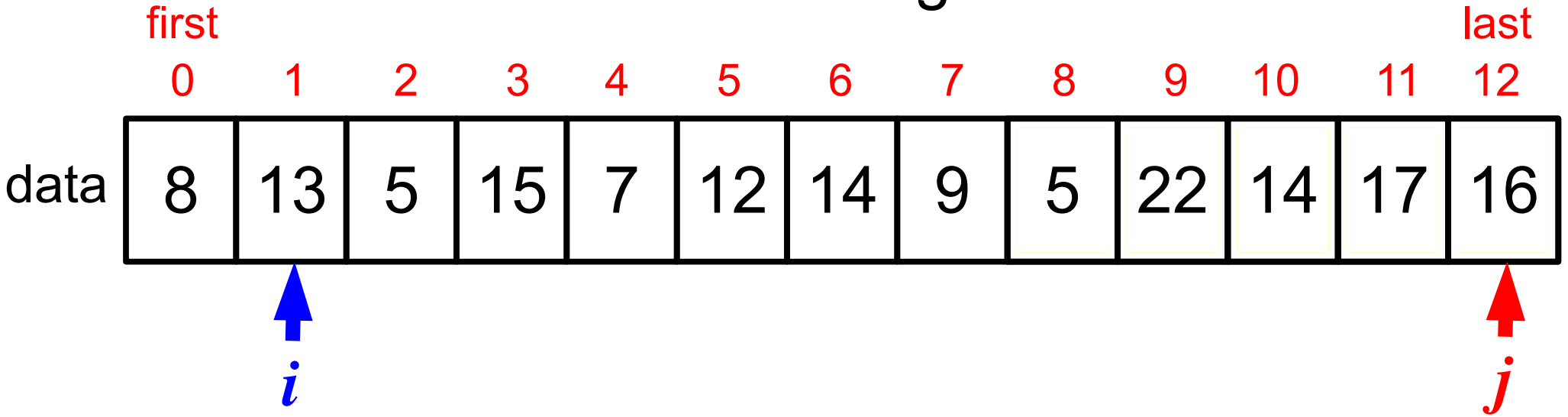
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

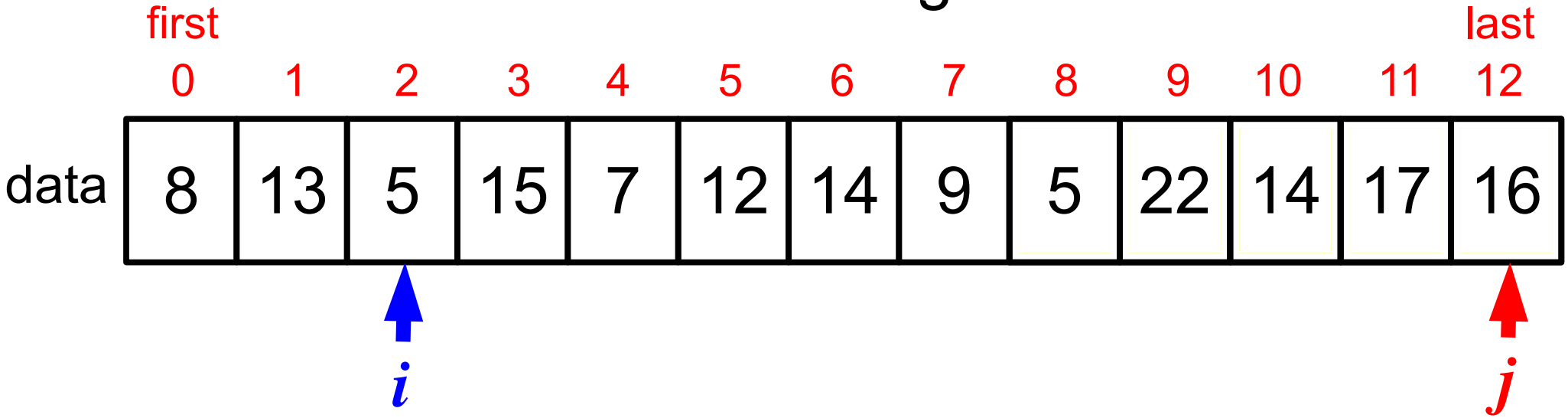
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

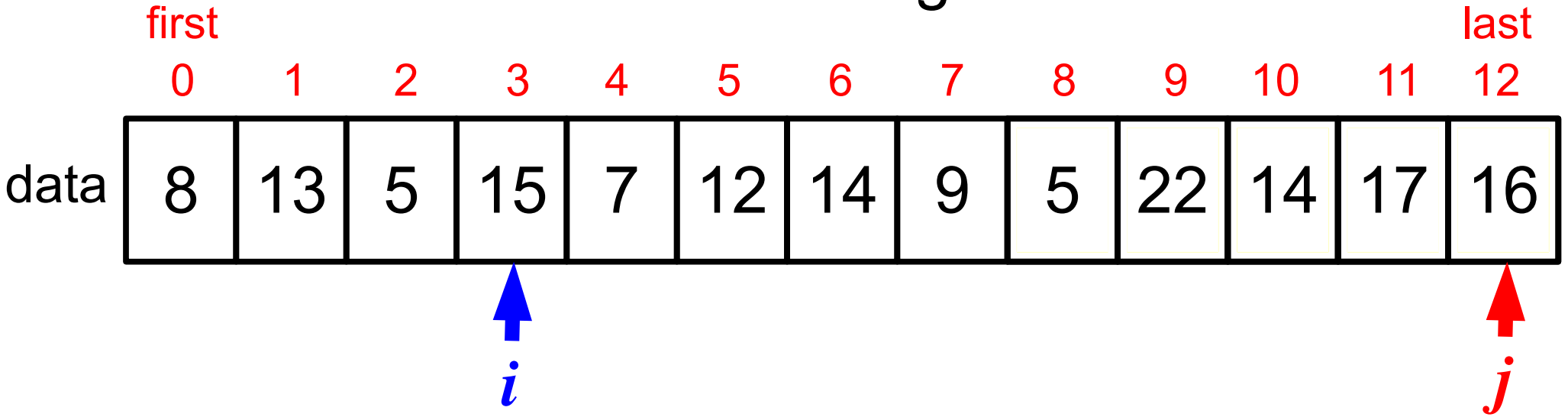
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

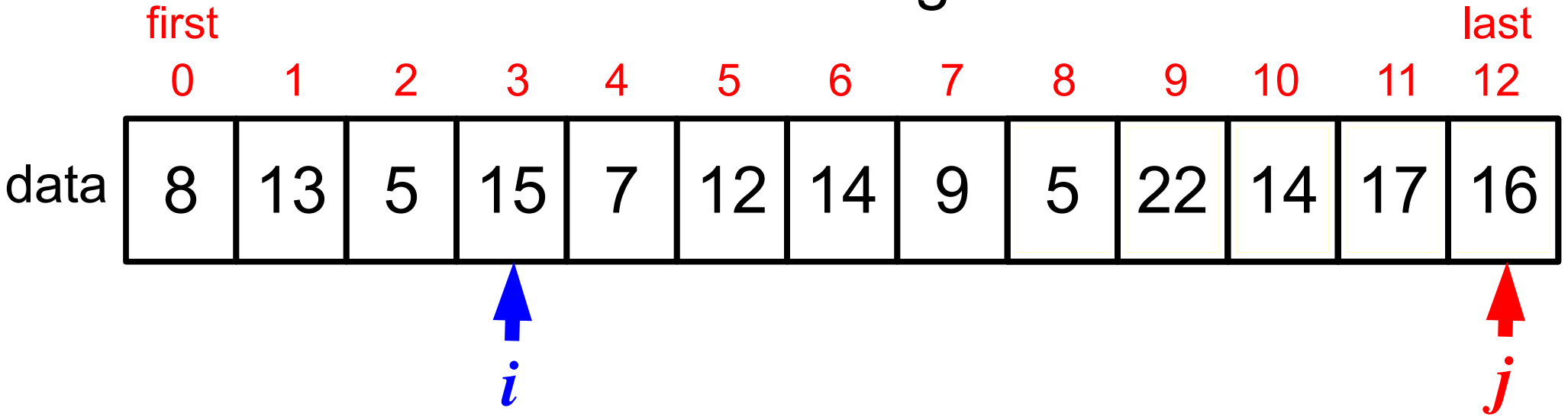
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

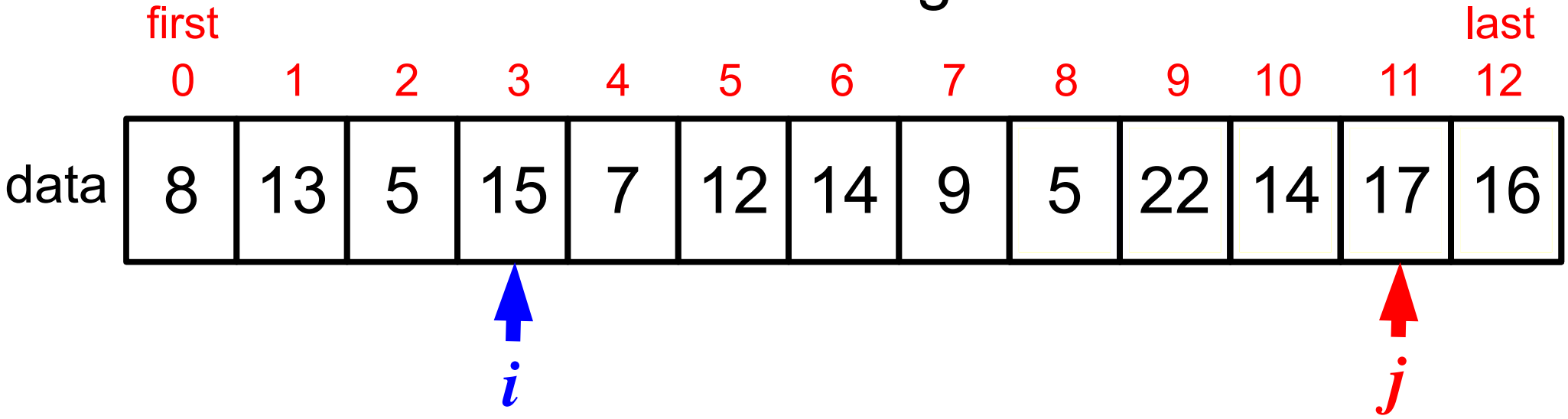
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

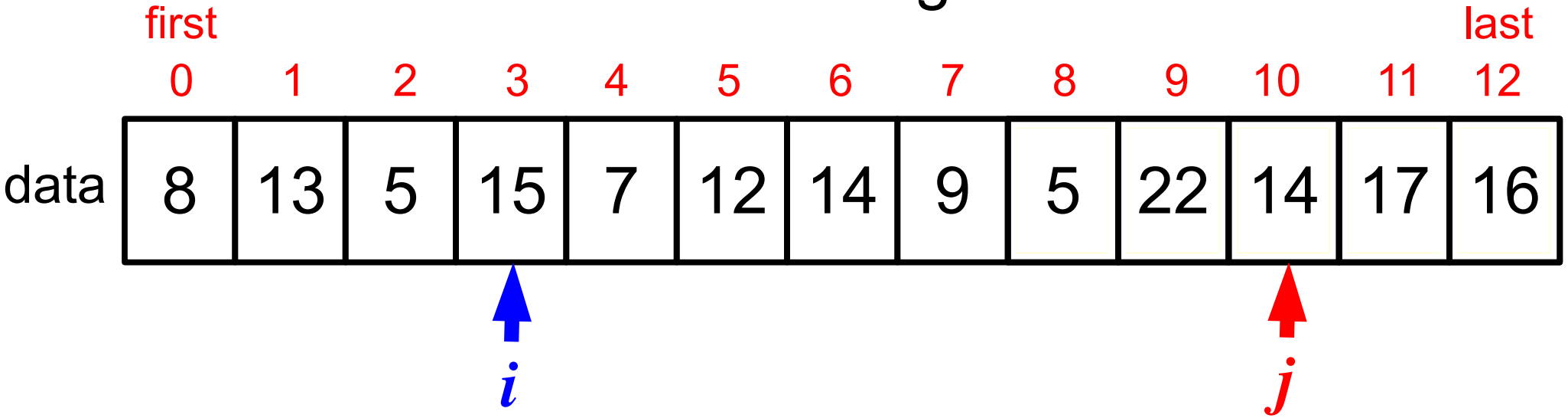
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

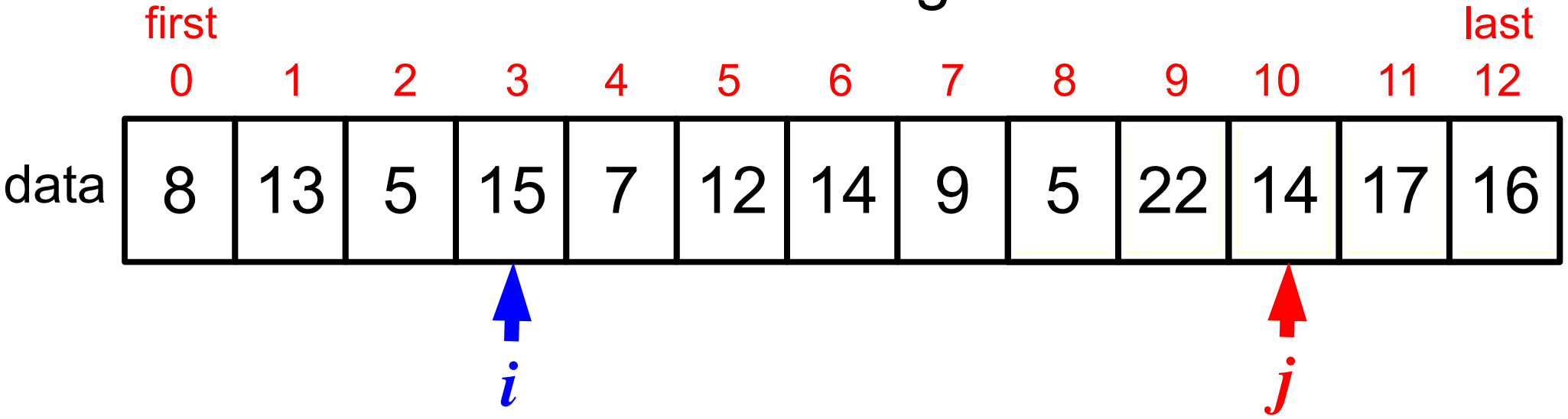
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

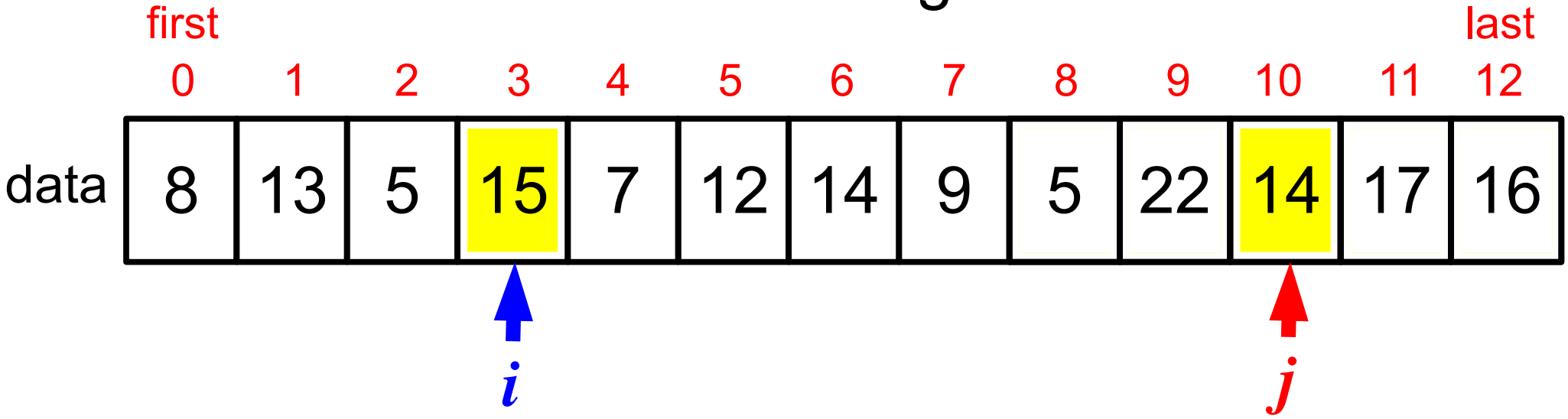
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

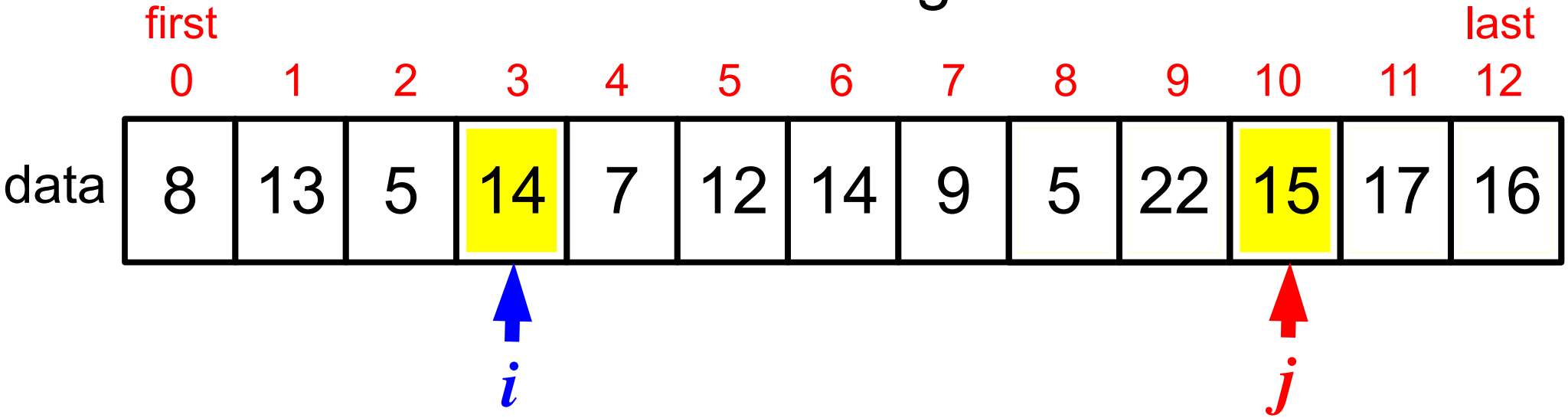
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

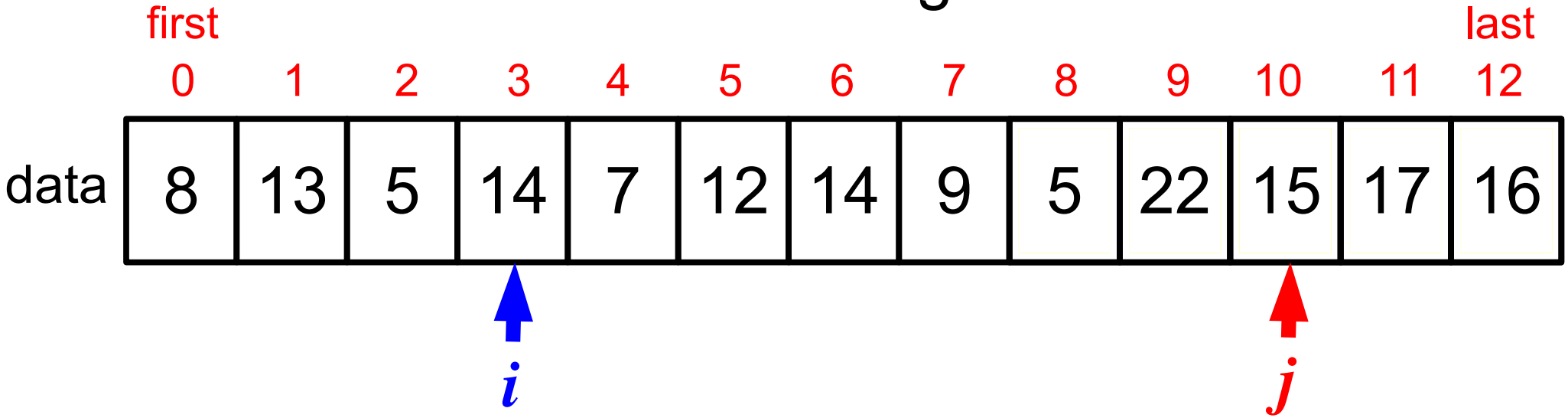
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

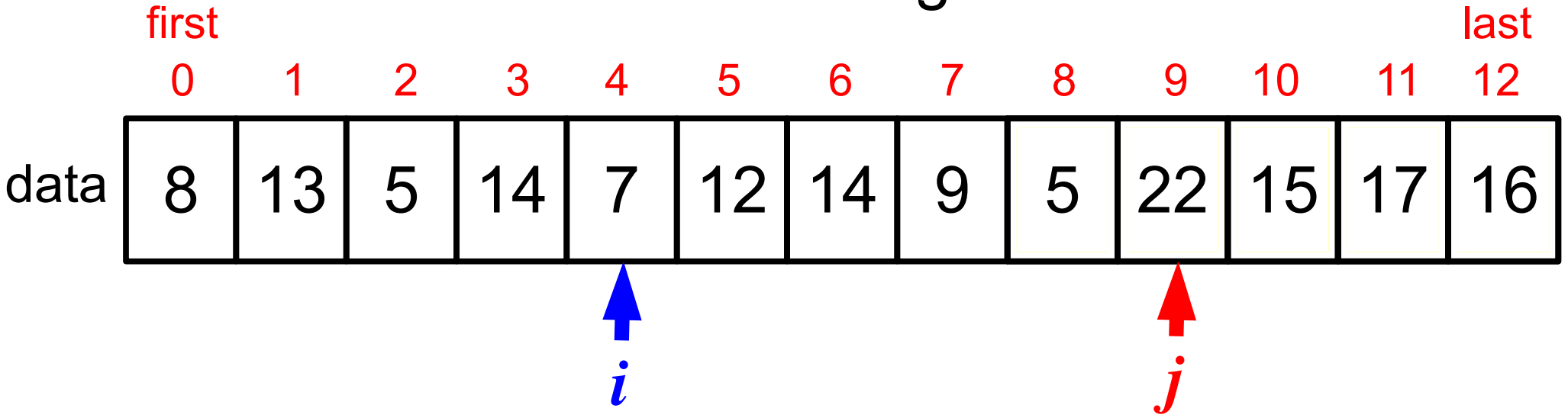
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

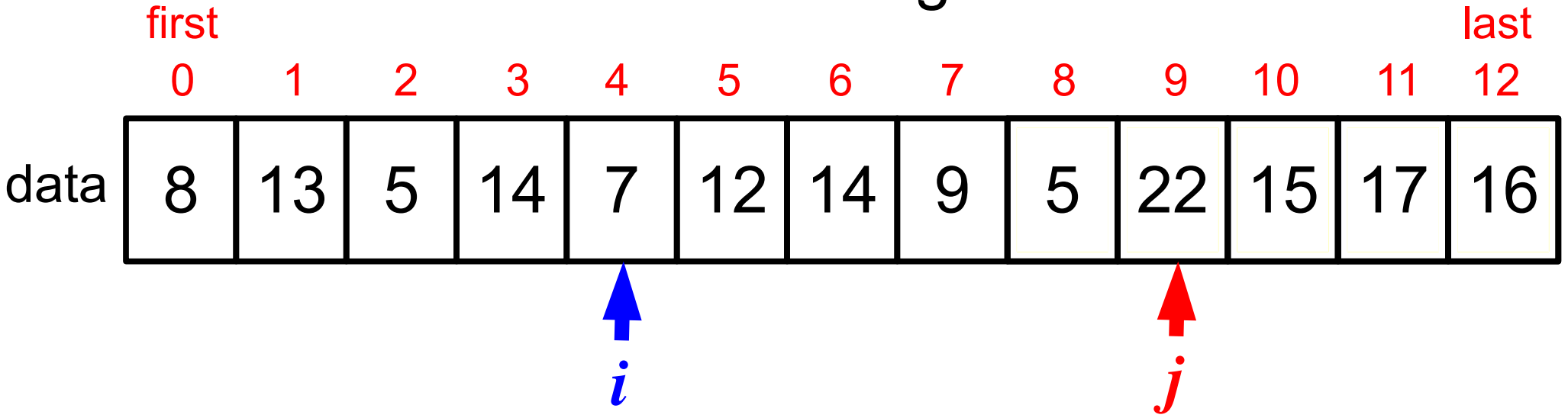
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

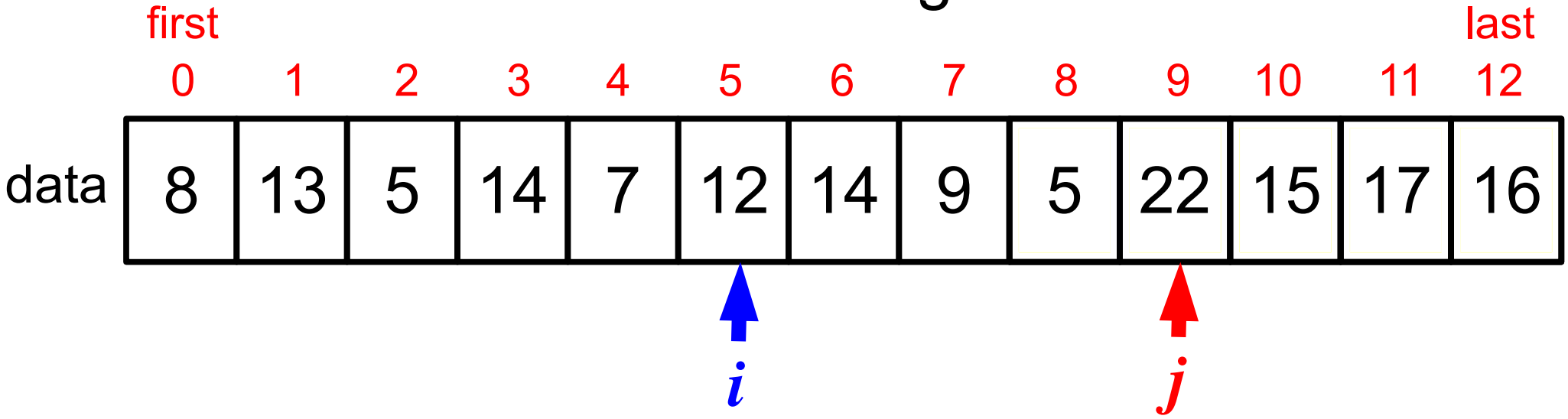
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

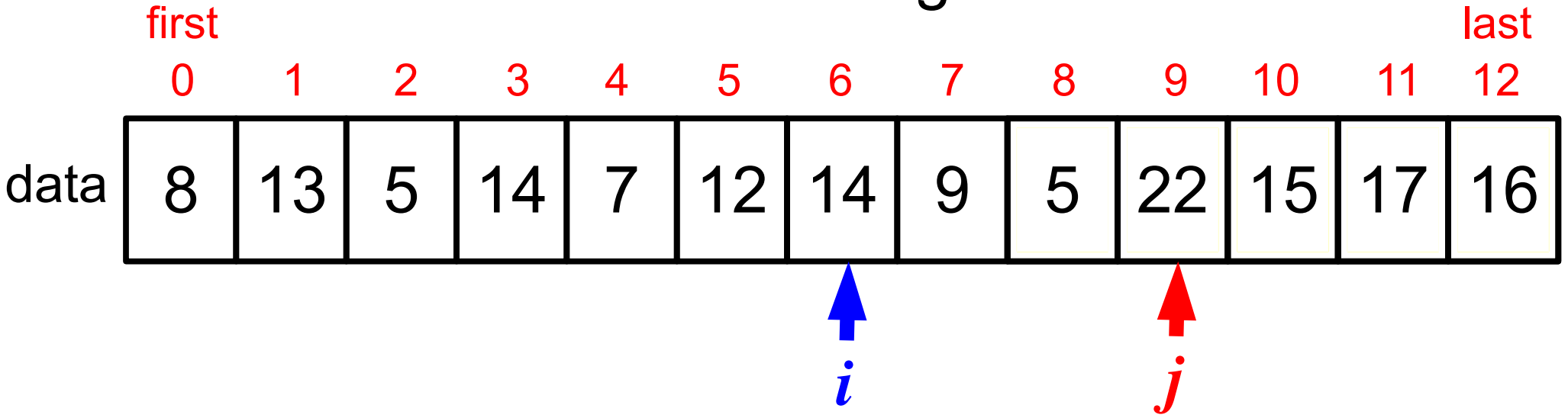
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

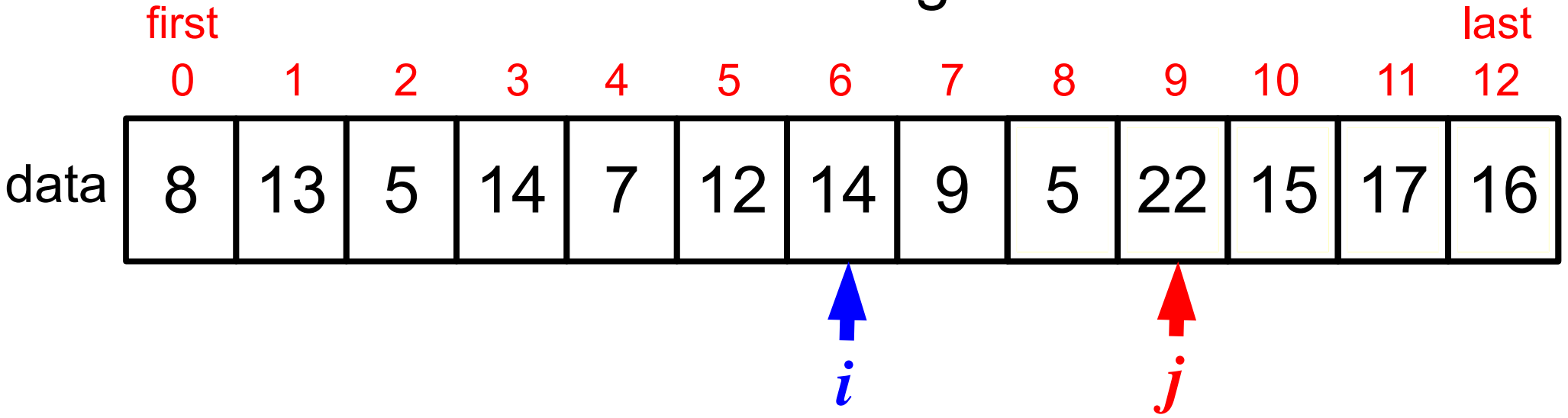
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

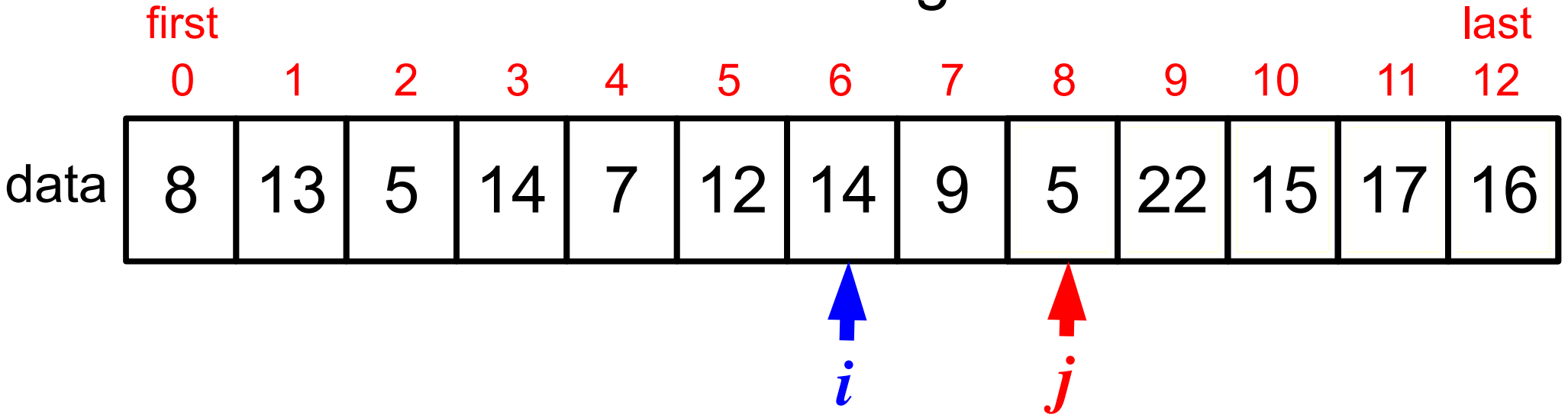
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

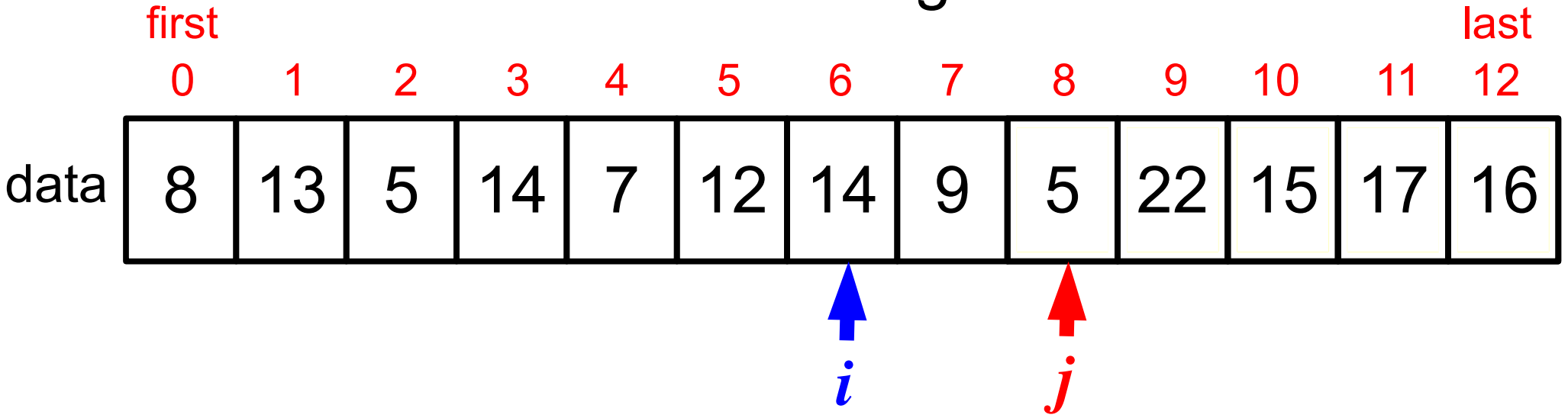
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

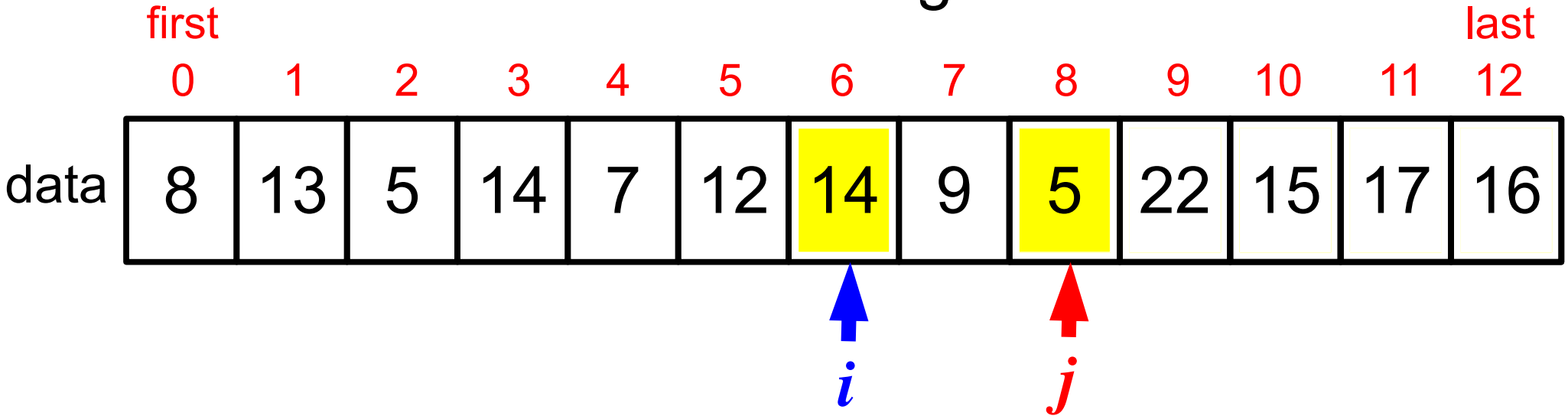
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

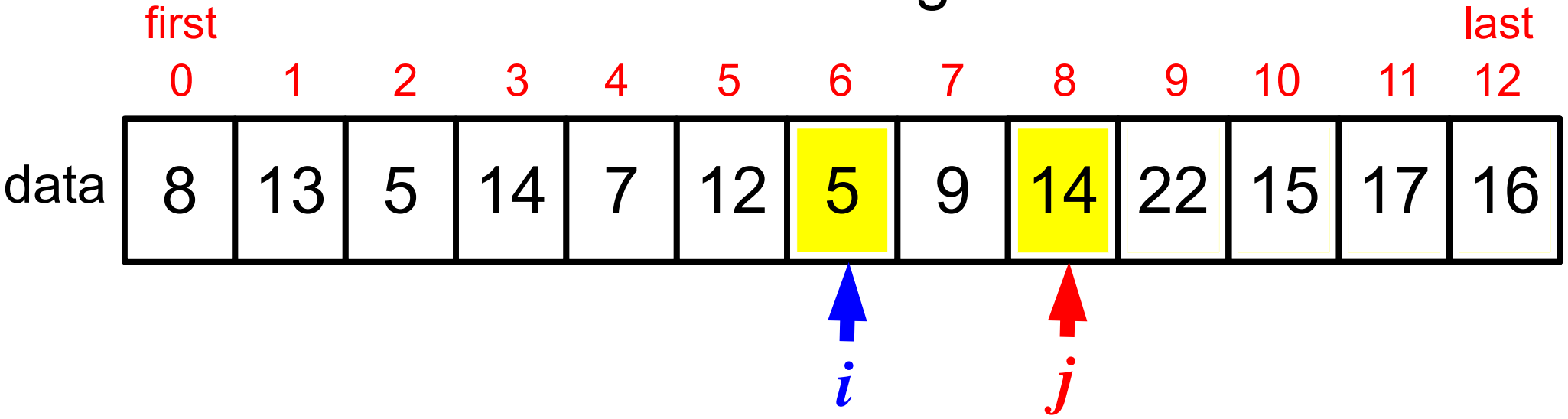
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

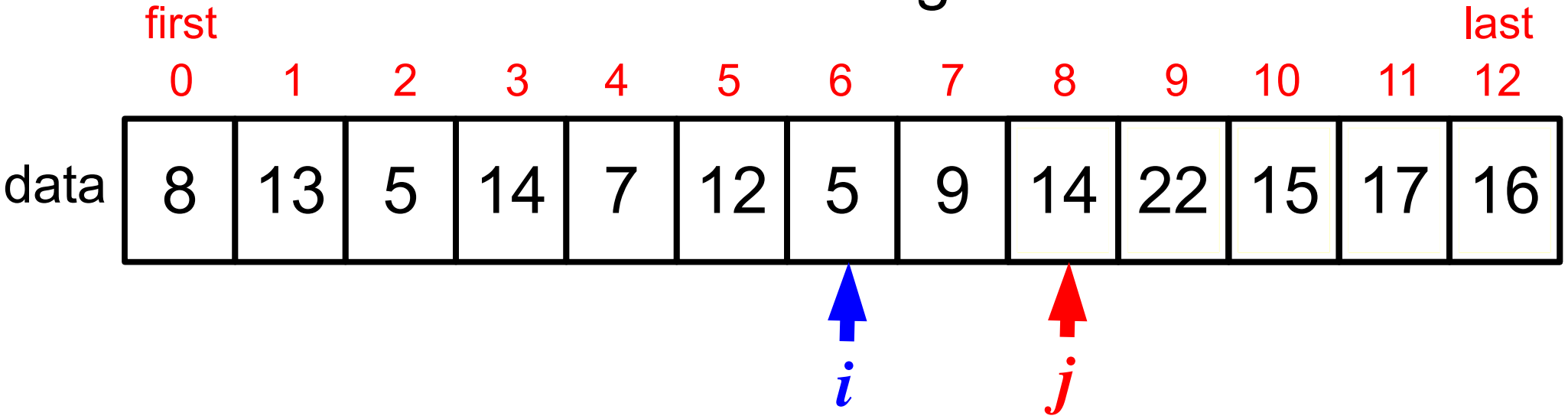
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

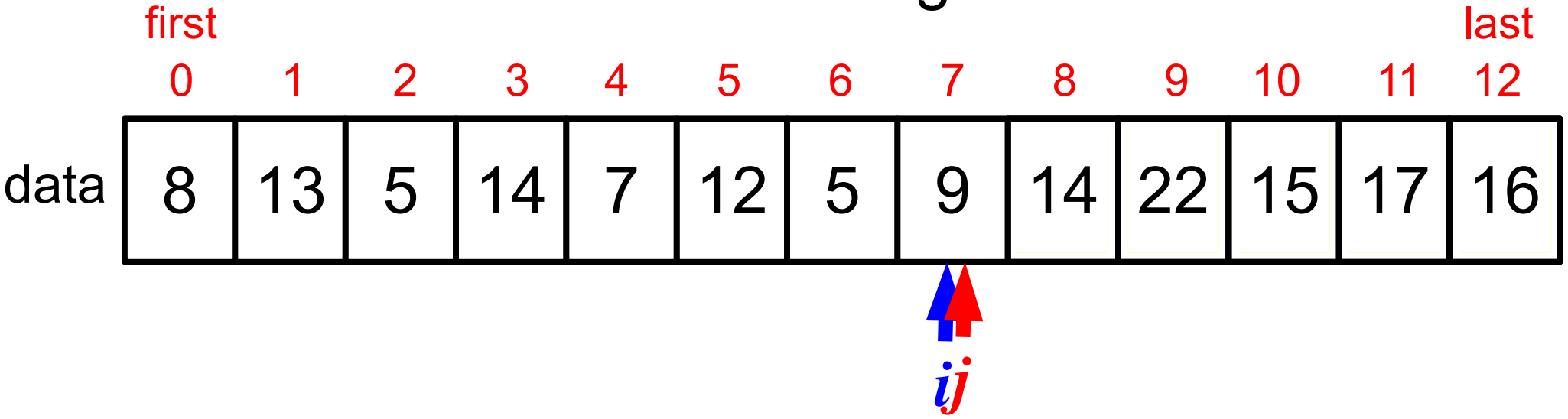
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

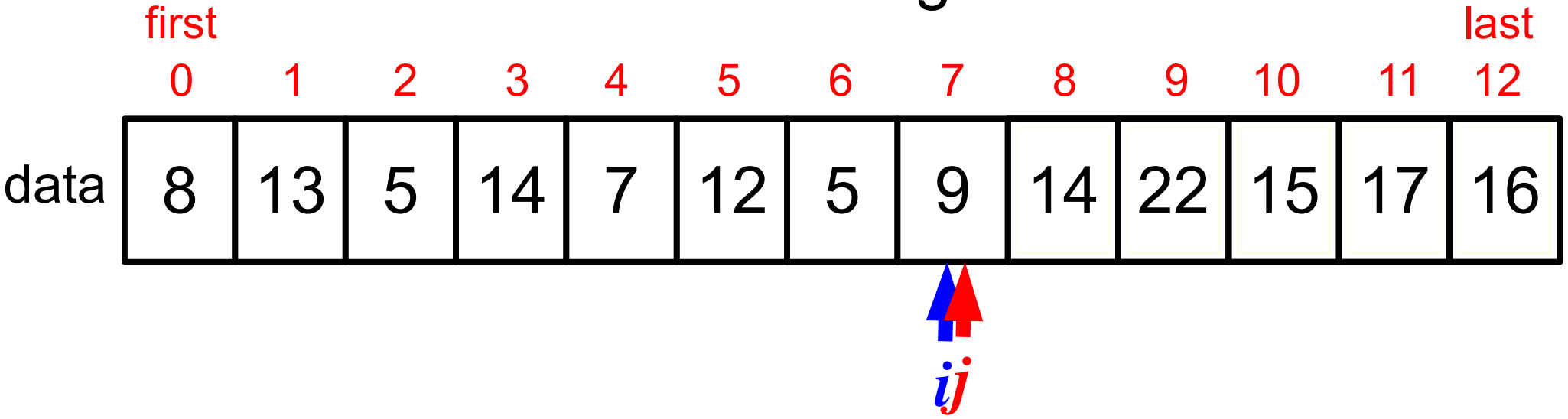
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

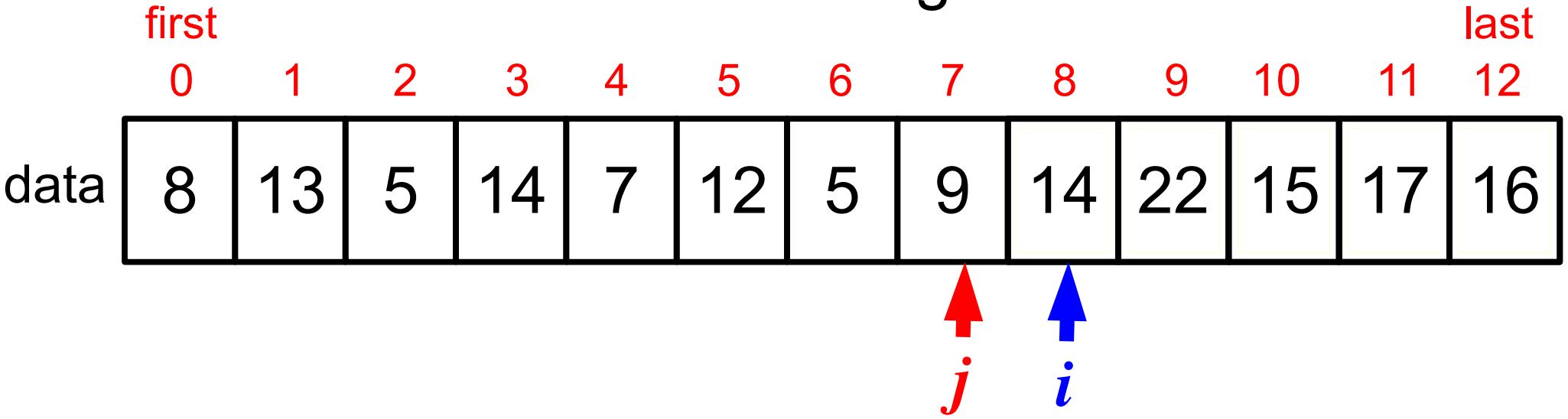
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

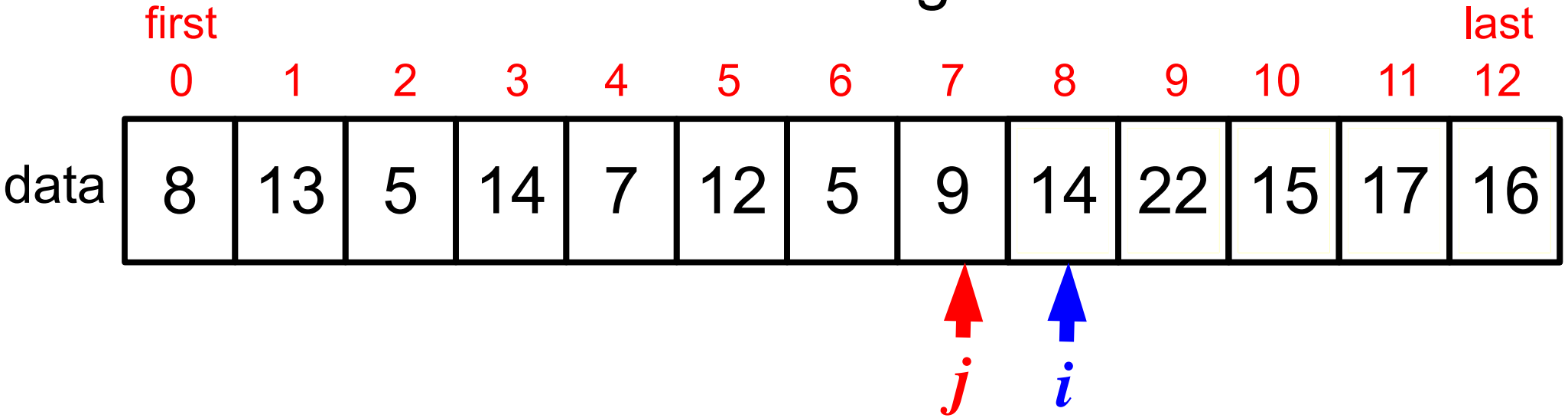
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

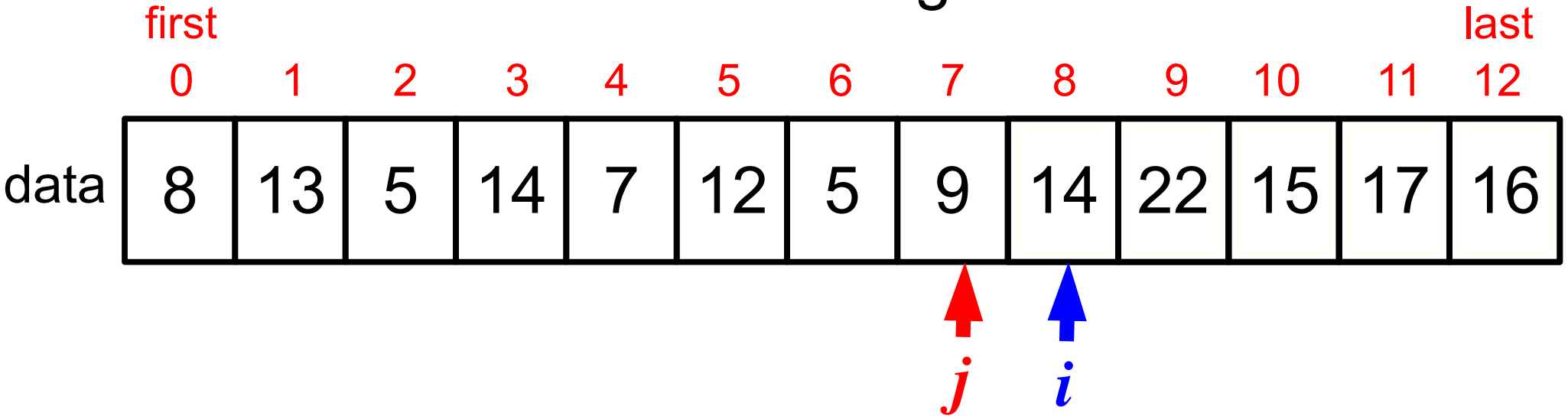
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

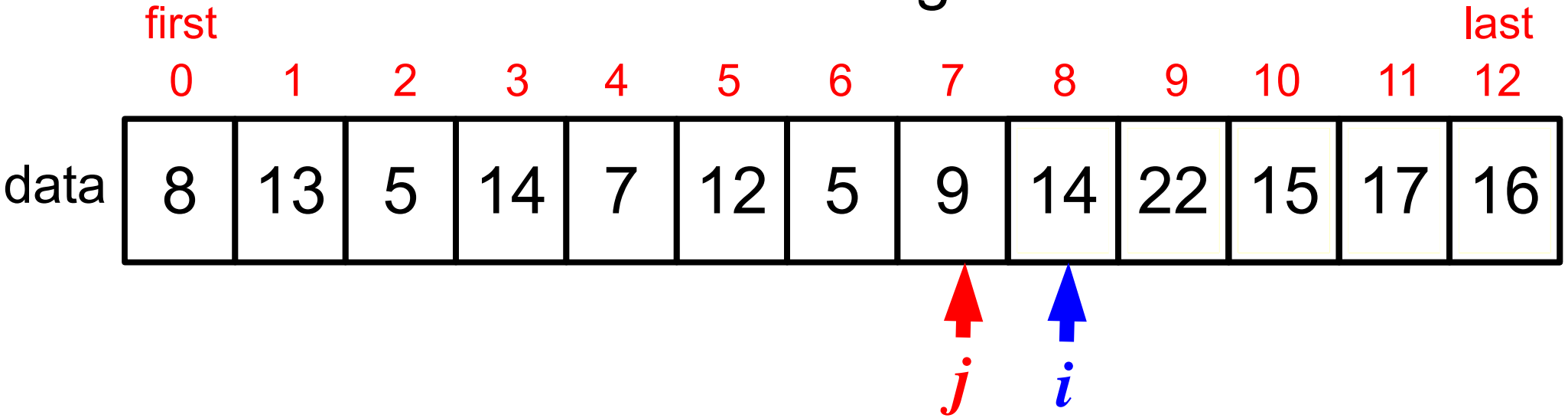
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

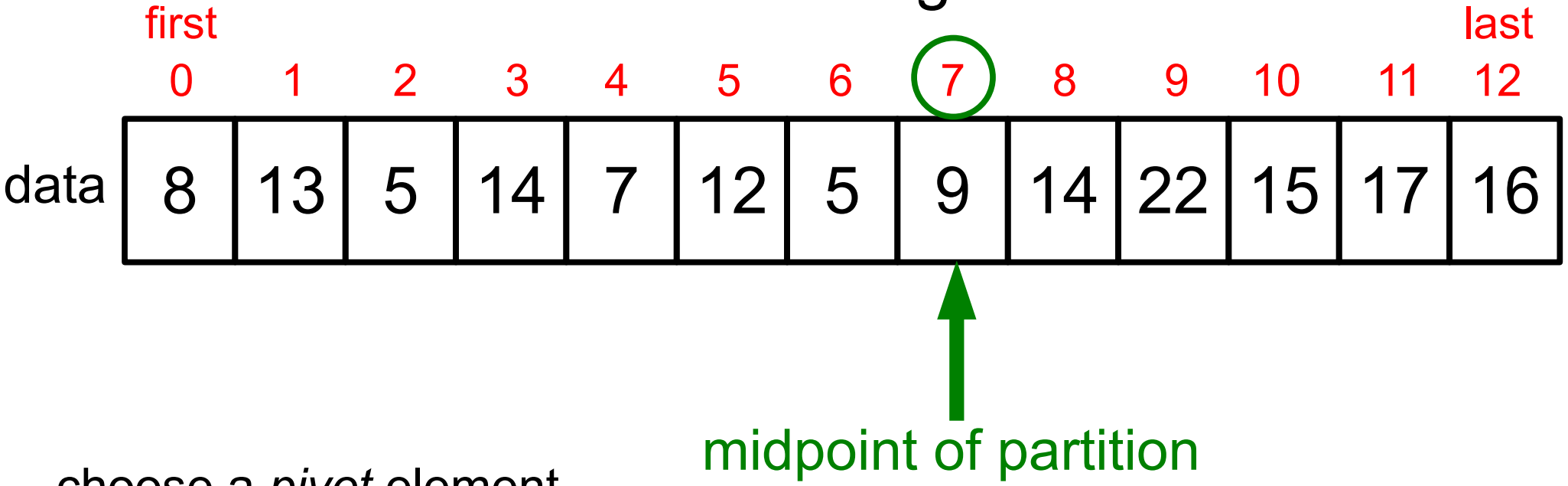
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

pivot element = 14

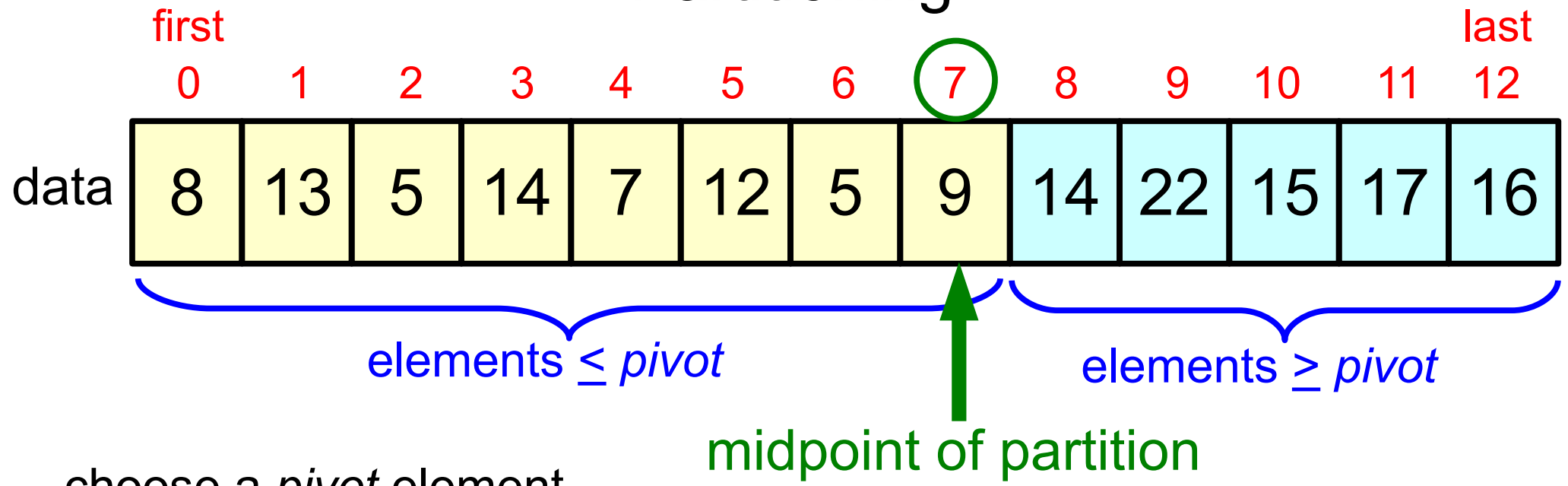
Partitioning



choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

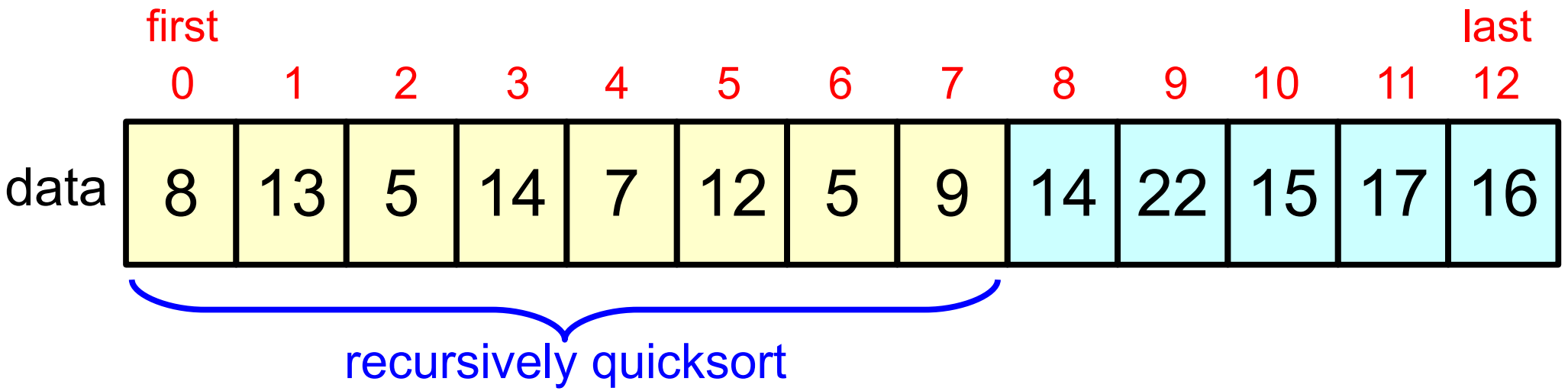
pivot element = 14

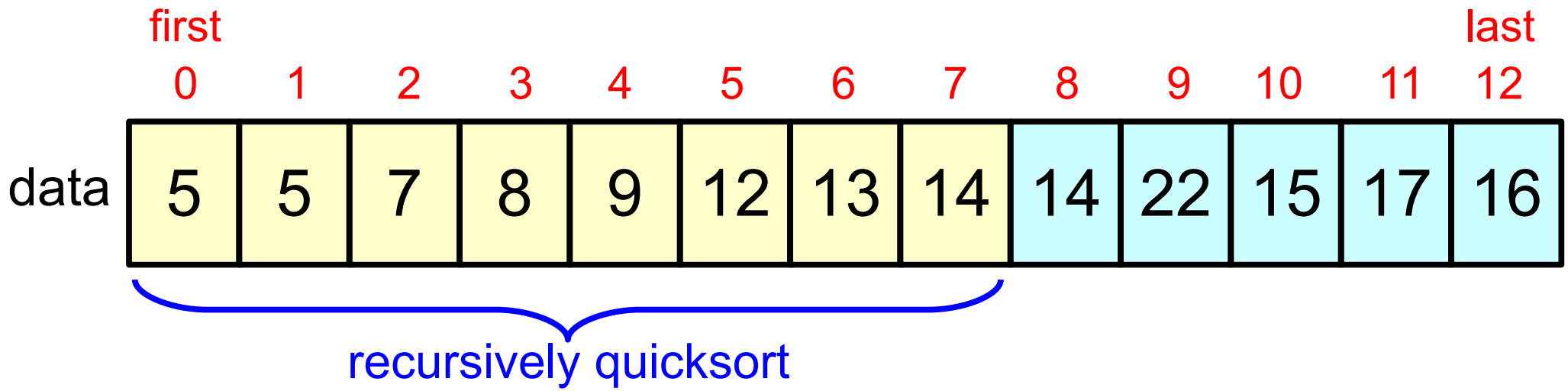
Partitioning

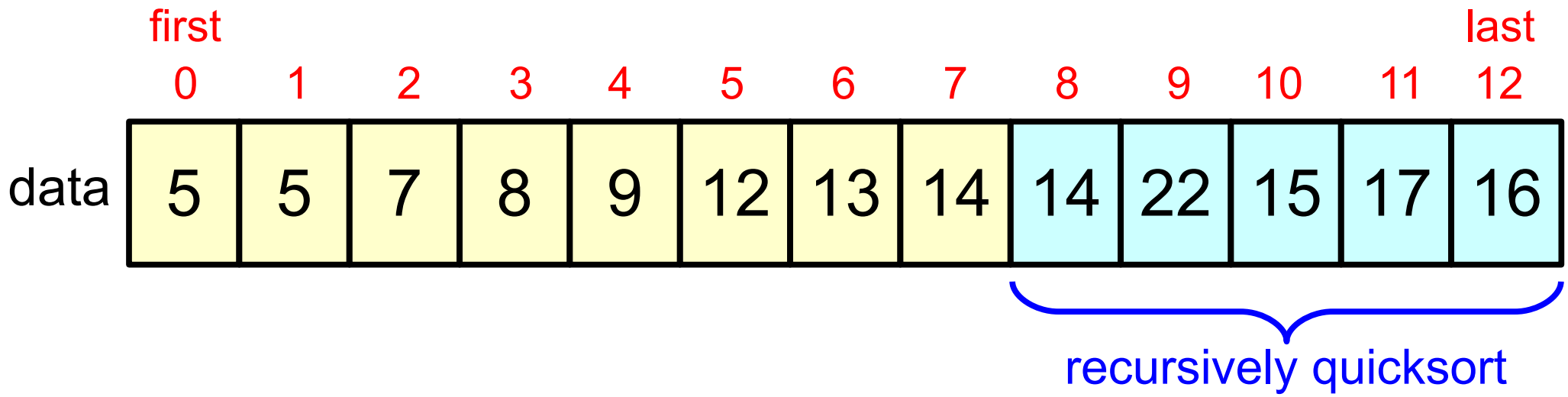


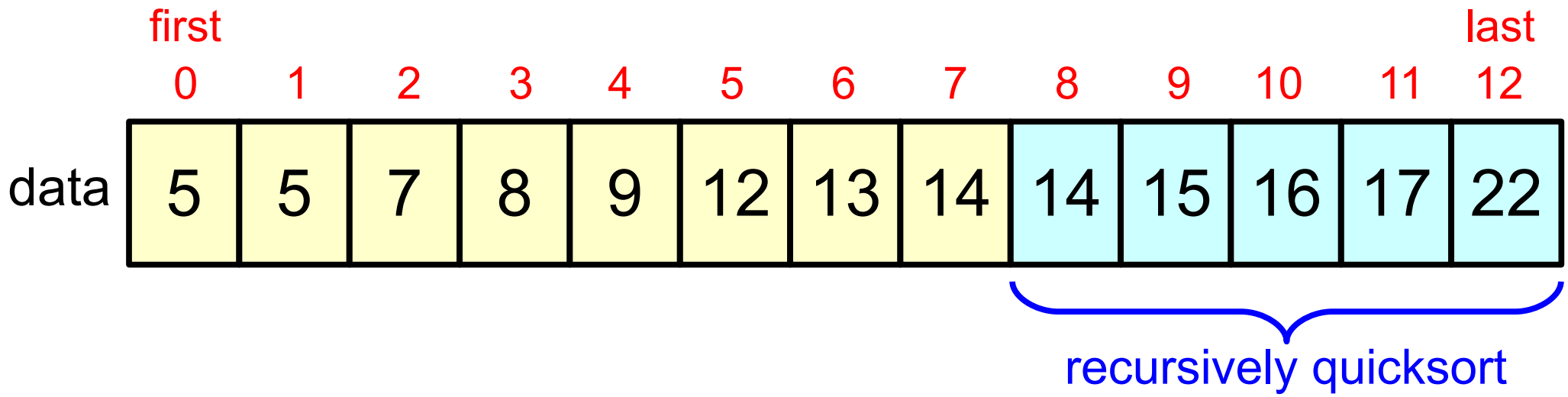
choose a *pivot* element
initialize left and right indices i and j
loop:
    move left index i rightward until we encounter an element ≥ *pivot*
    move right index j leftward until we encounter an element ≤ *pivot*
    if indices have met or crossed, quit the loop
    otherwise swap elements at positions i and j
    move i to right and j to left one position, and continue loop
return j as the midpoint position of the partition

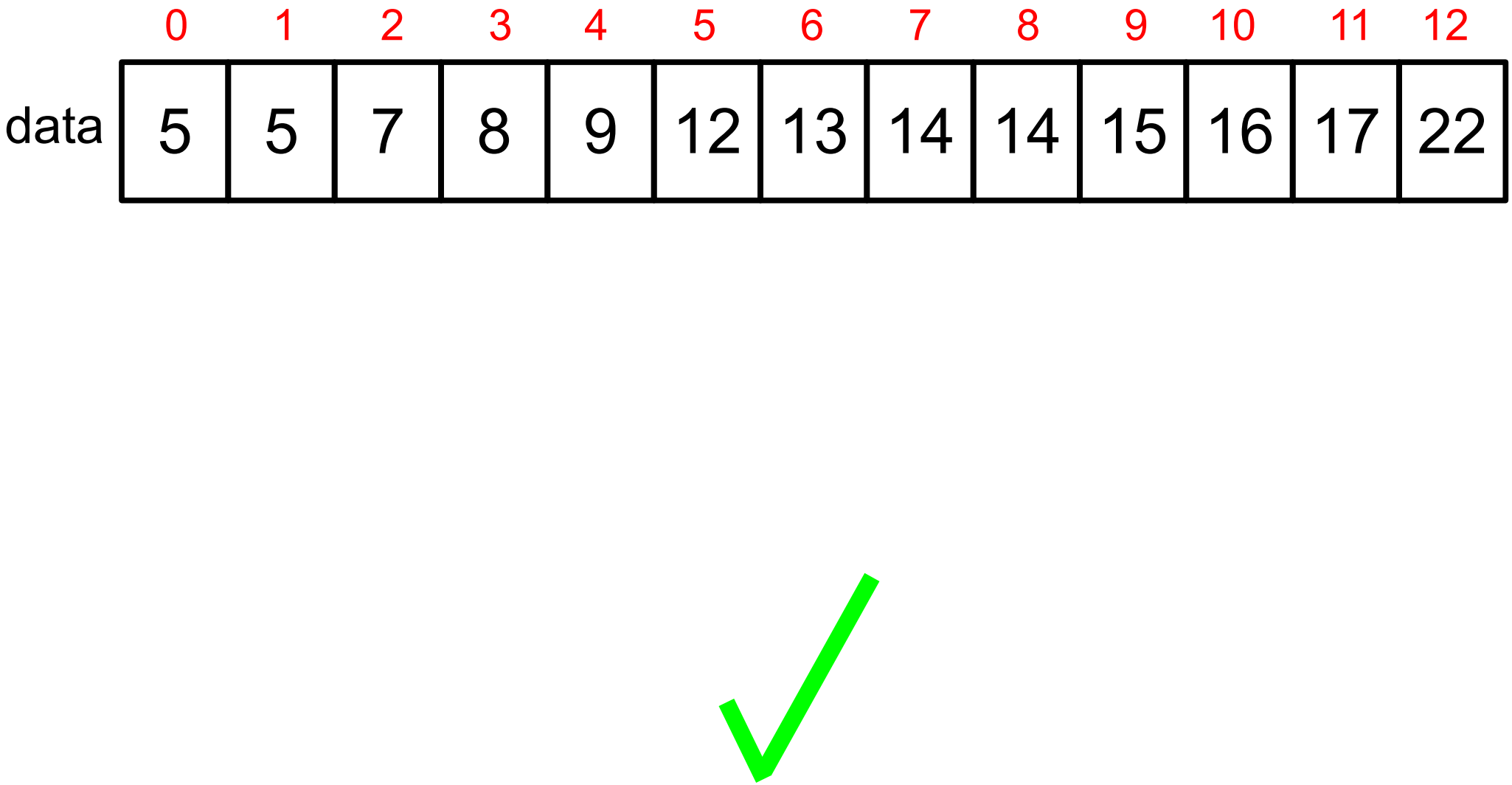
pivot element = 14


first
last
0
1
2
3
4
5
6
7
8
9
10
11
12
data
8
13
5
14
7
12
5
9
14
22
15
17
16
recursively quicksort


first
last
0 1 2 3 4 5 6 7 8 9 10 11 12
data
5 5 7 8 9 12 13 14 14 22 15 17 16
recursively quicksort


first
last
0
1
2
3
4
5
6
7
8
9
10
11
12
data
5
5
7
8
9
12
13
14
14
22
15
17
16
recursively quicksort


first
last
0
1
2
3
4
5
6
7
8
9
10
11
12
data
5
5
7
8
9
12
13
14
14
15
16
17
22
recursively quicksort


0 1 2 3 4 5 6 7 8 9 10 11 12
data
5 5 7 8 9 12 13 14 14 15 16 17 22