Quick Sort

TestLeaf
Always Ahead

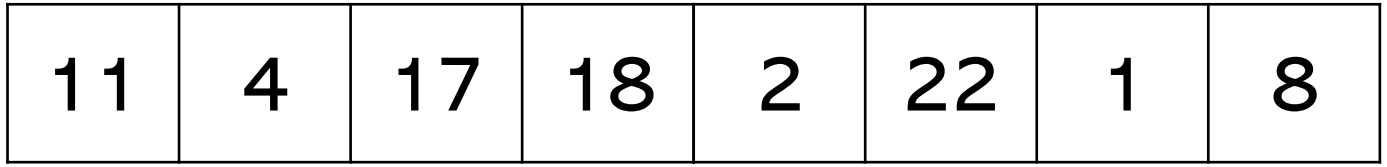
- Find an item in a sorted item [List or Array or String]
- Highly efficient in terms of time complexity against Linear Search
- Run time Complexity : O(log n)
- Based on divide and conquer technique

Example

Sort the given array

11	4	17	18	2	22	1	8

Partitioning



Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

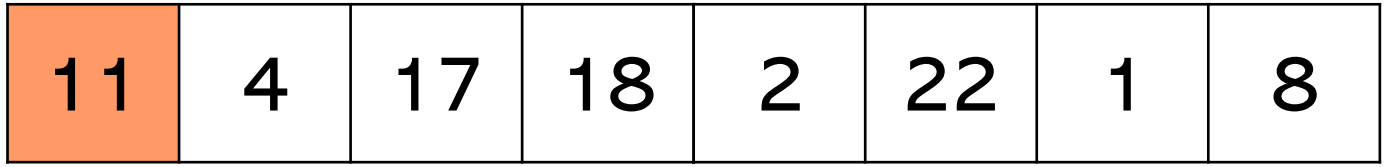
Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Step 6) Continue until left >= right

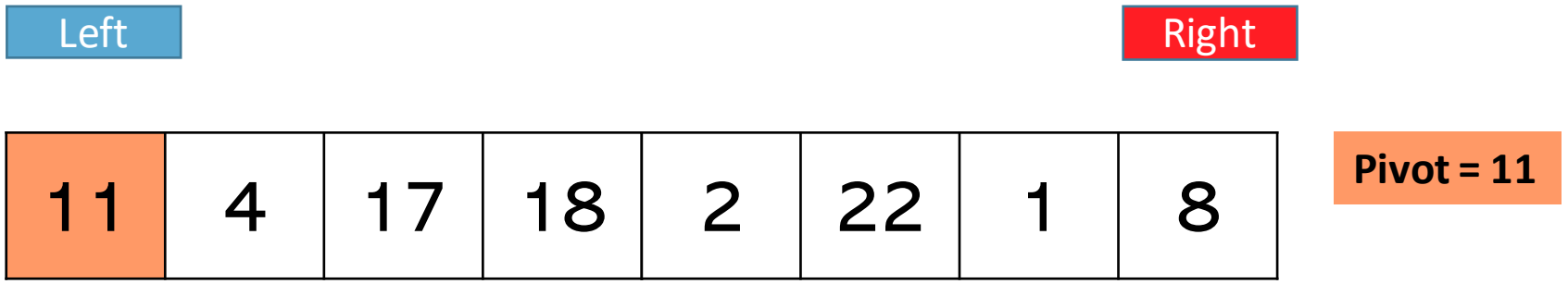
Partitioning



Pivot = 11

Step 1) Get the pivot (left most)

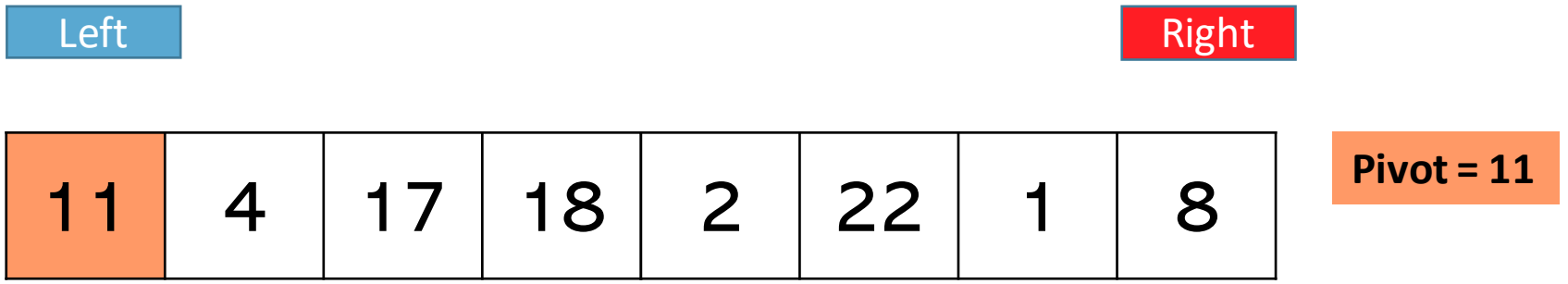
Partitioning



Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Partitioning

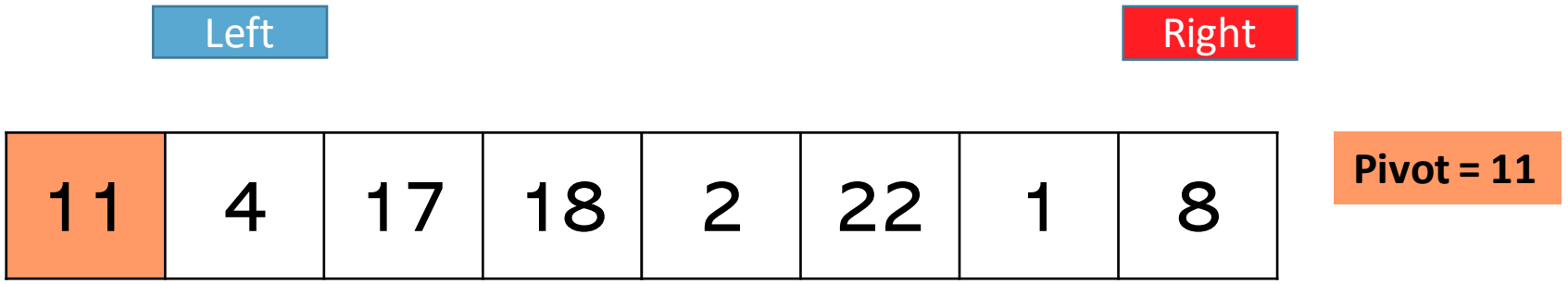


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning

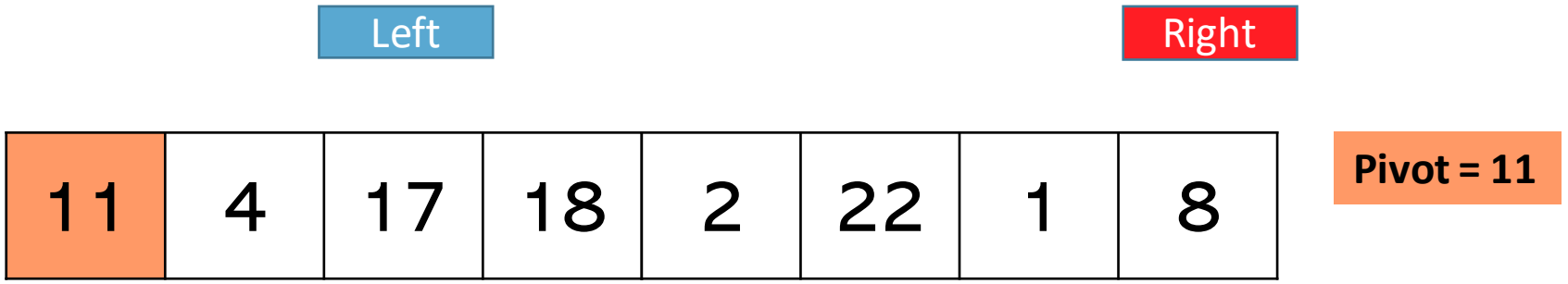


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning

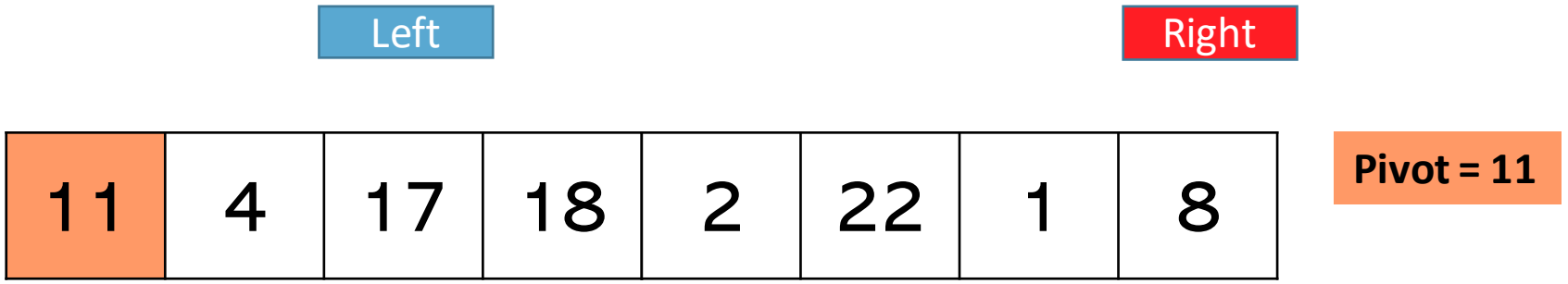


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning



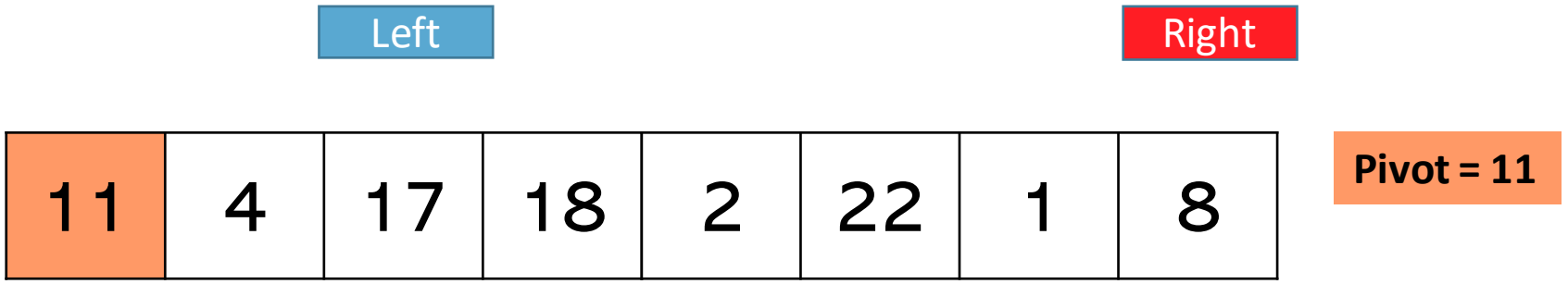
Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Partitioning



Step 1) Get the pivot (left most)

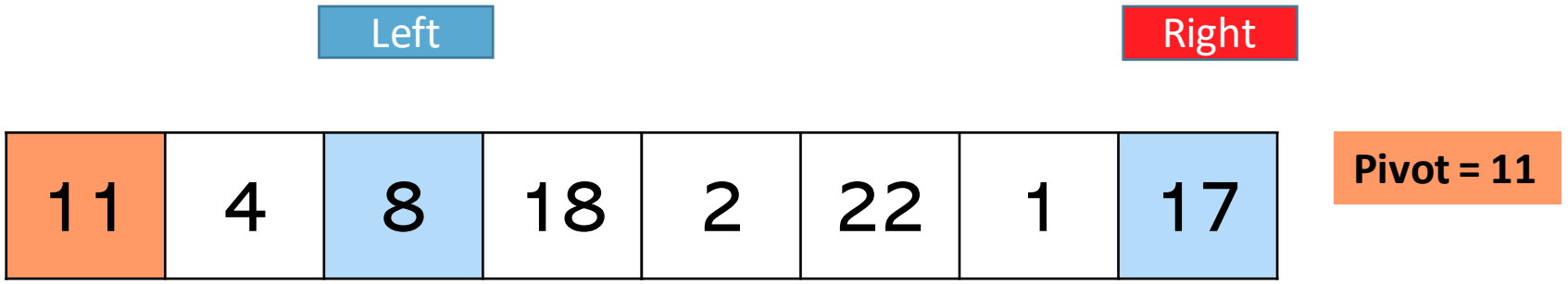
Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Partitioning



Step 1) Get the pivot (left most)

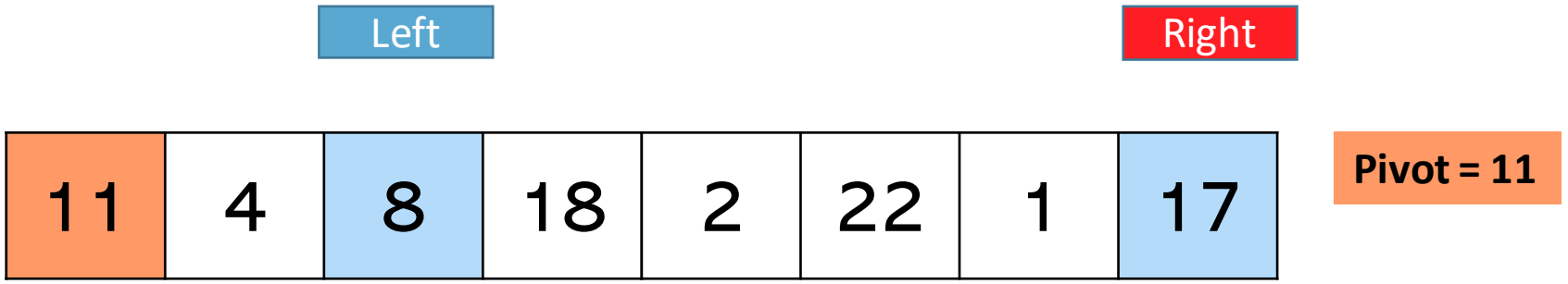
Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Partitioning



Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

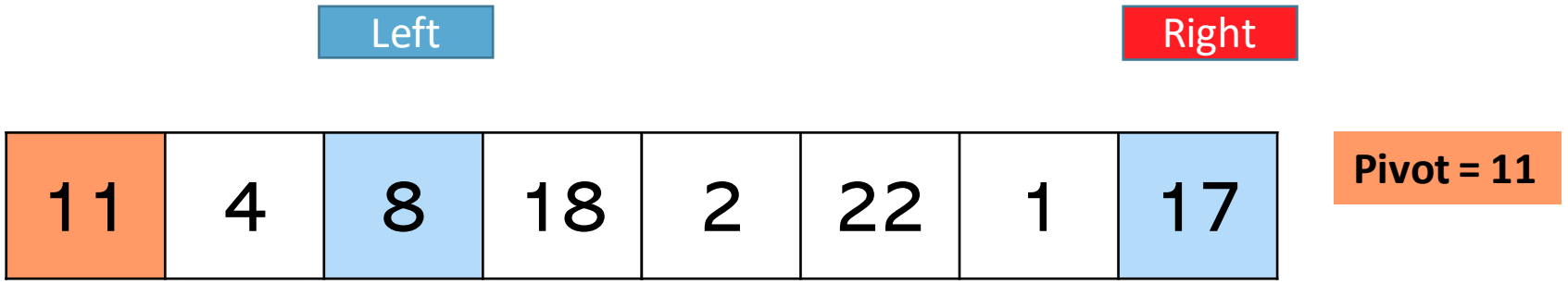
Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Step 6) Continue until left >= right

Partitioning

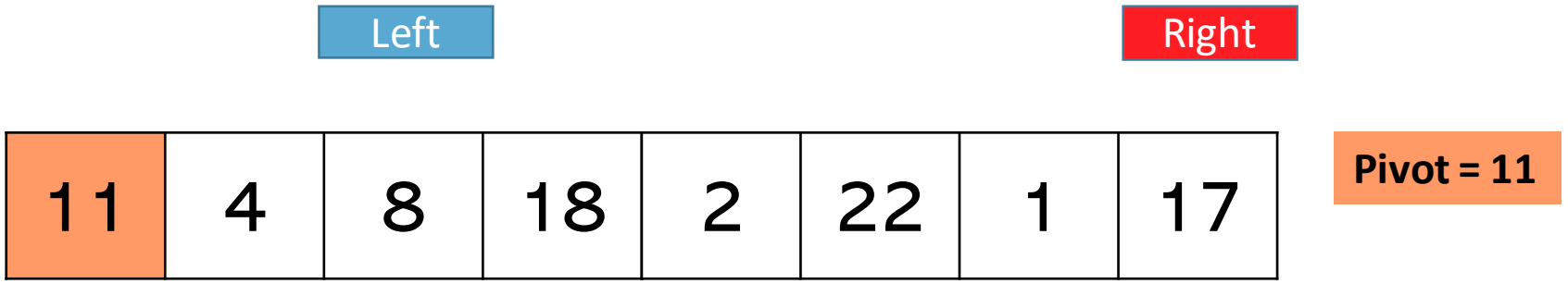


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning

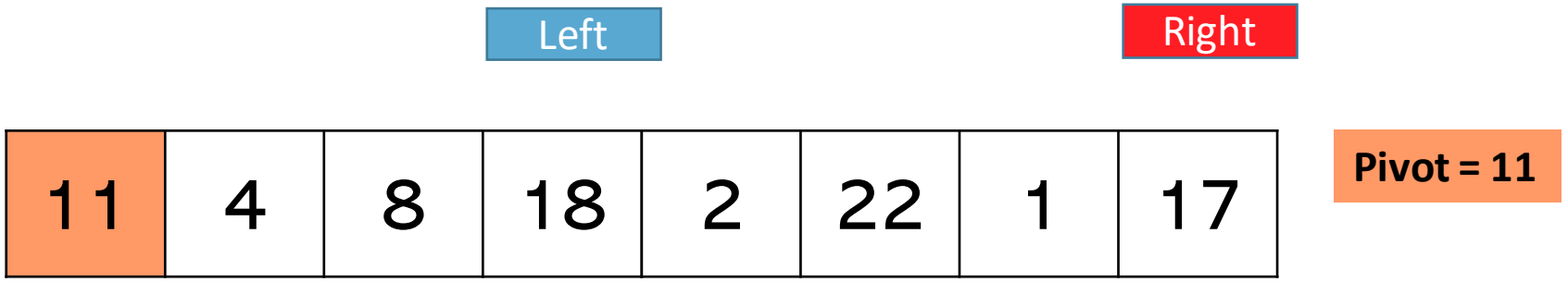


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning

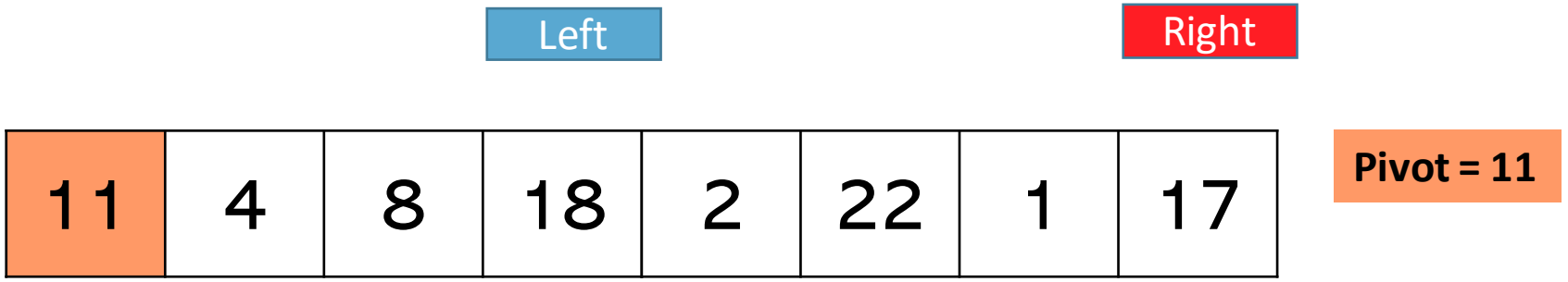


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning



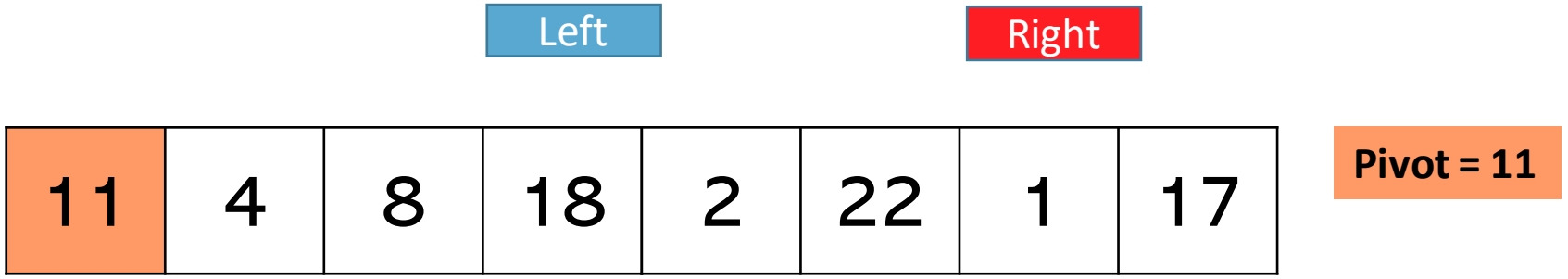
Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Partitioning



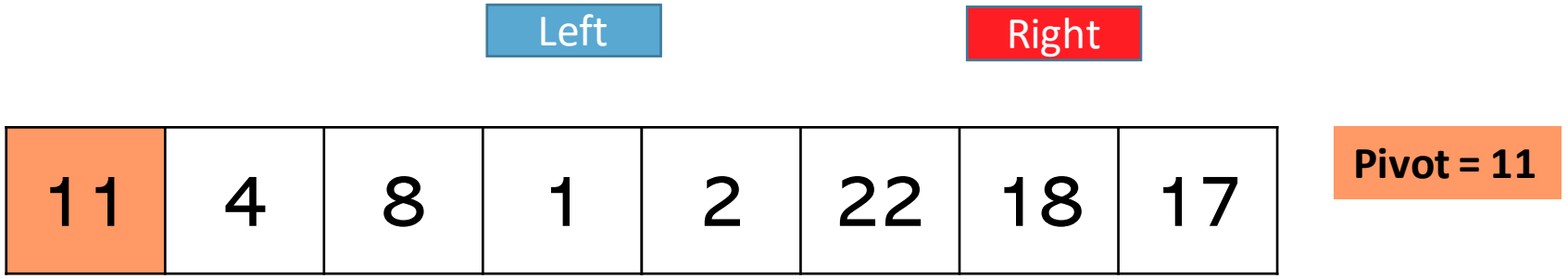
Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Partitioning



Step 1) Get the pivot (left most)

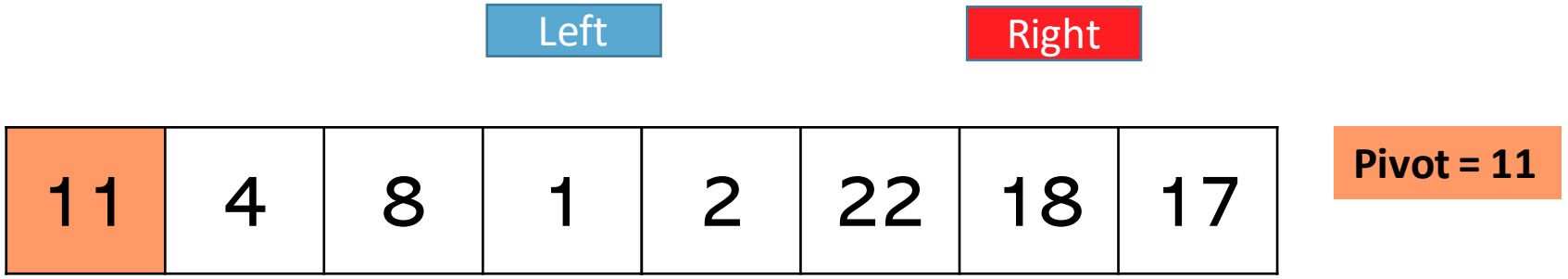
Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Partitioning



Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

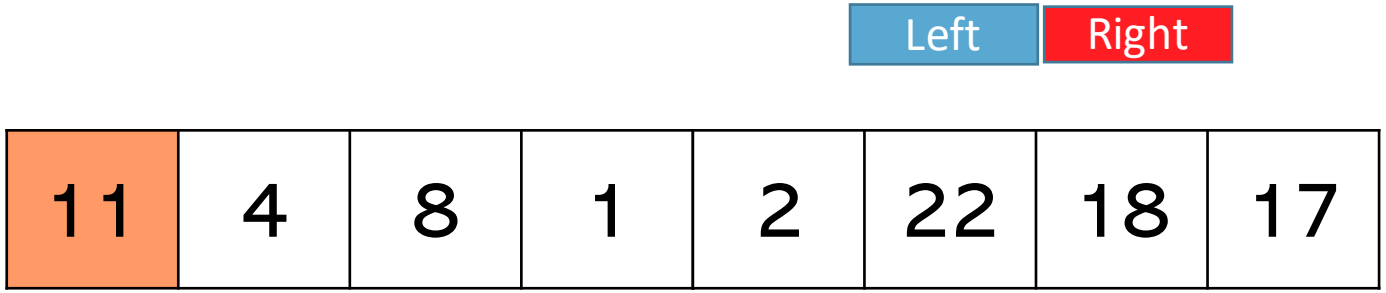
Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Step 6) Continue until left > right

Partitioning

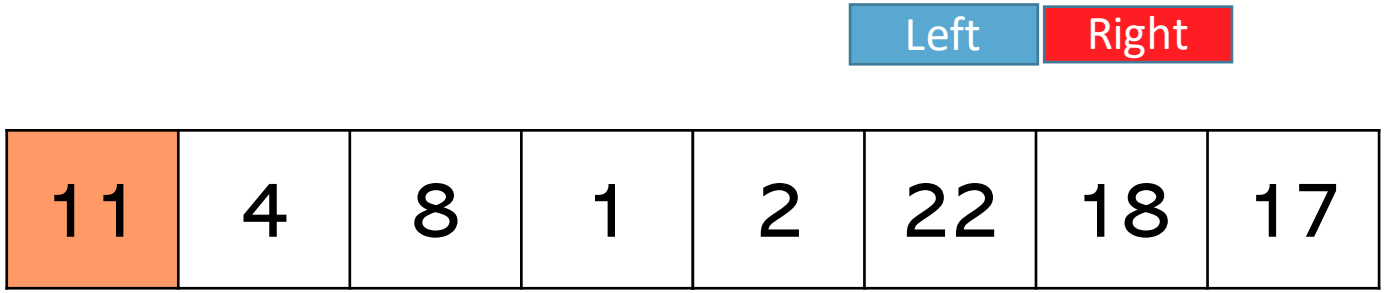


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning



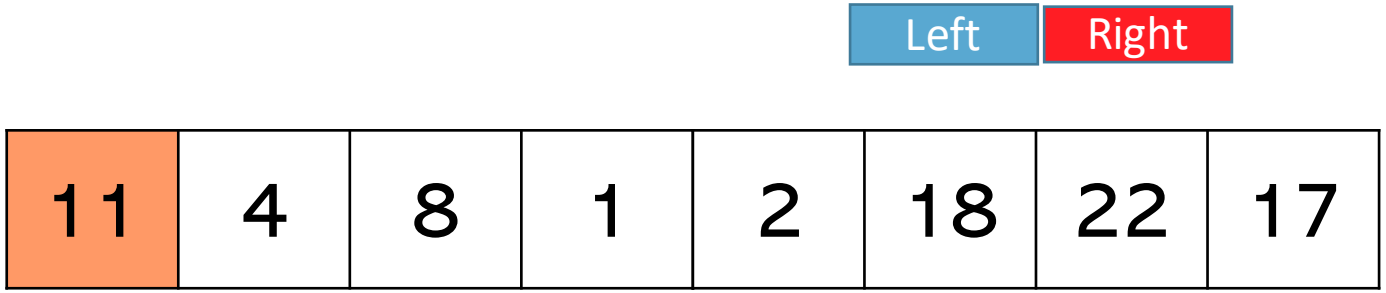
Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Partitioning



Step 1) Get the pivot (left most)

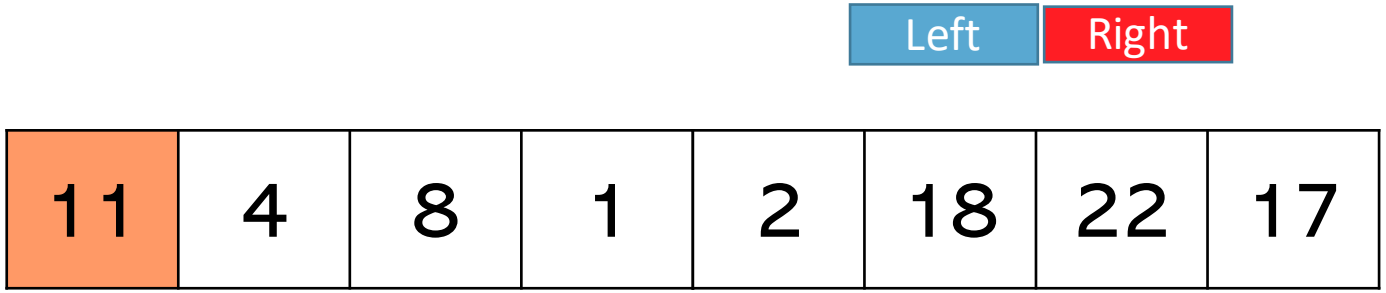
Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Partitioning



Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

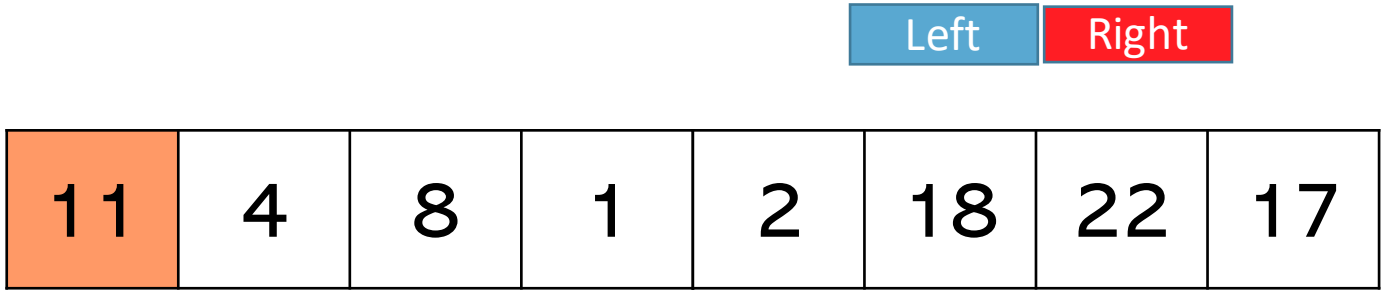
Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Step 5) Swap the value of left with right

Step 6) Continue until left > right

Partitioning

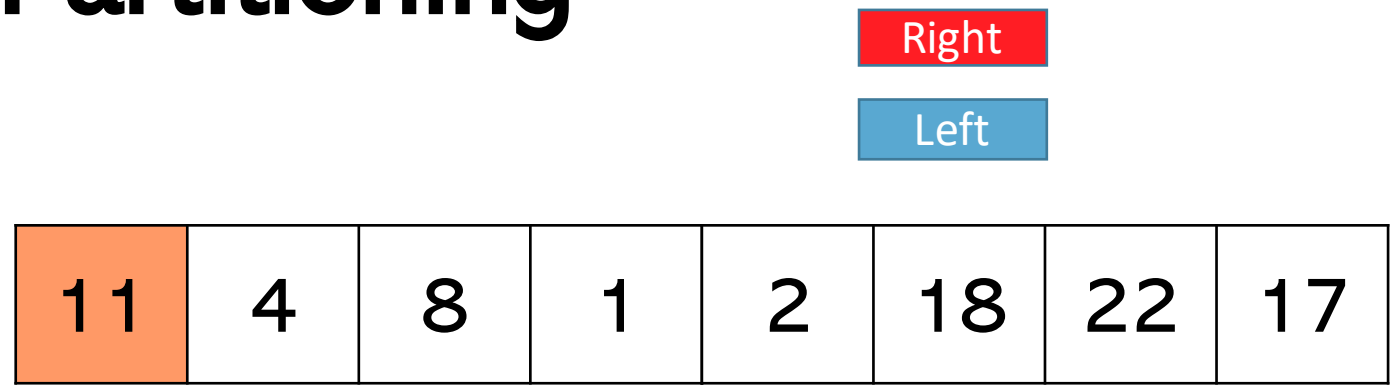


Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Partitioning



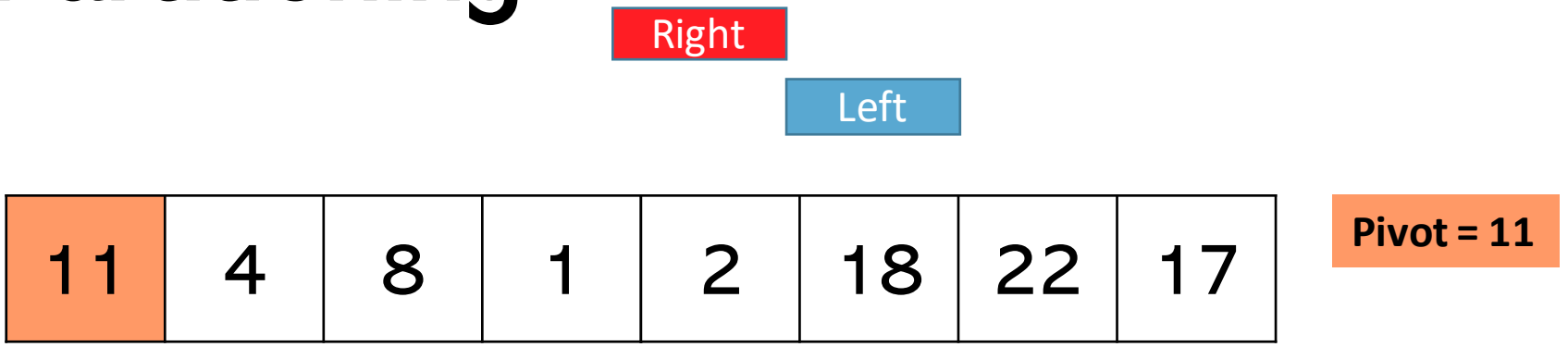
Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Partitioning



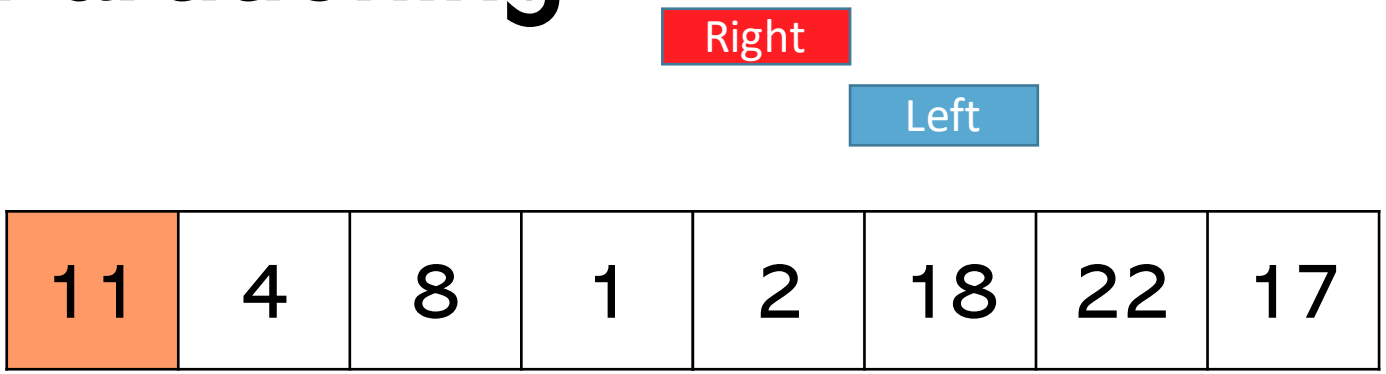
Step 1) Get the pivot (left most)

Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

Partitioning



Step 1) Get the pivot (left most)

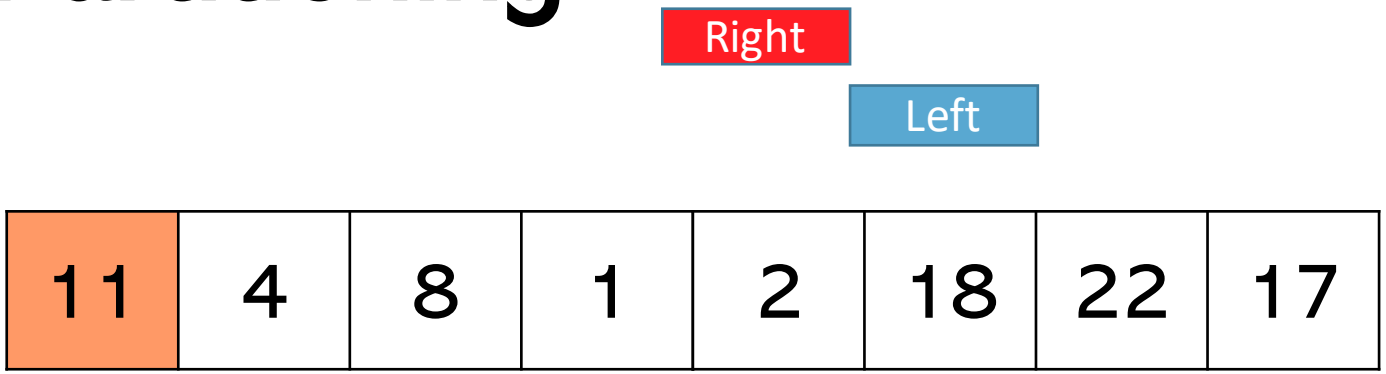
Step 2) Set 2 Pointers : left -> 0 , right -> length - 1

Step 3) Increment left until you find that number >= pivot

Step 4) Decrement right until you find that number <= pivot

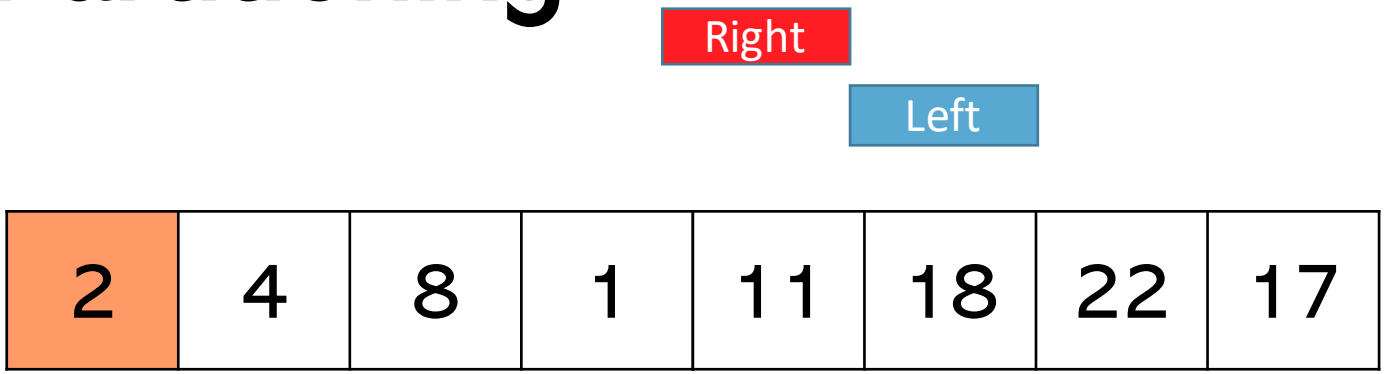
right < left -----> hence the loop has to break

Partitioning



Step 7) Swap the value of pivot with right

Partitioning



Step 7) Swap the value of pivot with right

Now, we found partitioning position :: All left side has smaller and all right has bigger

Recursion on left

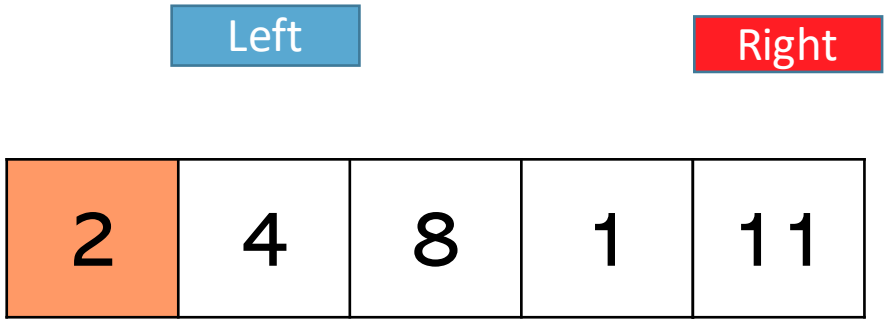
Left

Right

2	4	8	1	11

Pivot = 2

Recursion on left



Pivot = 2

Recursion on left

Left

Right

2	4	8	1	11

Pivot = 2

Recursion on left

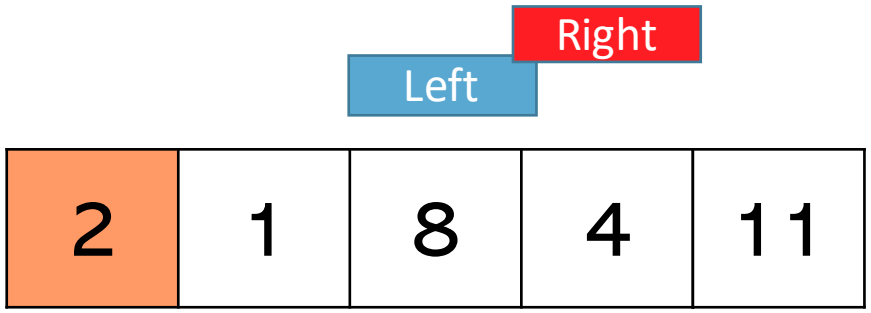
Left

Right

2	1	8	4	11

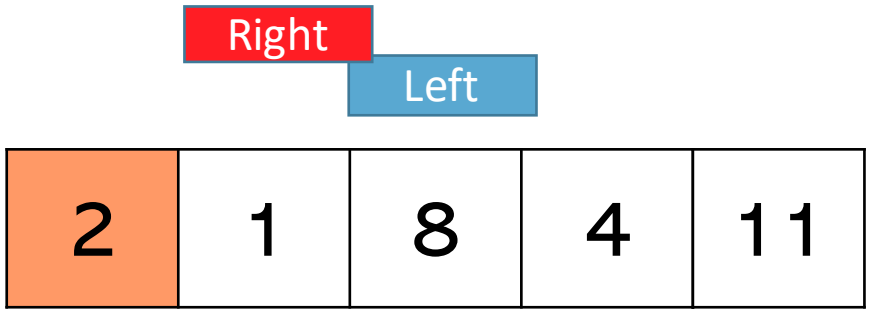
Pivot = 2

Recursion on left



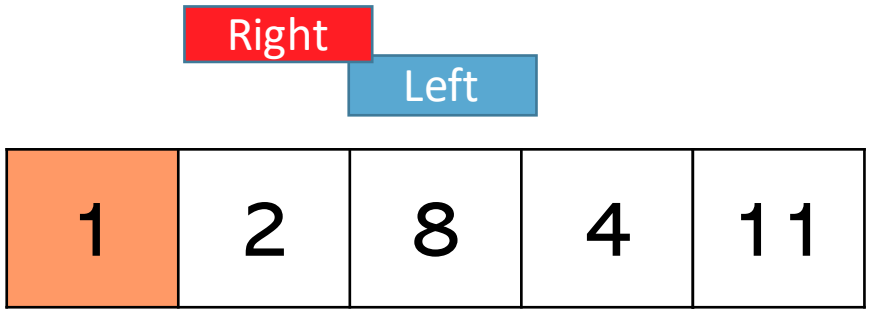
Pivot = 2

Recursion on left



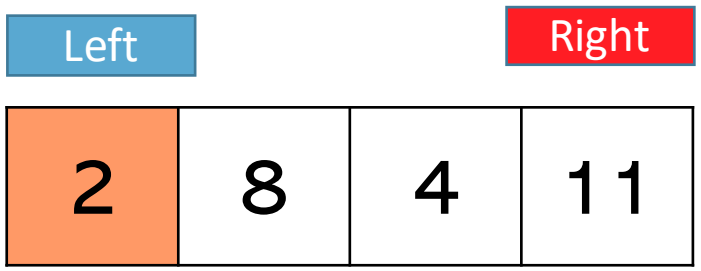
Pivot = 2

Recursion on left



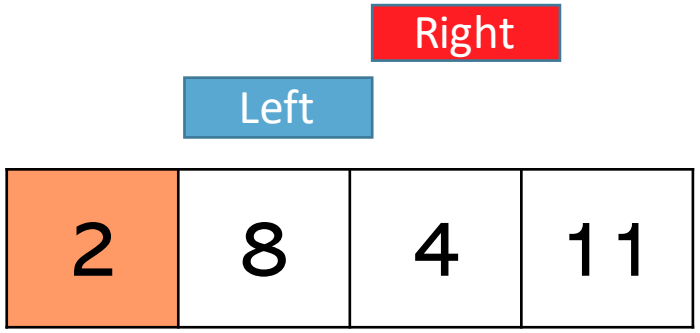
Pivot = 1

Recursion on right



Pivot = 2

Recursion on right



Pivot = 2

Recursion on right

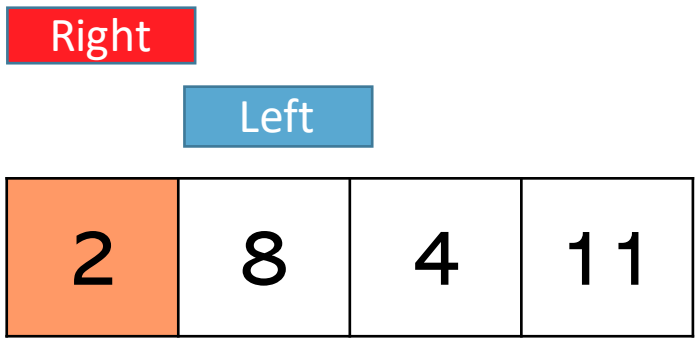
Right

Left

2	8	4	11

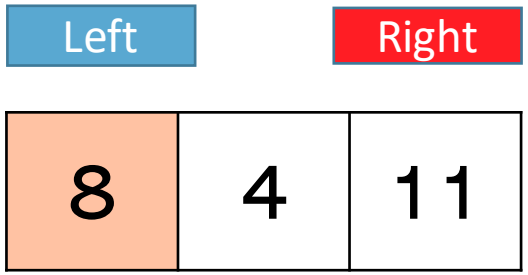
Pivot = 2

Recursion on right

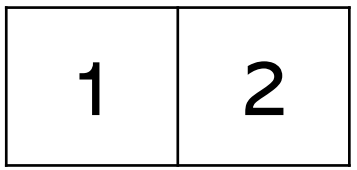


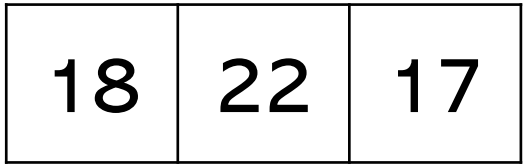
Pivot = 2

Recursion on right



Pivot = 8





Recursion on right

Left

Right

8	4	11

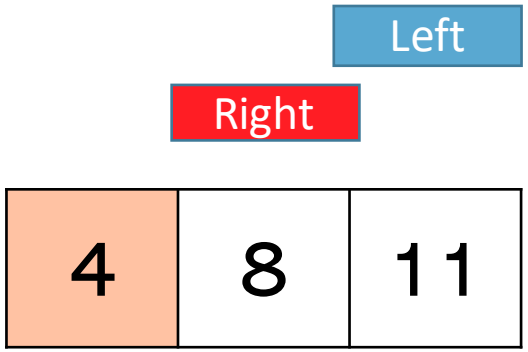
Pivot = 8

1	2

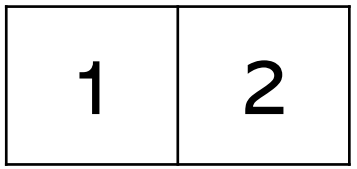
18	22	17

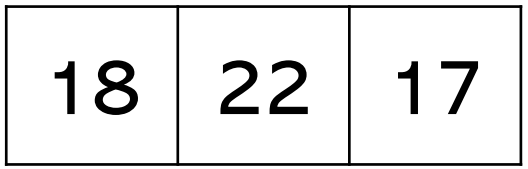
Recursion on right

Recursion on right

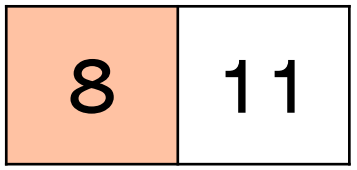


Pivot = 8

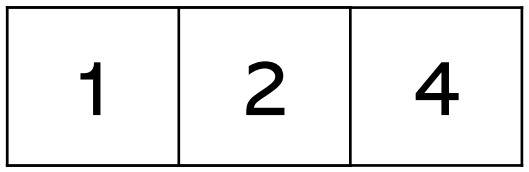


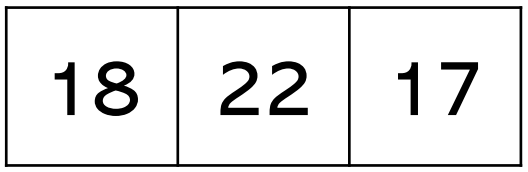


Recursion on right



Pivot = 8

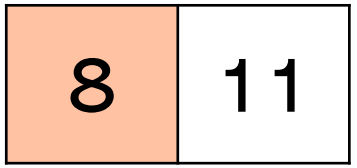




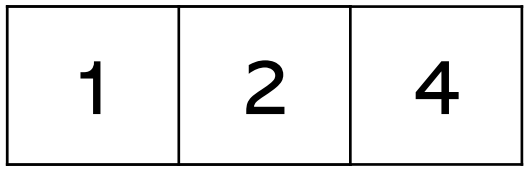
Recursion on right

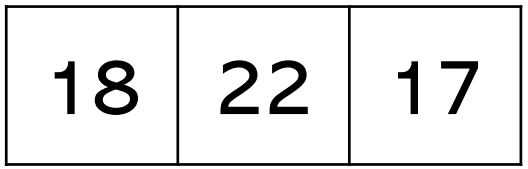
Left

Right



Pivot = 8

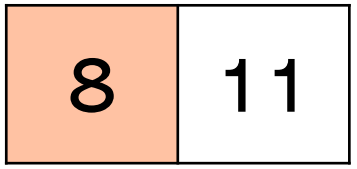




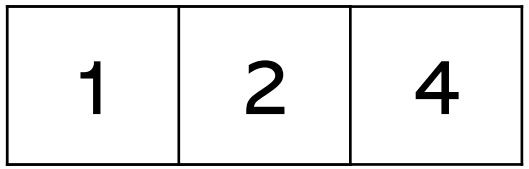
Recursion on right

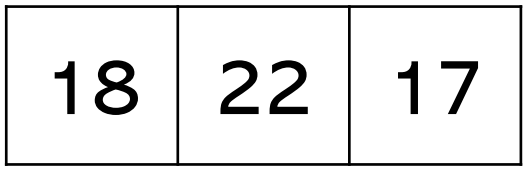
Left

Right



Pivot = 8

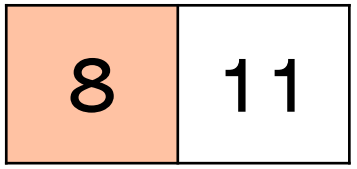




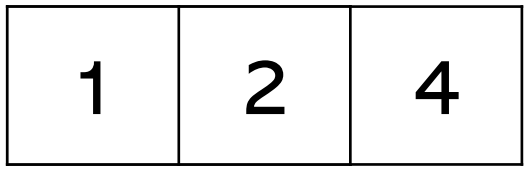
Recursion on right

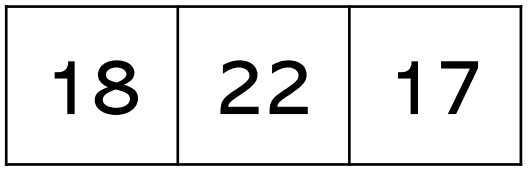
Left

Right



Pivot = 8





Recursion on right

Left

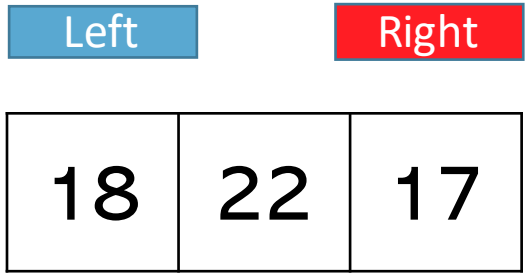
Right

Pivot = 8

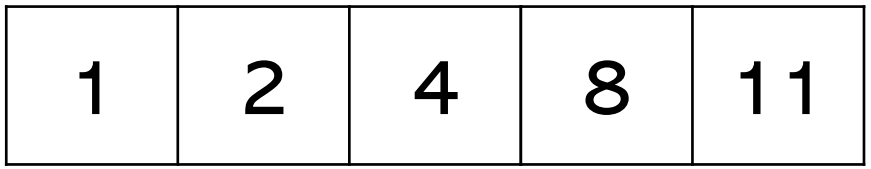
1	2	4	8	11

18	22	17

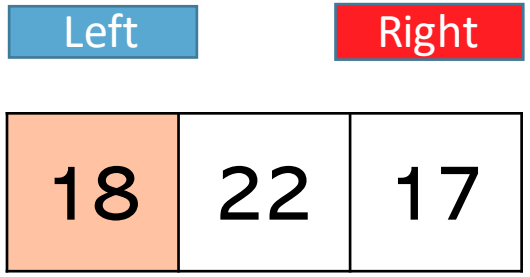
Recursion on right



Pivot = 18



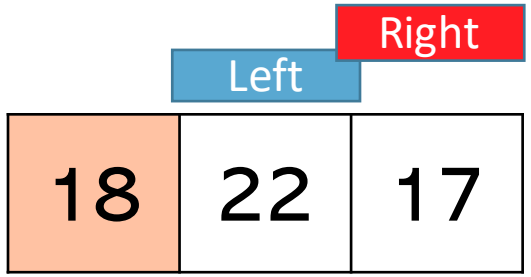
Recursion on right



Pivot = 18

1	2	4	8	11

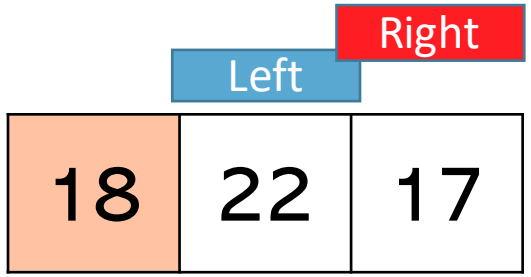
Recursion on right



Pivot = 18

1	2	4	8	11

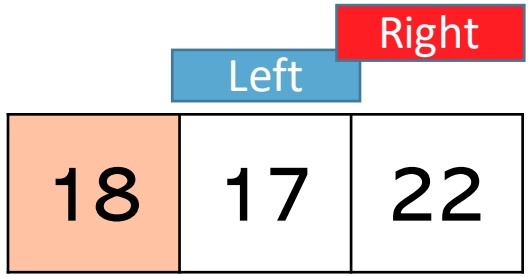
Recursion on right



Pivot = 18

1	2	4	8	11

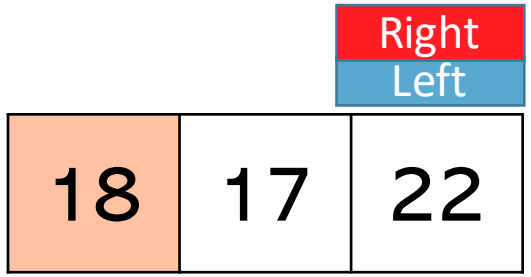
Recursion on right



Pivot = 18

1	2	4	8	11

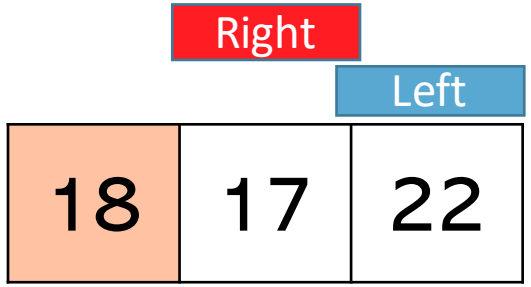
Recursion on right



Pivot = 18

1	2	4	8	11

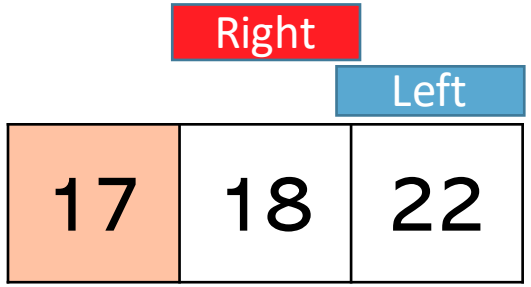
Recursion on right



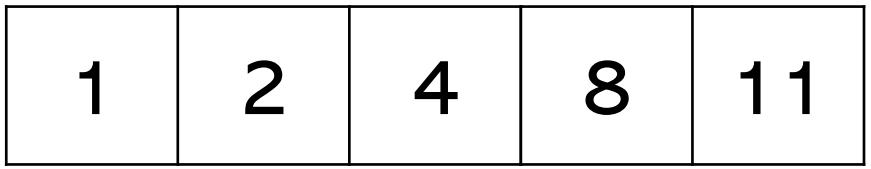
Pivot = 18

1	2	4	8	11

Recursion on right



Pivot = 18



Recursion on right

Right

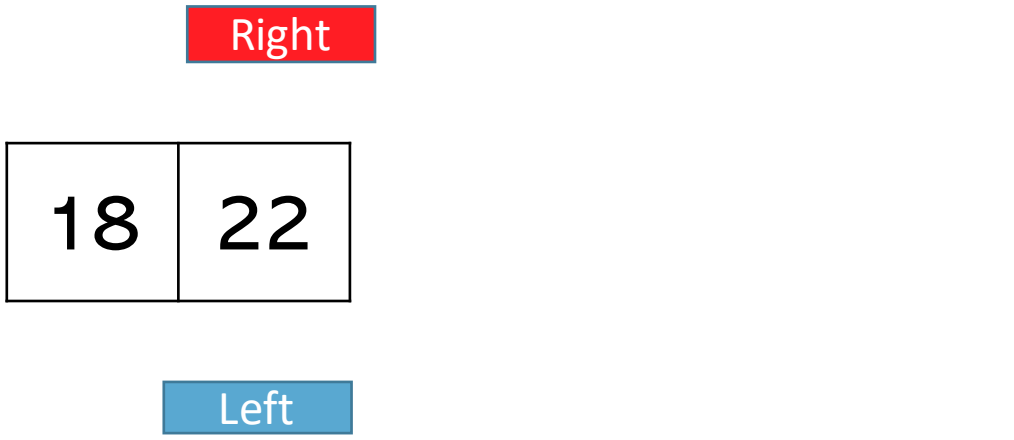
18	22

Left

Pivot = 18

1	2	4	8	11	17

Recursion on right



Recursion on right

Right

18	22

Left

Pivot = 18

1	2	4	8	11	17

Recursion on right

Right

18	22

Left

Pivot = 18

1	2	4	8	11	17

Recursion on right