Divide & Conquer

- ✓ Powerful approach for solving conceptually difficult problems
- ✓ Often used to find an optimal solution of a problem.
- ✓ Decompose a given problem into two or more similar, but simpler, subproblems, to solve them in turn, and
- ✓ Compose their solutions to solve the given problem.

Divide & Conquer

3 Step Process

1. Divide. Divide the problem into a set of subproblems
2. Conquer. Solve each subproblem recursively.
3. Combine. Combine the results of each subproblem.

Count the number of ones

Given a binary sorted array, find the number of ones

Example 1:

Input: nums =[0,0,0,0,1,1,1]
Output: 3

Count the Ones

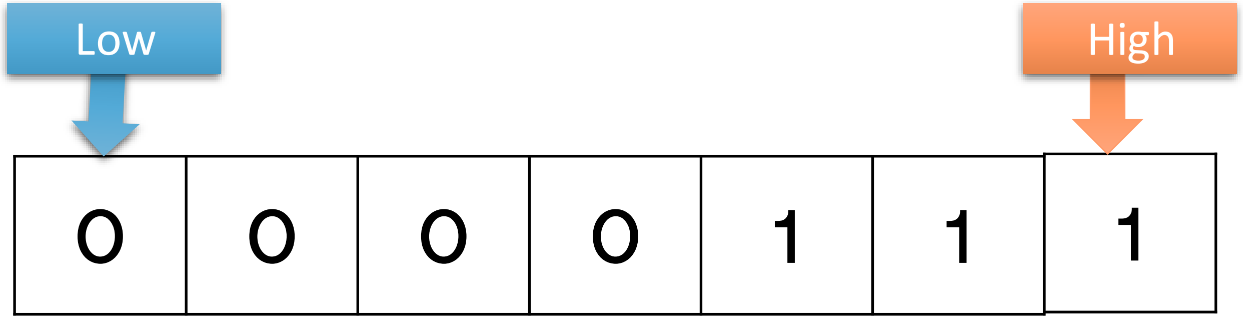
0	0	0	0	1	1	1

Rules to break the recursion:

1) Numbers on right is 0, then return count as 0
2) Numbers on left is 1, return (right – left + 1)

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

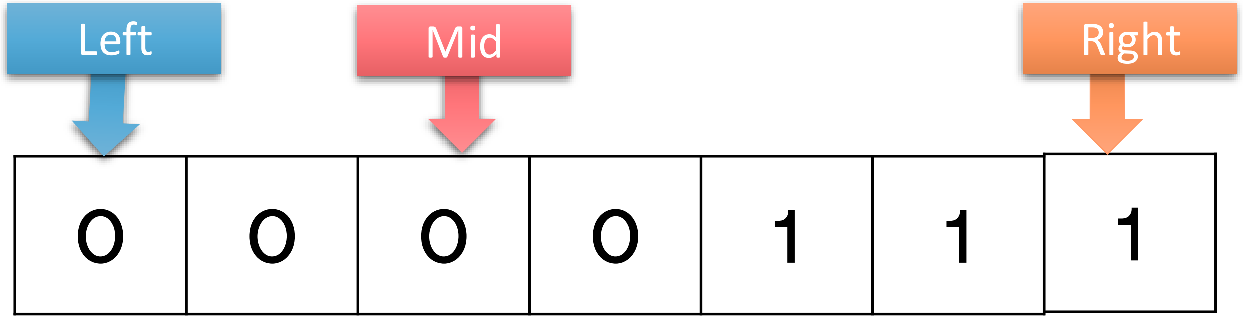
Count the Ones



0	0	0	0	1	1	1

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones



0	0	0	0	1	1	1

✓ 2 Index (left at 0 and right at length-1)

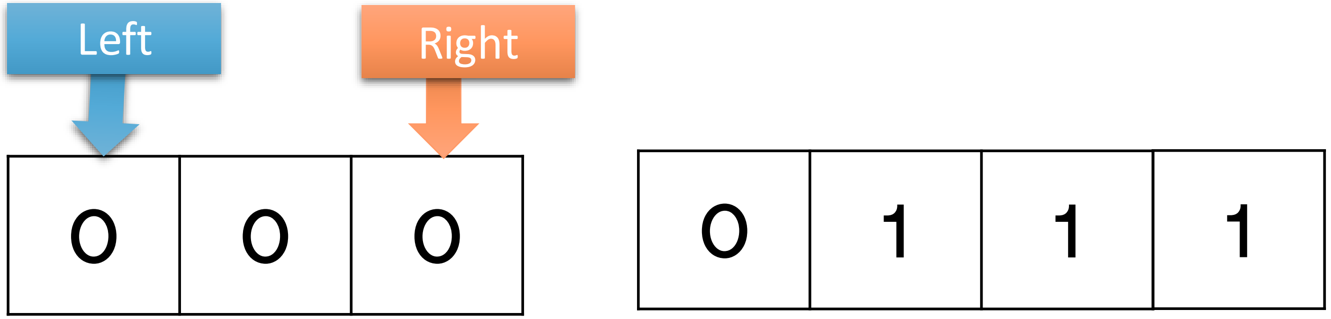
✓ Find the mid => (left + right) / 2 (and divide into half)

✓ Recursively call the left index to mid point to get count

✓ Recursively call the mid+1 to right index to get count

✓ Add the left recursive sum with right recursive sum

Count the Ones



Left

Right

0	0	0

0	1	1	1

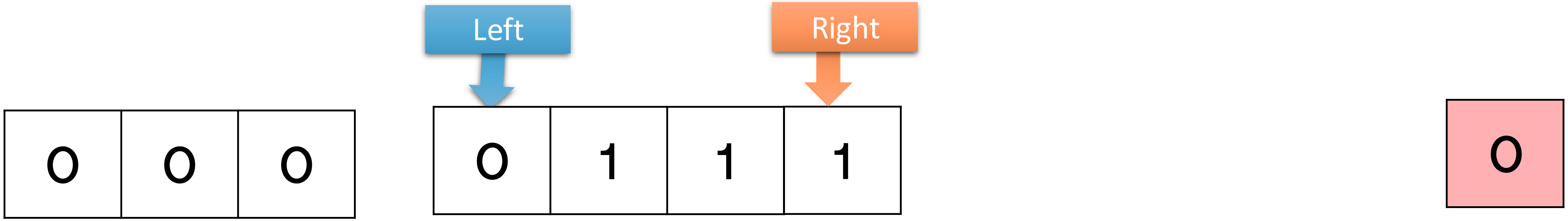
Rules to break the recursion:

1) Numbers on right is 0, then return count as 0
2) Numbers on left is 1, return (right – left + 1)

0

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones



Left

Right

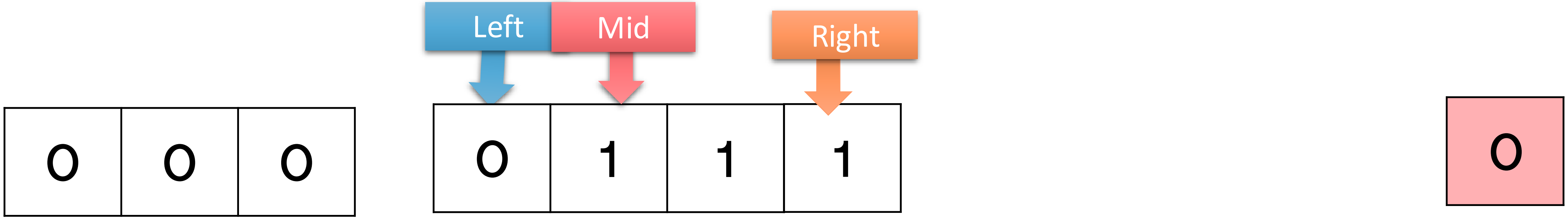
0	0	0

0	1	1	1

0

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones



Left

Mid

Right

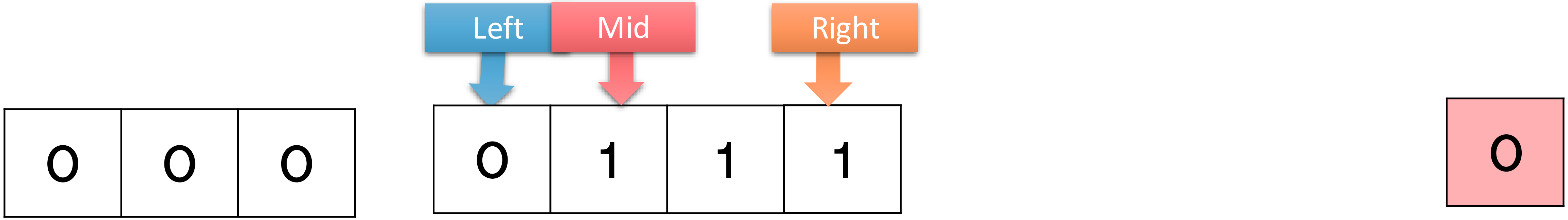
0	0	0

0	1	1	1

0

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones



Left Mid Right

0	0	0

0	1	1	1

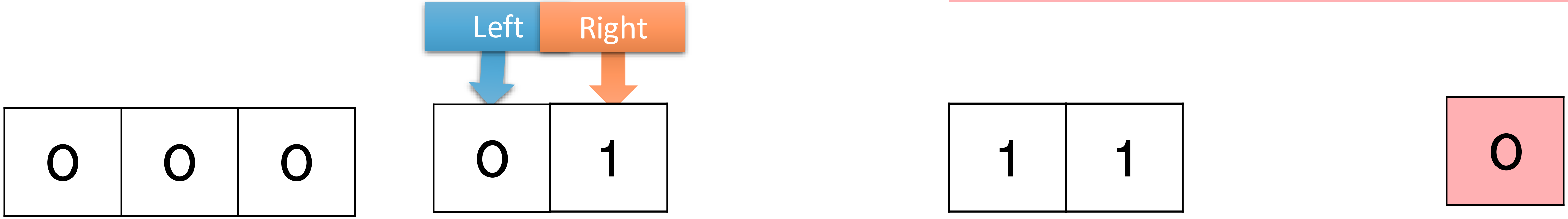
0

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones

Rules to break the recursion:

1) Numbers on right is 0, then return count as 0
2) Numbers on left is 1, return (right – left + 1)



Left

Right

0 0 0

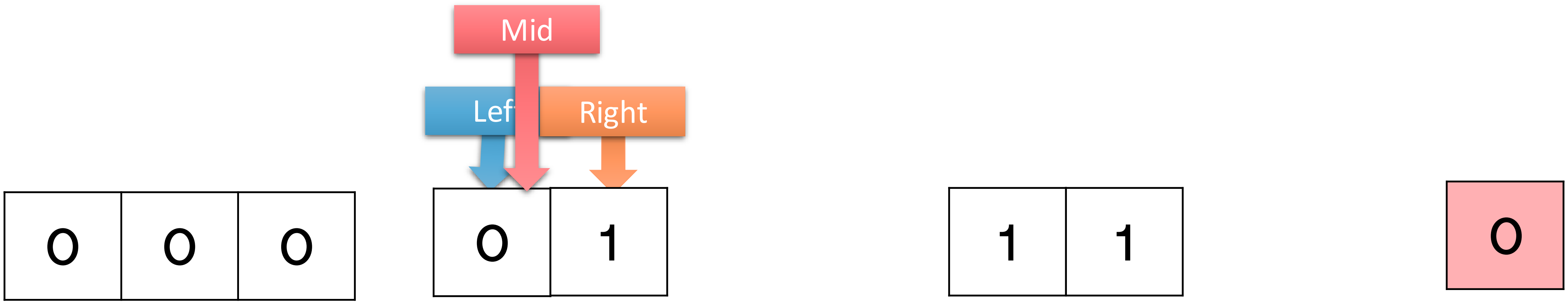
0 1

1 1

0

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones



Mid

Left

Right

0 0 0

0 1

1 1

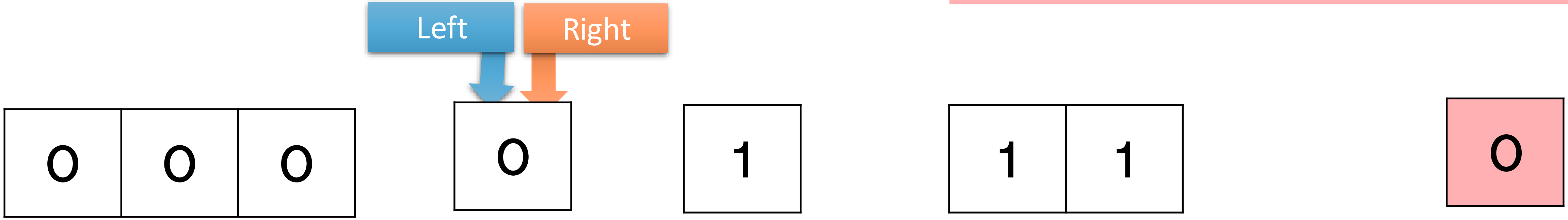
0

- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones

Rules to break the recursion:

1) Numbers on right is 0, then return count as 0
2) Numbers on left is 1, return (right – left + 1)



✓ 2 Index (left at 0 and right at length-1)

✓ Find the mid => (left + right) / 2 (and divide into half)

✓ Recursively call the left index to mid point to get count

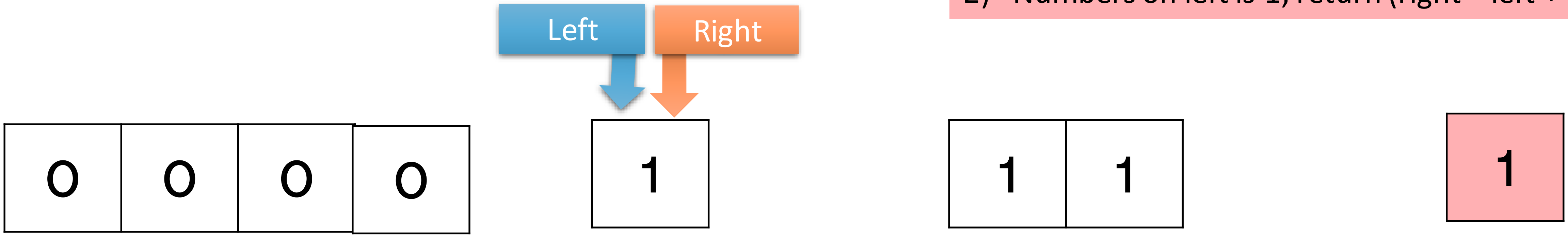
✓ Recursively call the mid+1 to right index to get count

✓ Add the left recursive sum with right recursive sum

Count the Ones

Rules to break the recursion:

1) Numbers on right is 0, then return count as 0
2) Numbers on left is 1, return (right – left + 1)

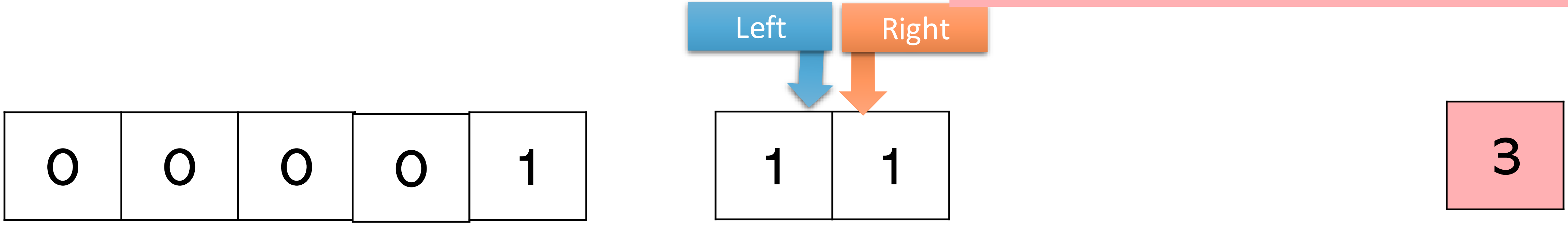


- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum

Count the Ones

Rules to break the recursion:

1) Numbers on right is 0, then return count as 0
2) Numbers on left is 1, return (right – left + 1)



- ✓ 2 Index (left at 0 and right at length-1)
- ✓ Find the mid => (left + right) / 2 (and divide into half)
- ✓ Recursively call the left index to mid point to get count
- ✓ Recursively call the mid+1 to right index to get count
- ✓ Add the left recursive sum with right recursive sum