Problem Definition

Given an (unsorted) integer array, find a pair with the given sum.

Example:

Inputs:

array = {3, 4, 2, 7, 11, 8} and sum = 12

Output:

{4,8}

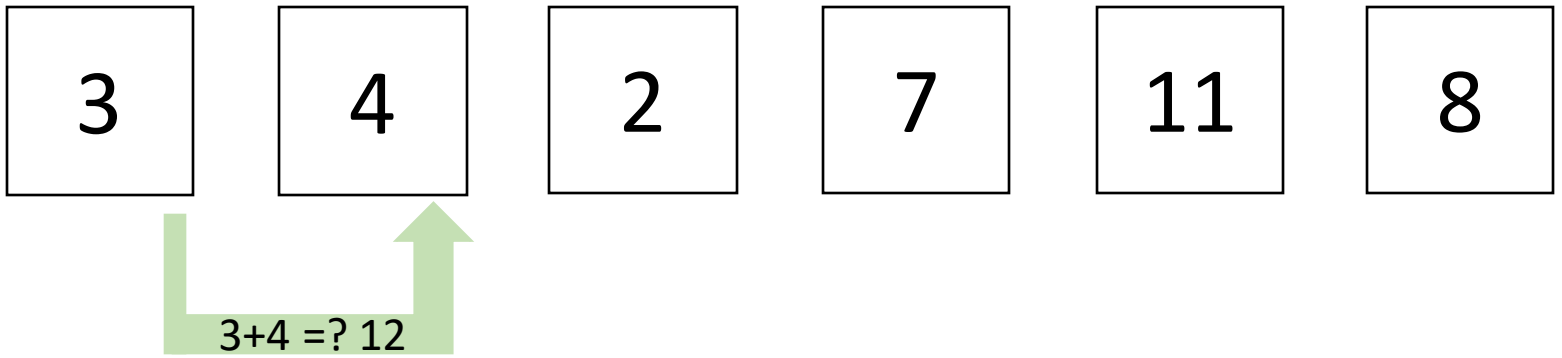
Solution 1 (brute force)

```
findPair (array, sum):
    n ← length of the array
    for i from 0 to n:
        for j from i to n:
            if array[i] + array[j] == sum then
                return (array[i], array[j])
            end if
        end for
    end for
    return -1
```

Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

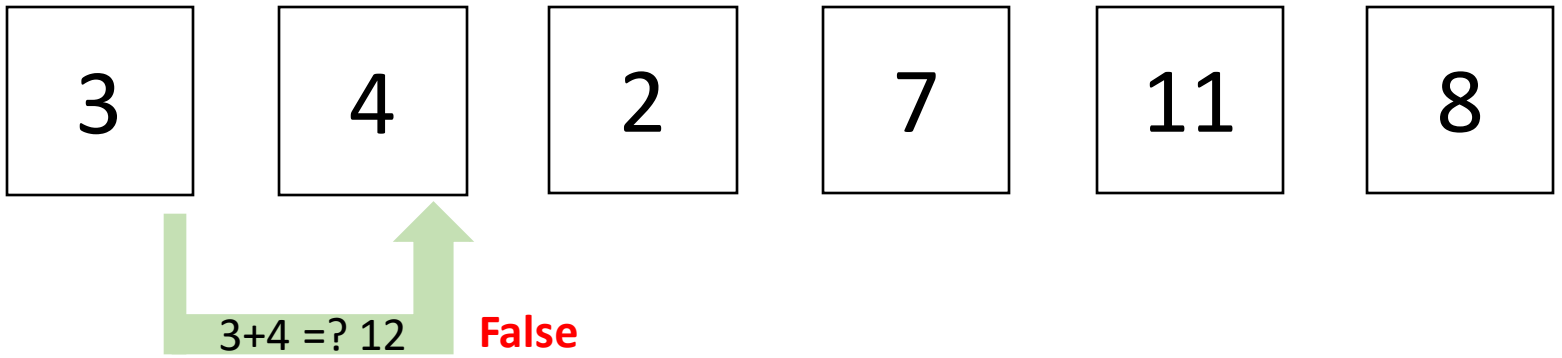
First iteration of the outer loop:



Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

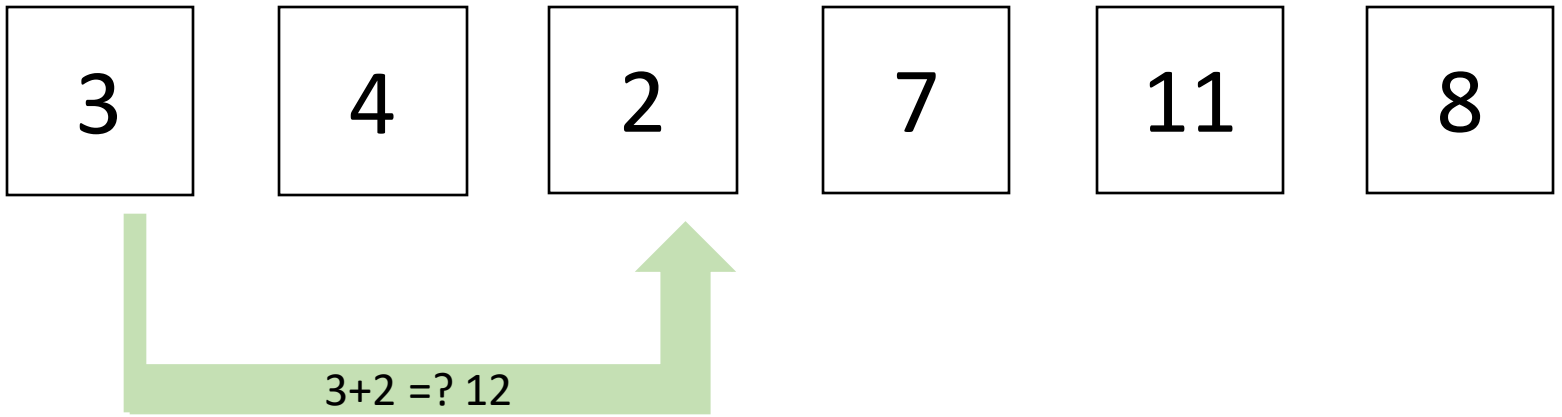
First iteration of the outer loop:



Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

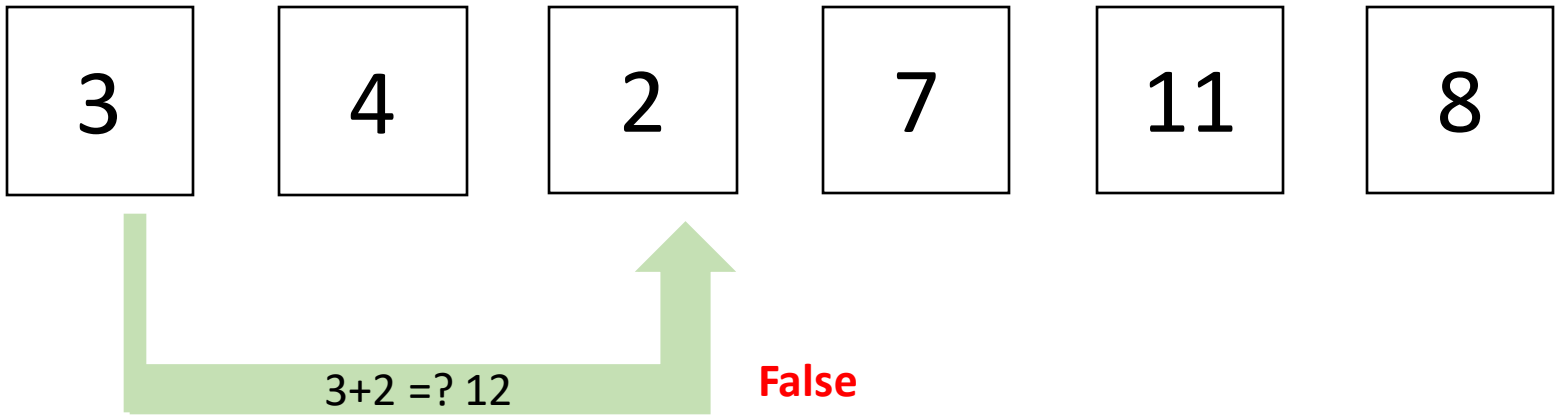
First iteration of the outer loop:



Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

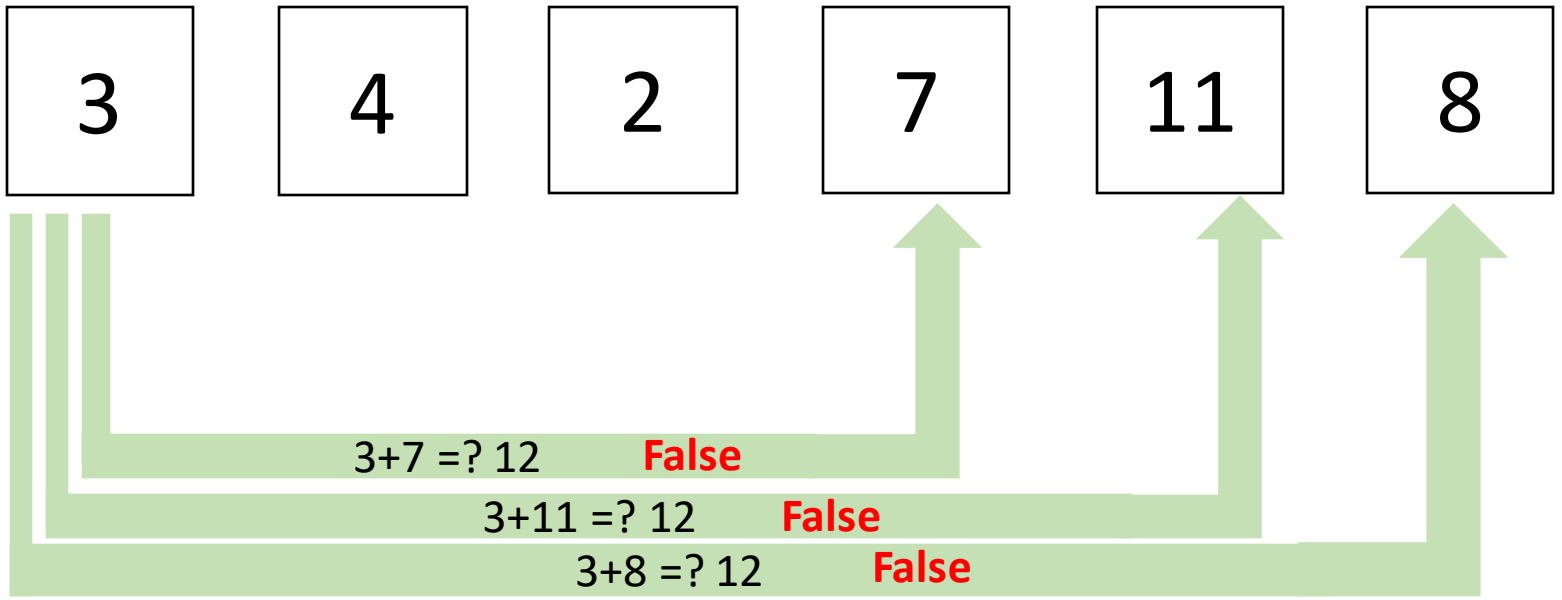
First iteration of the outer loop:



Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

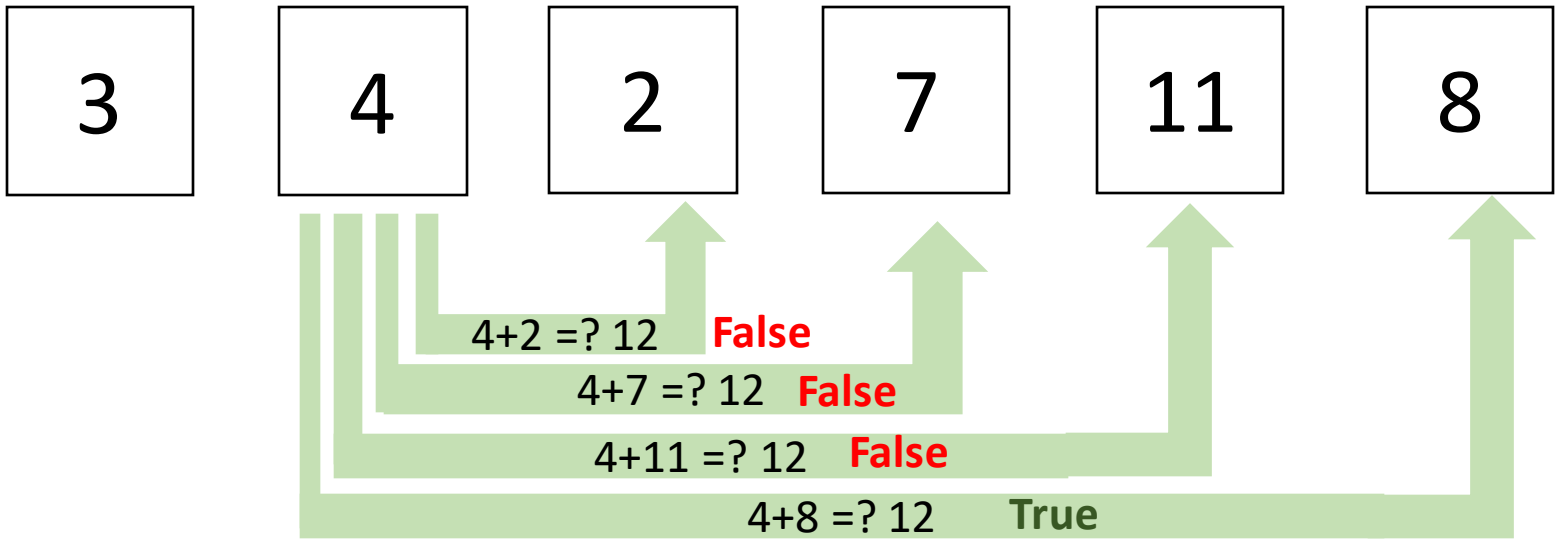
First iteration of the outer loop:



Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

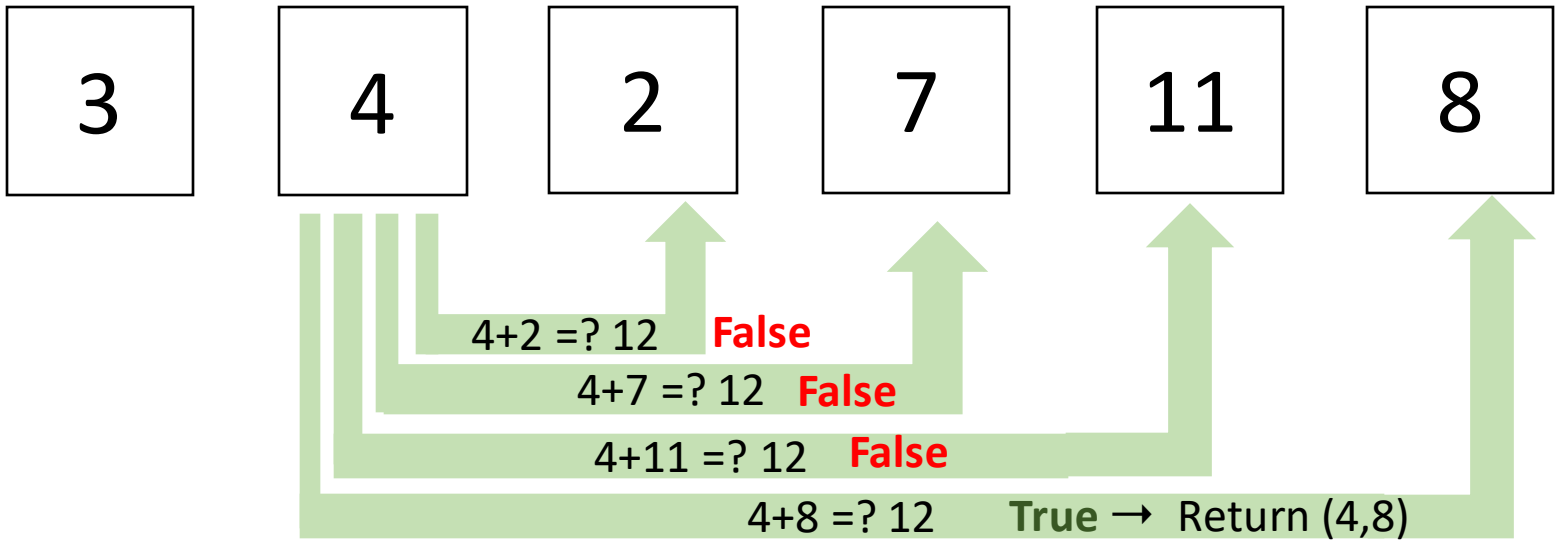
First iteration of the outer loop:



Solution 1 (brute force)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

First iteration of the outer loop:



Solution 2 (using sorting)

Time complexity?

- In the best case, the algorithm returns the first two elements. Therefore, the complexity is $\Omega(1)$.
- In the worst case, there might not be such a pair in the array, or the pair is the last two elements of the array. In this case nested loops will be completed which takes O(n^2) time.

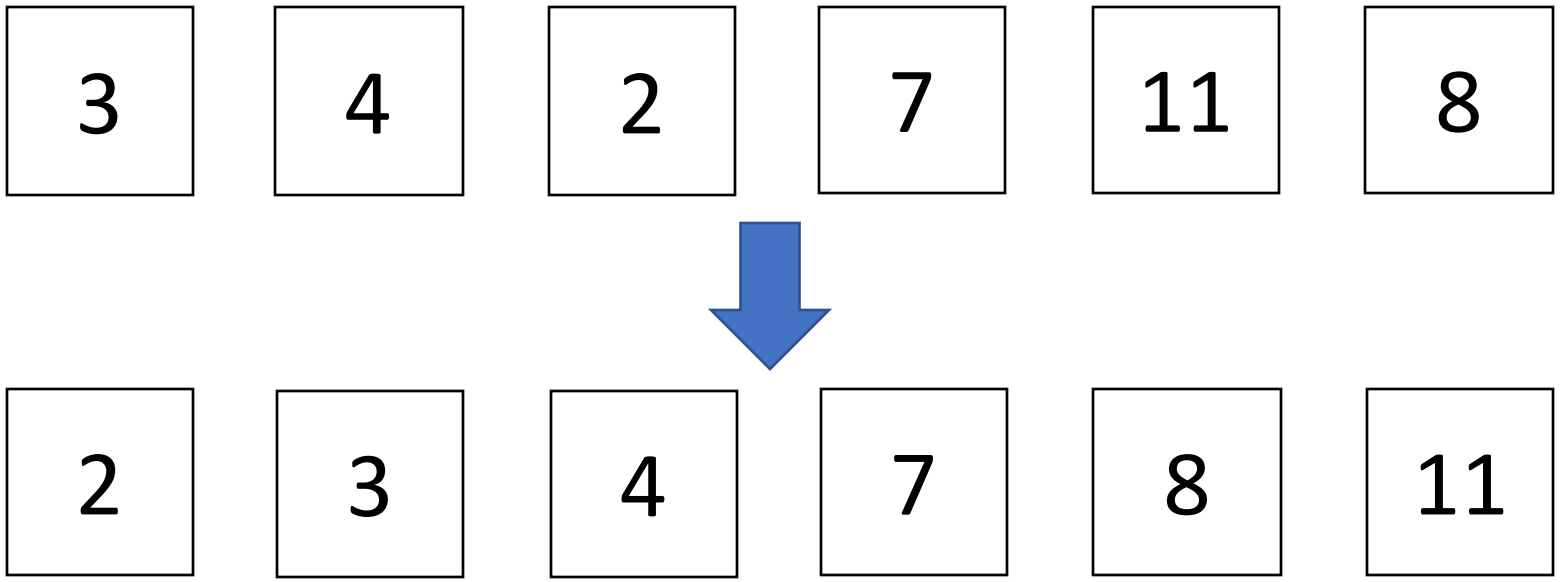
Solution 2 (using sorting)

```
findPair (array, sum):
    array ← merge_sort (array)      # sorting the array by using merge sort
    left  ← 0
    right ← length of the array – 1
    while left < right:
        if array[left] + array[right] == sum then
            return (array[left], array[right])
        else if array[left] + array[right] < sum then
            left ← left + 1
        else  # if array[left] + array[right] > sum
            right ← right – 1
        end if
    end while
    return -1
```

Solution 2 (using sorting)

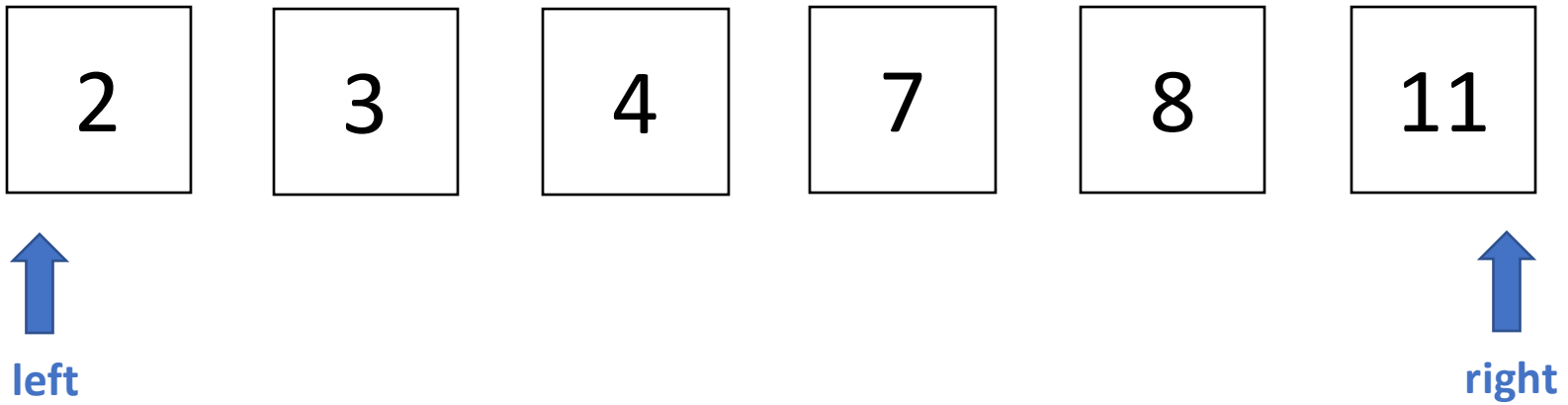
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

First step is to sort the array:



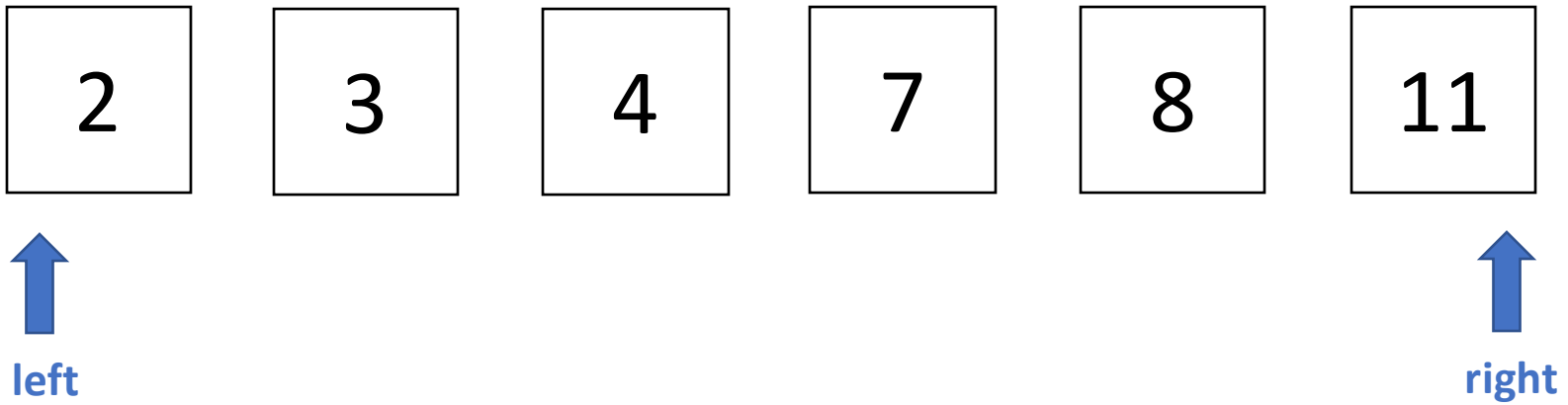
Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 2 (using sorting)

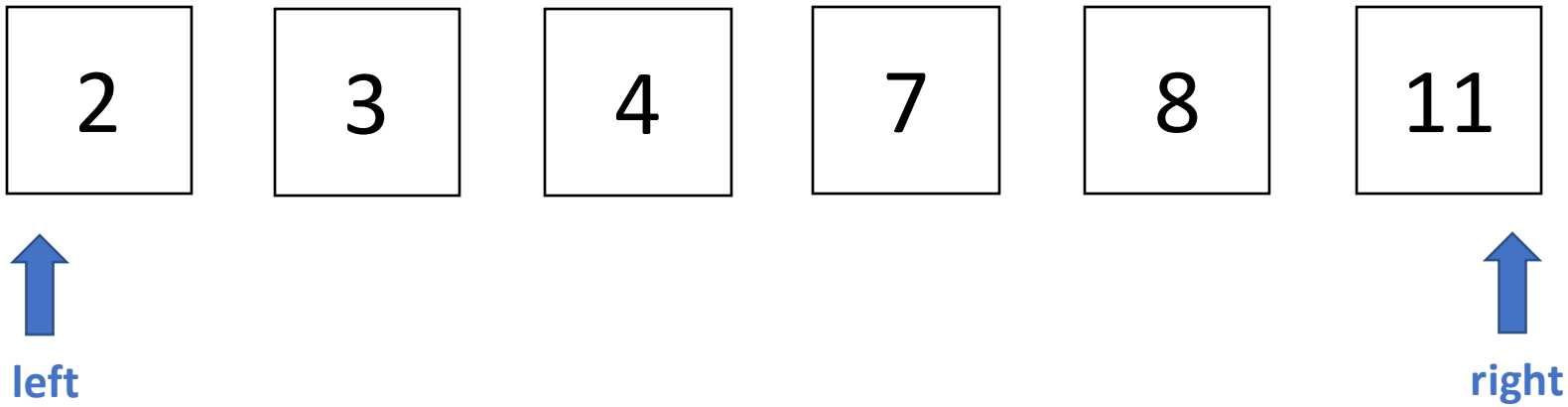
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



array [left] + array [right] =? sum

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

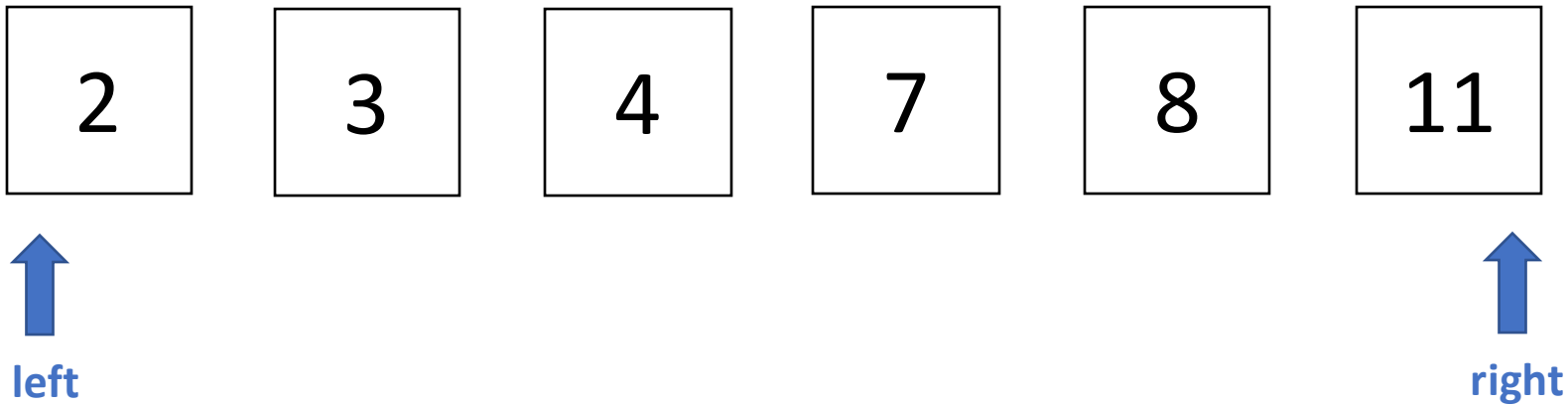


array [left] + array [right] =? sum

2 + 11 =? 12

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

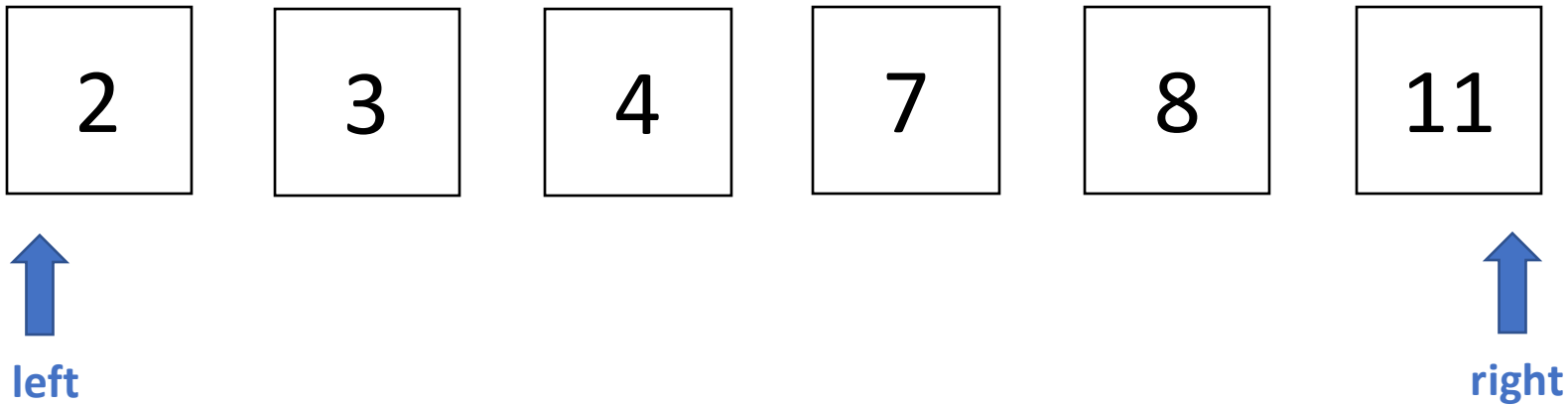


array [left] + array [right] =? sum

2 + 11 =? 12 **False**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

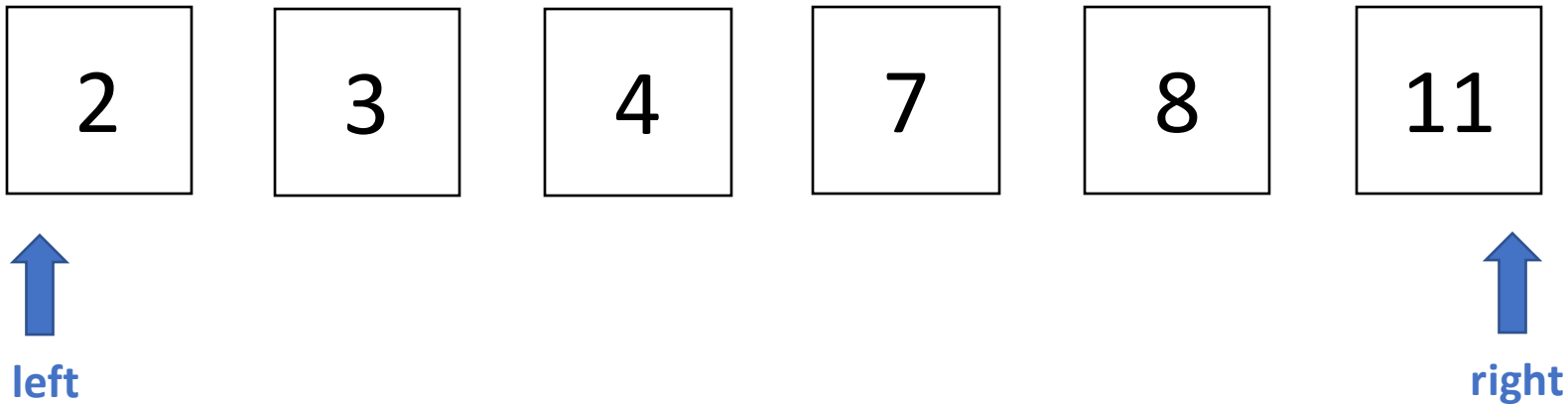


array [left] + array [right] =? sum

2 + 11 =? 12 **False** → array [left] + array [right] <? sum

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

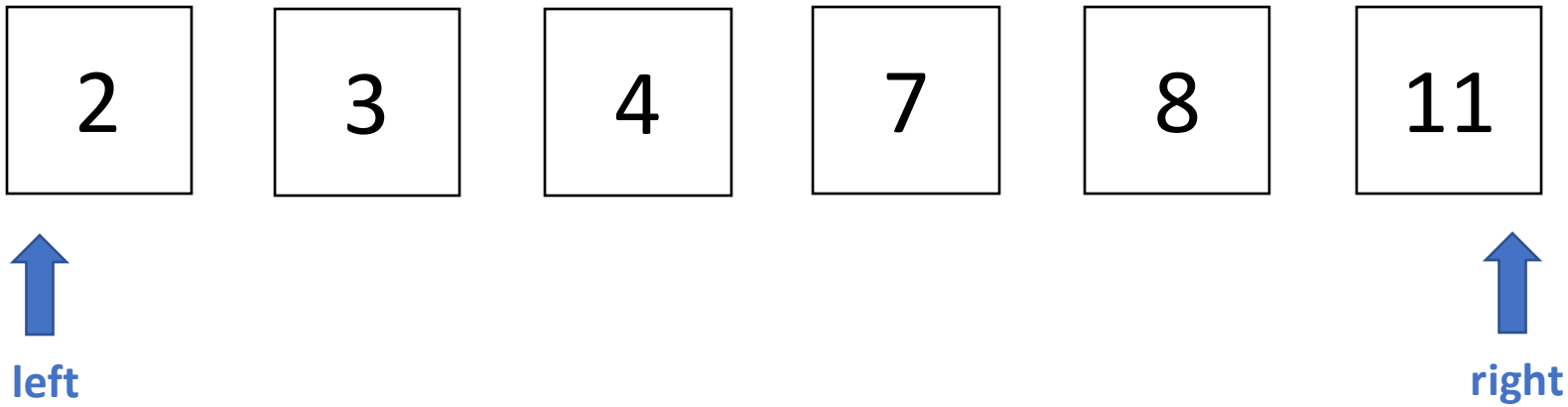


array [left] + array [right] =? sum

2 + 11 =? 12 **False** → array [left] + array [right] <? sum
2 + 11 <? 11

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

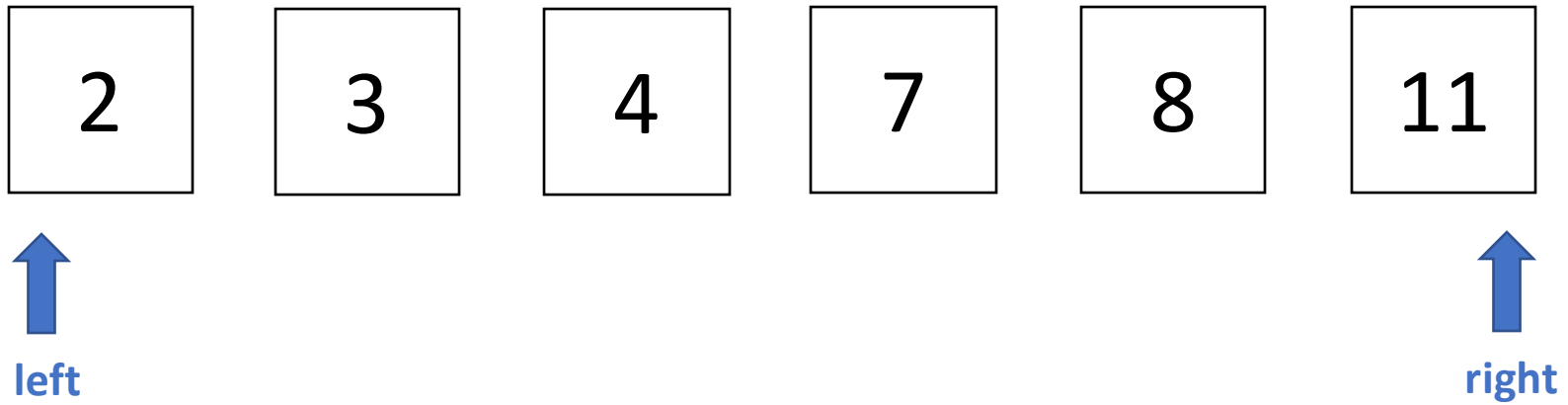


array [left] + array [right] =? sum

2 + 11 =? 12 **False** → array [left] + array [right] <? sum
2 + 11 <? 11 **False**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

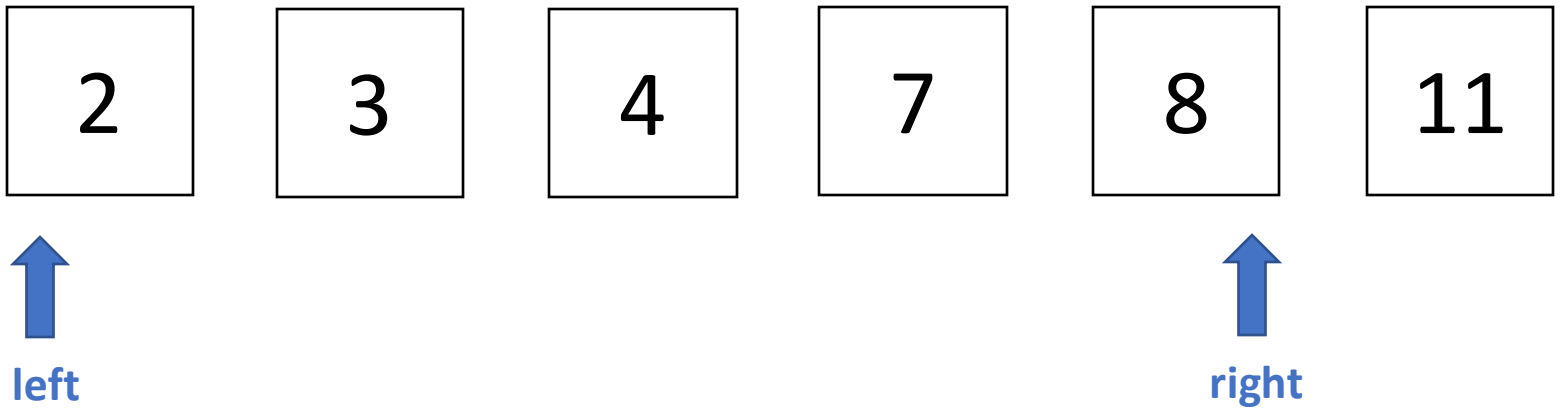


array [left] + array [right] =? sum

2 + 11 =? 12 **False** → array [left] + array [right] <? sum
2 + 11 <? 11 **False** **then** update 'right' as 'right -1'

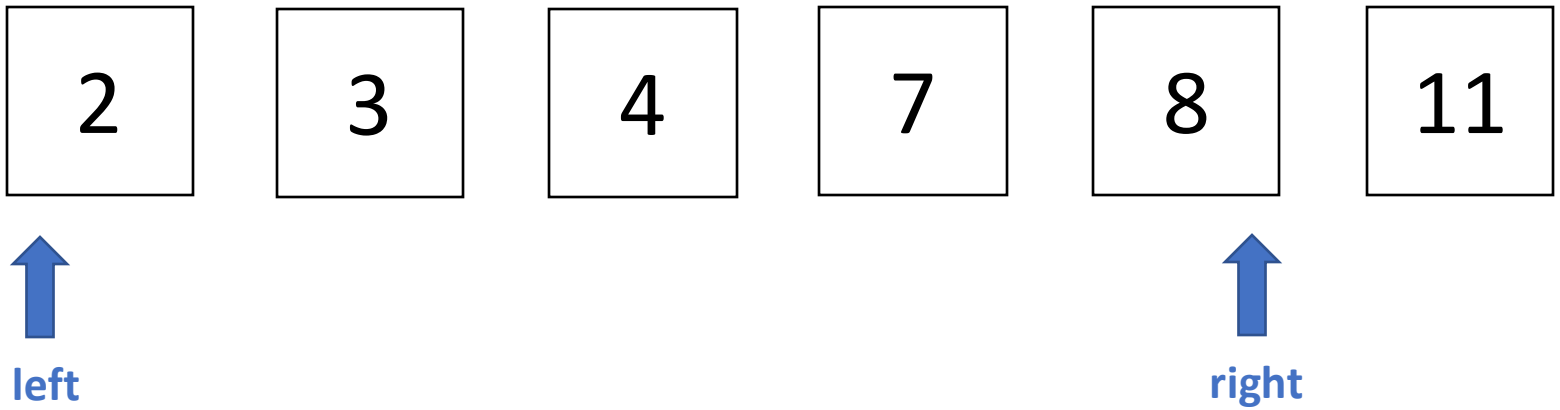
Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 2 (brute force)

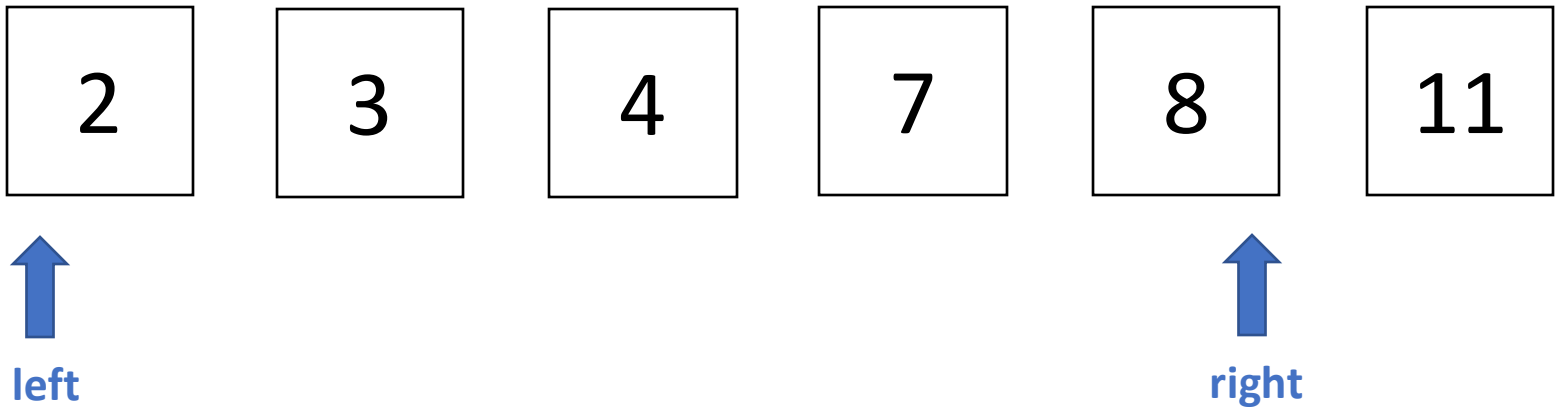
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



array [left] + array [right] =? sum

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

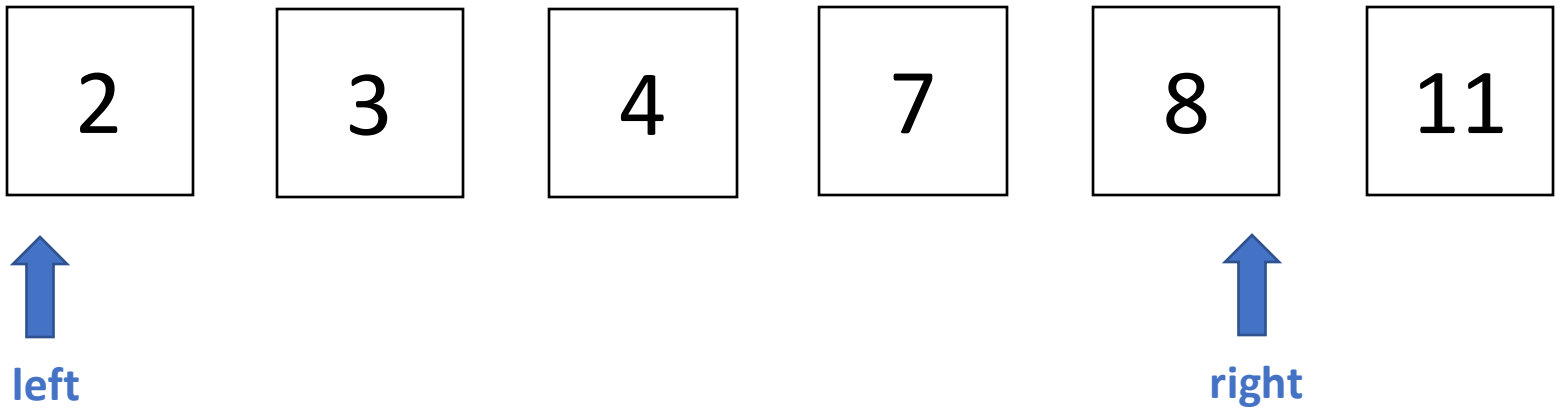


array [left] + array [right] =? sum

2 + 8 =? 12

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

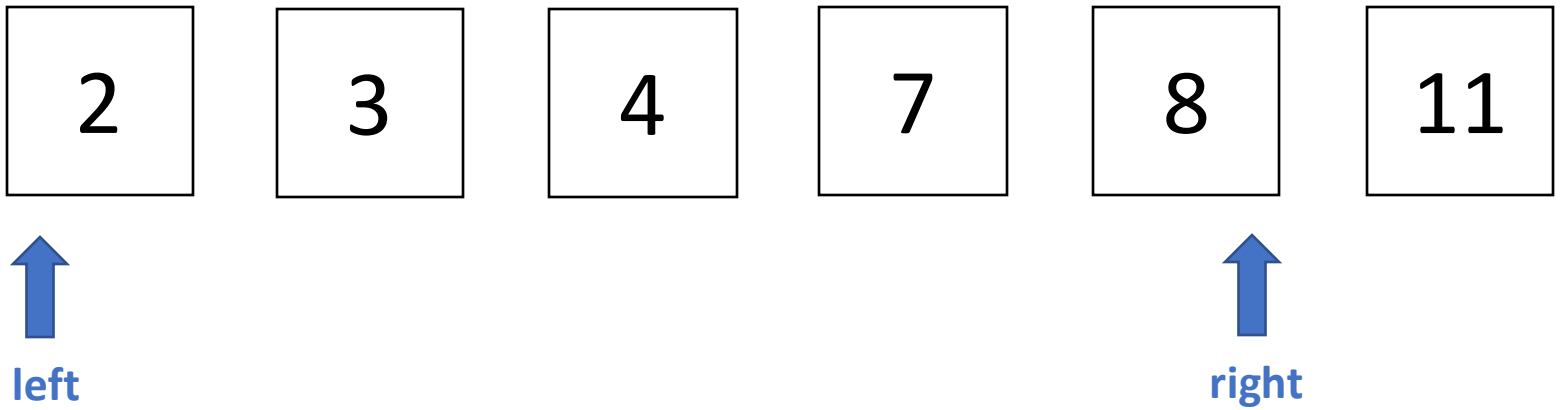


array [left] + array [right] =? sum

2 + 8 =? 12 **False**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

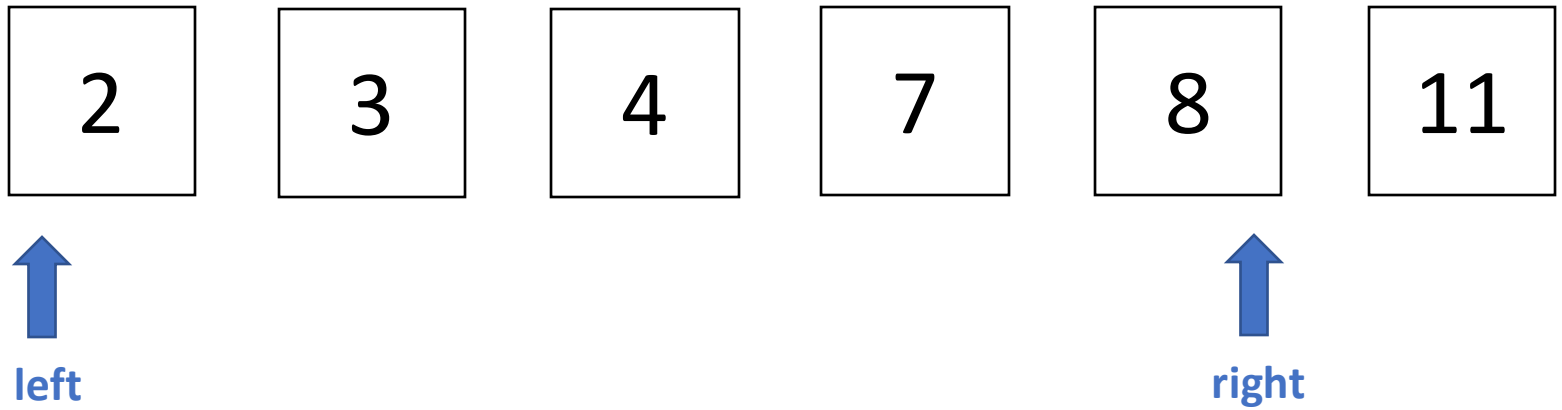


array [left] + array [right] =? sum

2 + 8 =? 12 **False** → array [left] + array [right] <? sum

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

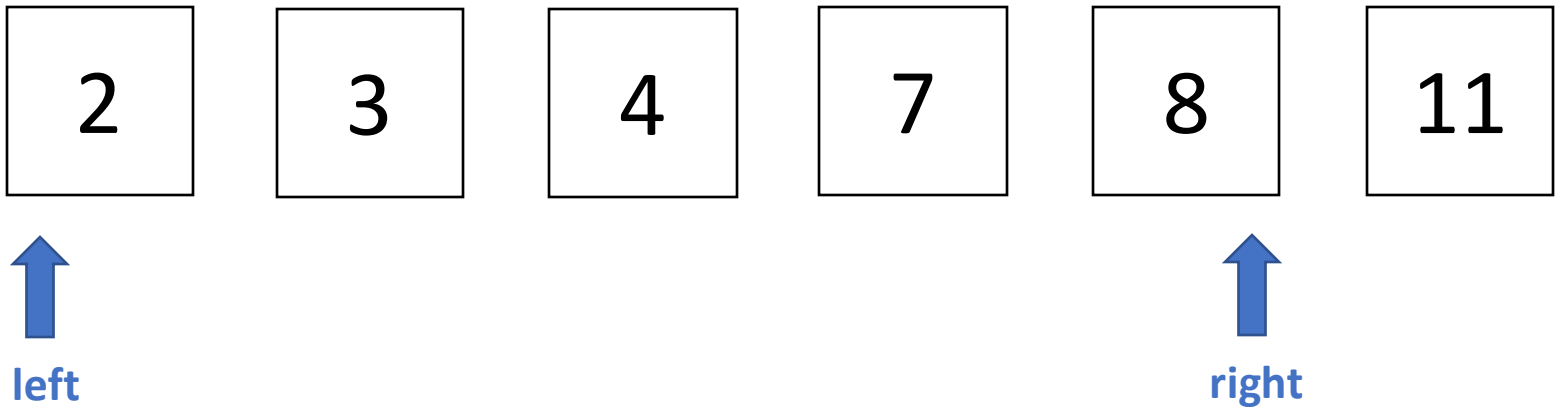


array [left] + array [right] =? sum

2 + 8 =? 12 **False** → array [left] + array [right] <? sum
2 + 8 <? 12

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



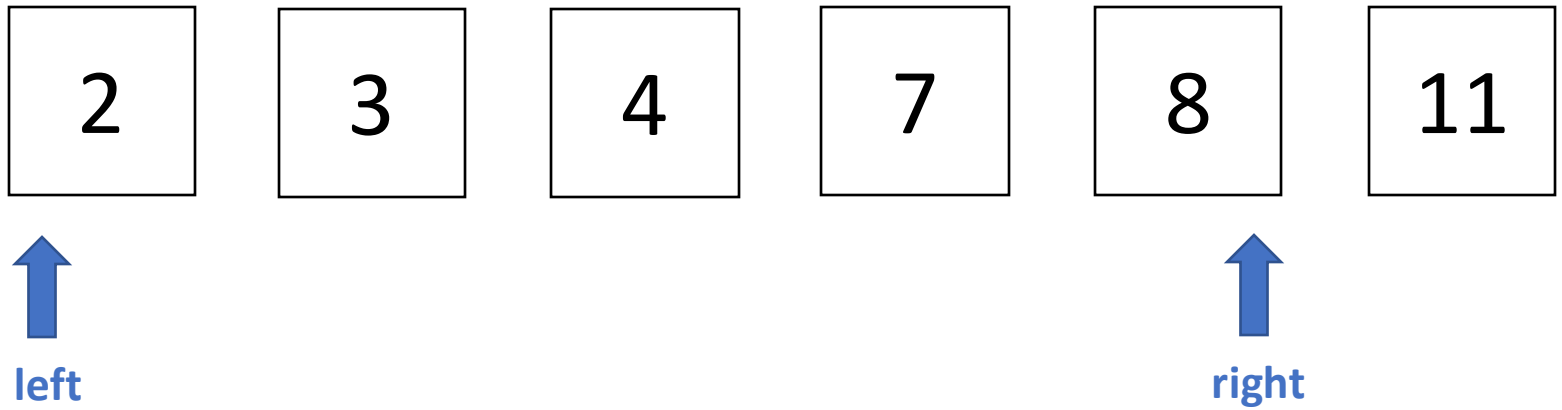
array [left] + array [right] =? sum

2 + 8 =? 12 **False** → array [left] + array [right] <? sum

2 + 8 <? 12 **True**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



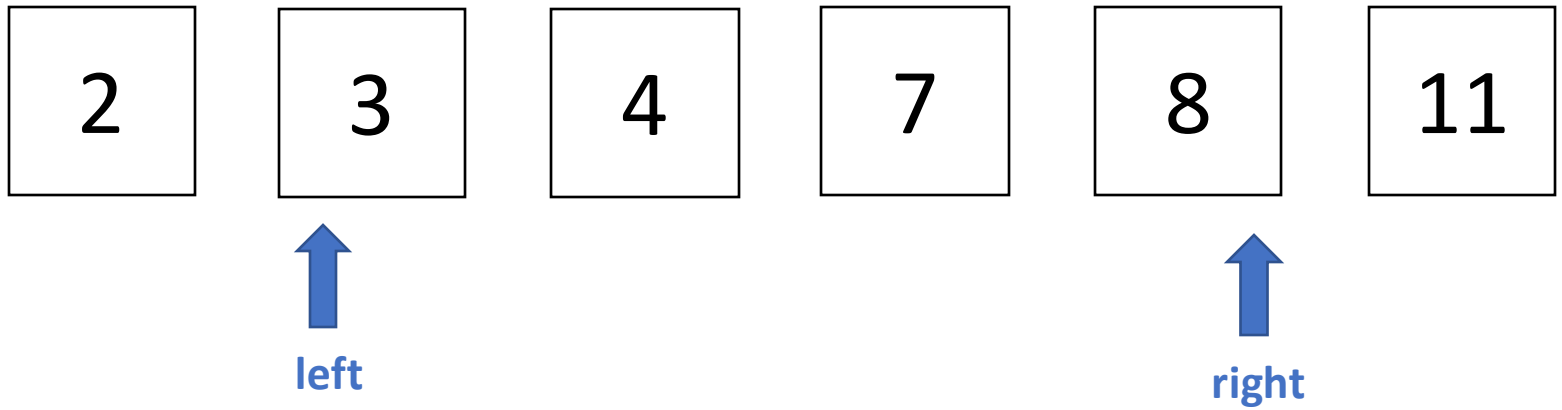
array [left] + array [right] =? sum

2 + 8 =? 12 **False** → array [left] + array [right] <? sum

2 + 8 <? 12 **True** → **then** update 'left' as 'left+1'

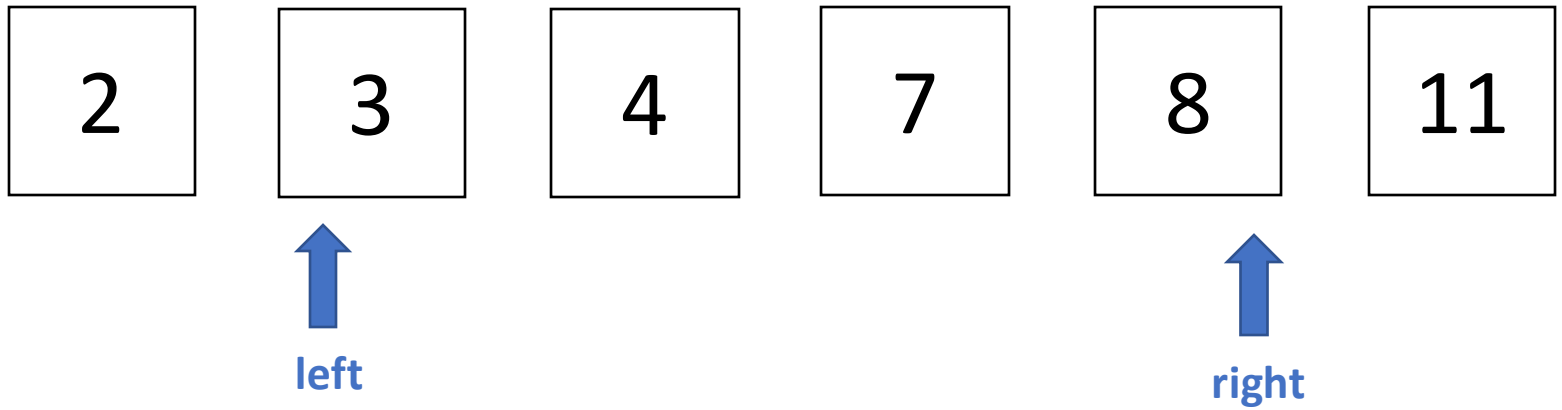
Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 2 (using sorting)

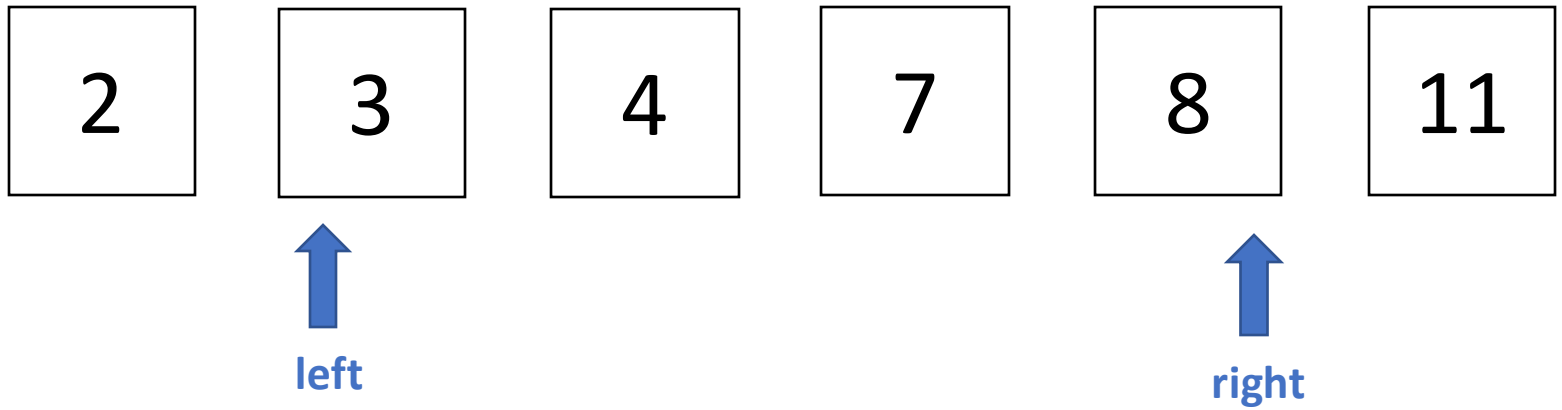
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



array [left] + array [right] =? sum

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

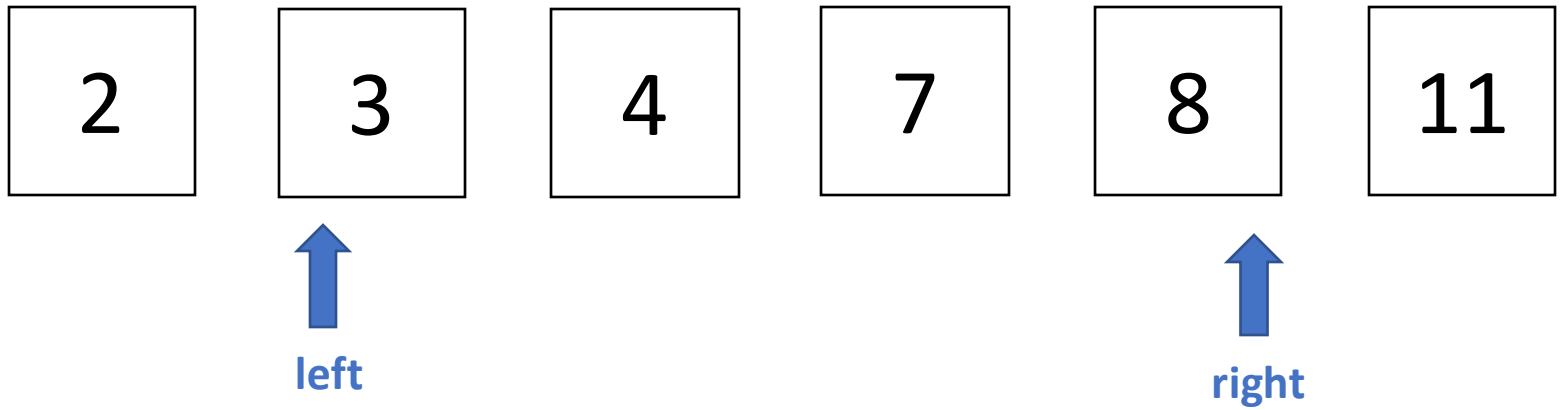


array [left] + array [right] =? Sum

3 + 8 =? 12

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

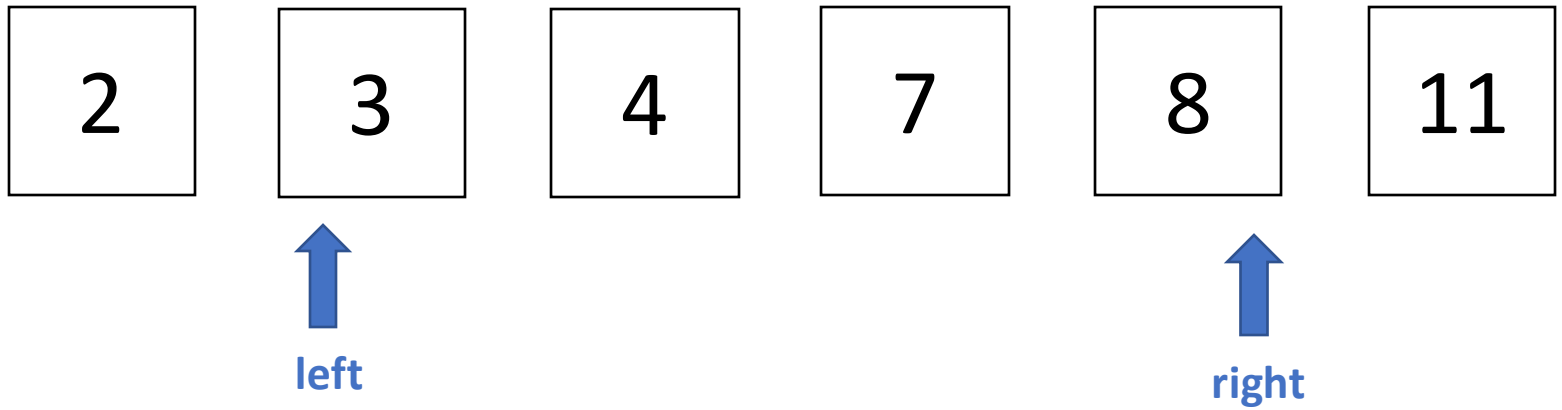


array [left] + array [right] =? Sum

3 + 8 =? 12 **False**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

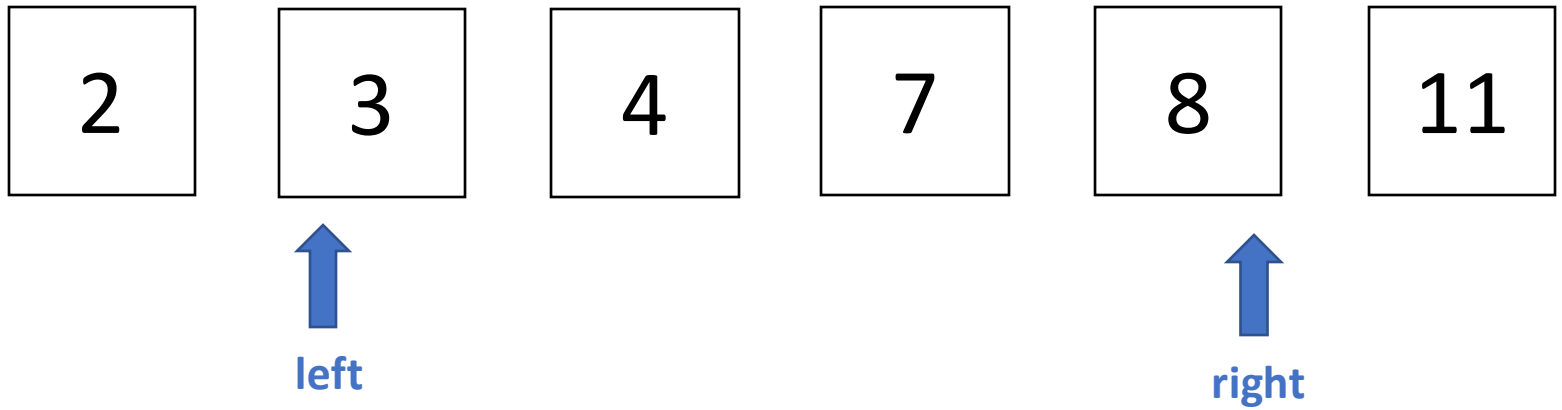


array [left] + array [right] =? Sum

3 + 8 =? 12 **False** → array [left] + array [right] <? sum

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

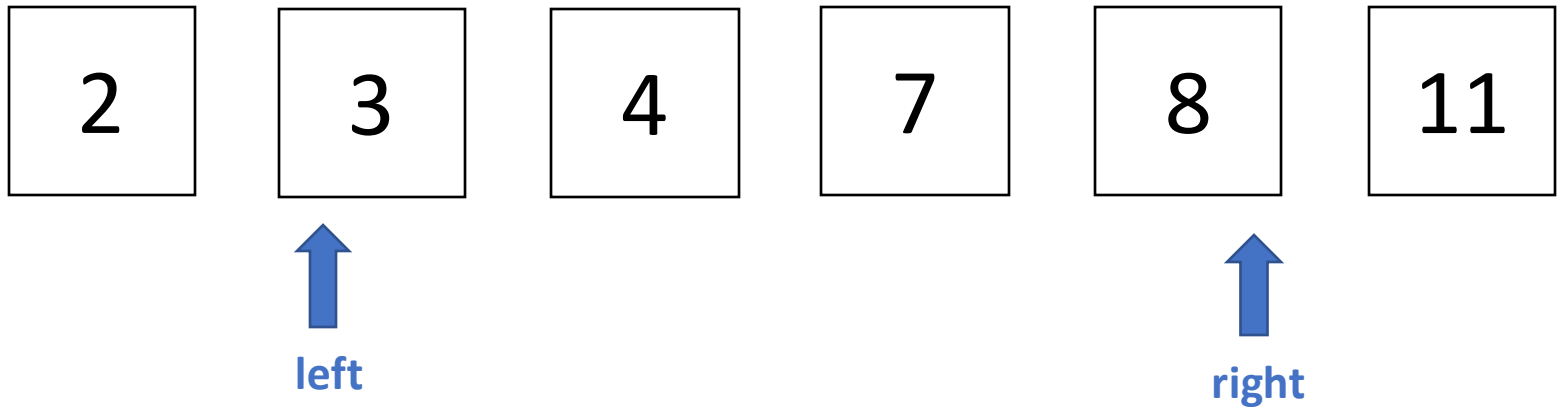


array [left] + array [right] =? Sum

3 + 8 =? 12 **False** → array [left] + array [right] <? sum
3 + 8 <? 11

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

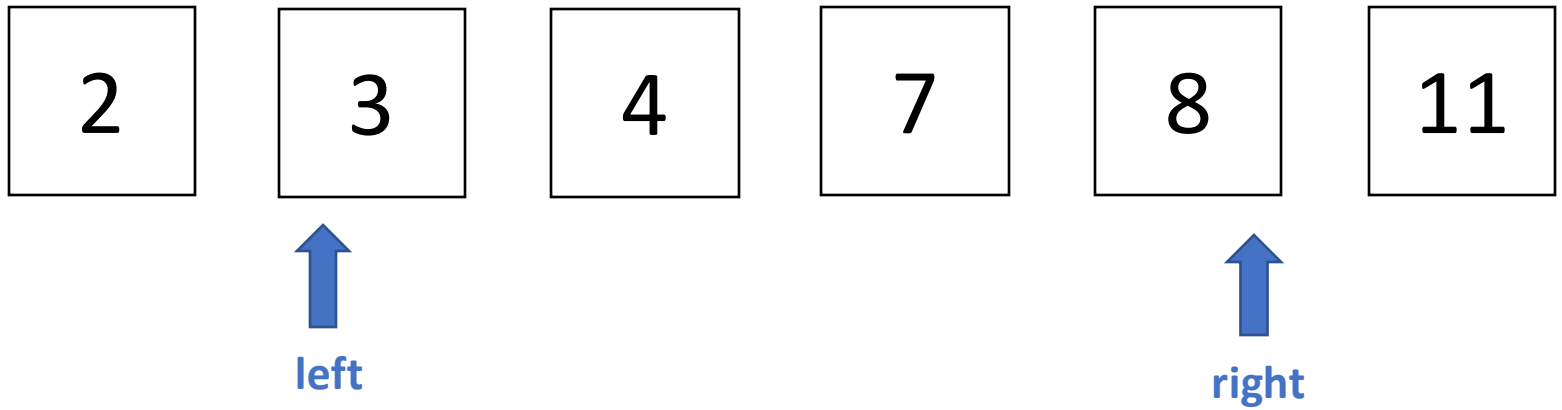


array [left] + array [right] =? Sum

3 + 8 =? 12 **False** → array [left] + array [right] <? sum
3 + 8 <? 11 **True**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



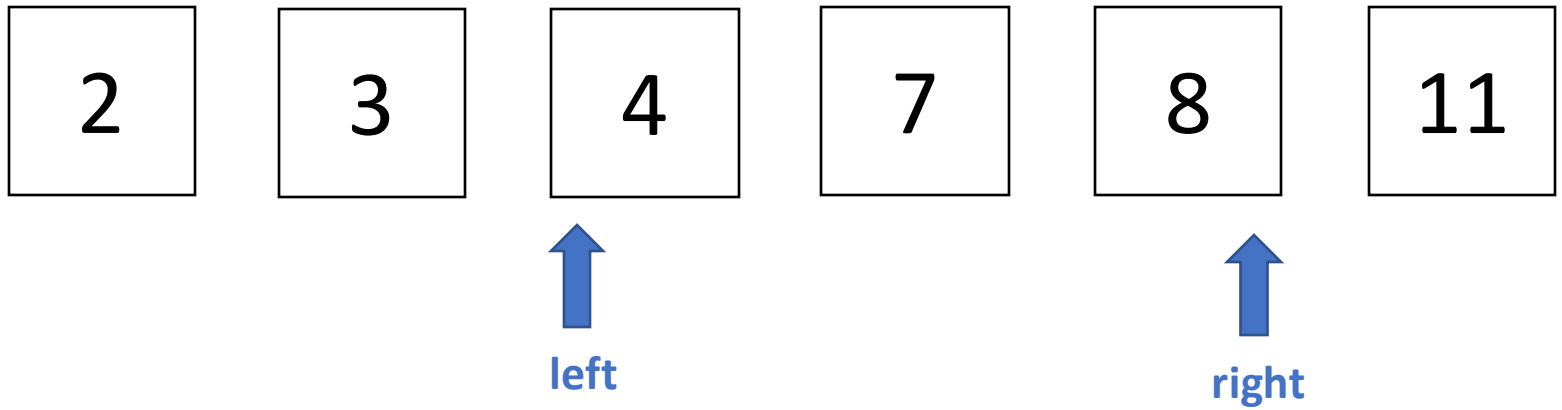
array [left] + array [right] =? Sum

3 + 8 =? 12 **False** → array [left] + array [right] <? sum

3 + 8 <? 11 **True** → **then** update 'left' as 'left+1'

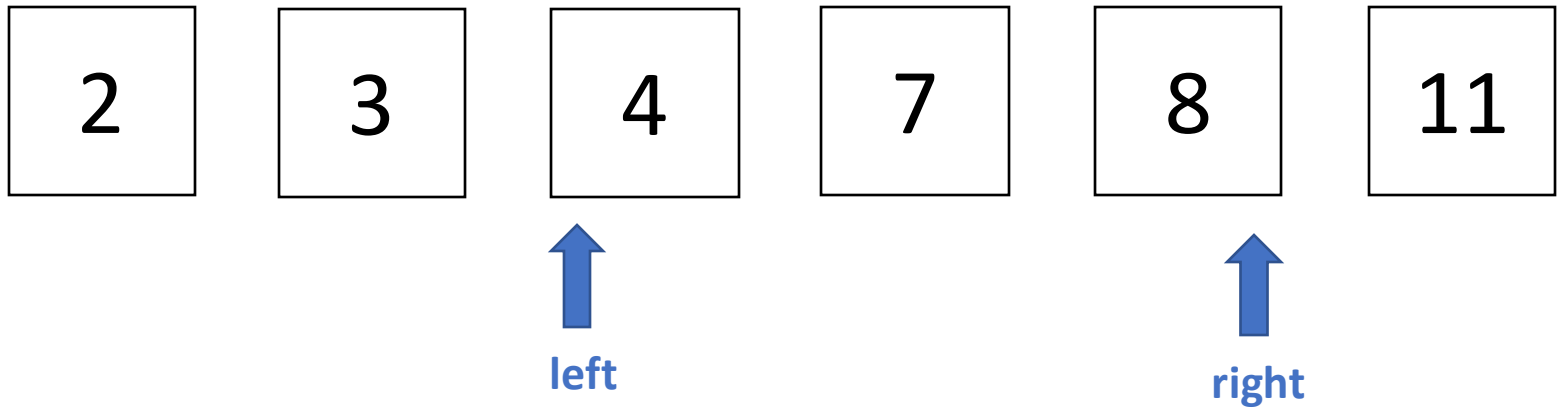
Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

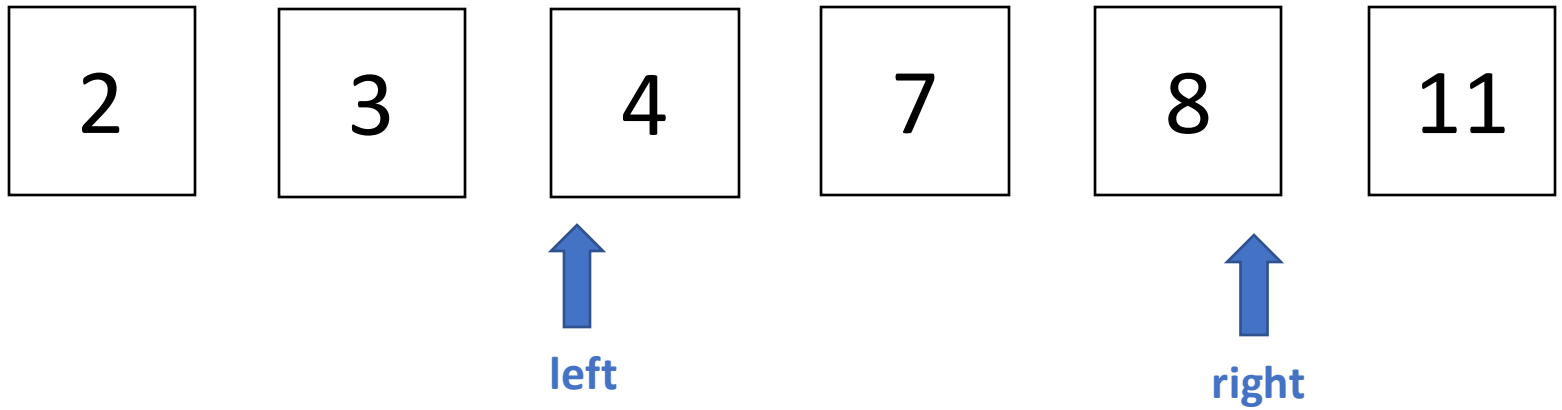


array [left] + array [right] =? Sum

4 + 8 =? 12

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

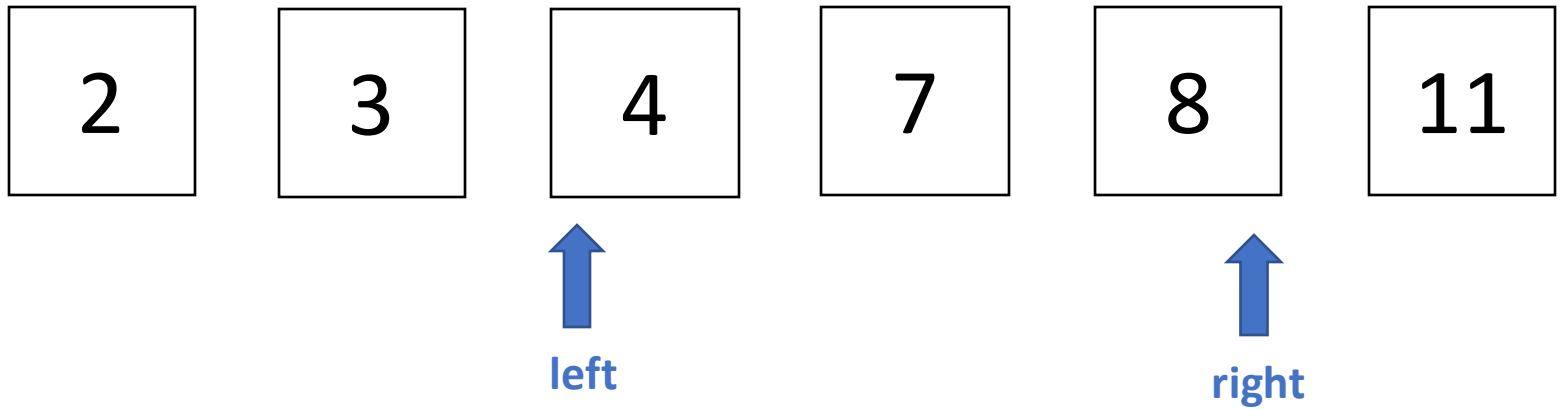


array [left] + array [right] =? Sum

4 + 8 =? 12 **True**

Solution 2 (using sorting)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



array [left] + array [right] =? Sum

4 + 8 =? 12 **True** → **then** return (4,8)

Solution 2 (using sorting)

Time complexity?

- In the best case, the algorithm returns the firstly compared pair. But it takes Ω(nlogn) time to sort the array, and that dominates the best case scenario. Therefore, best case time complexity is Ω(nlogn).

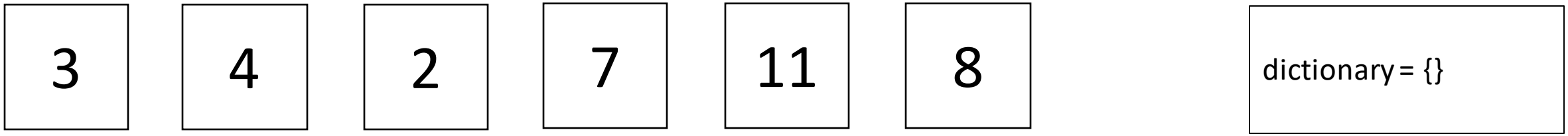
- In the worst case, there might not be such a pair in the array, or the pair might be the last one to be compared with sum. This comparisons will take O(n) time but again, sorting dominates the algorithm. Worst case scenario is the same as the best case: O(nlogn).

Solution 3 (hashing)

```
findPairs(array, sum):
    dictionary ← { }      # empty dictionary
    n ← length of the array
    for i from 0 to n:
        if sum – array[i] is in dictionary then
            return (array[i], sum-array[i])
        else
            append array[i] to dictionary
        end if
    end for
    return -1
```

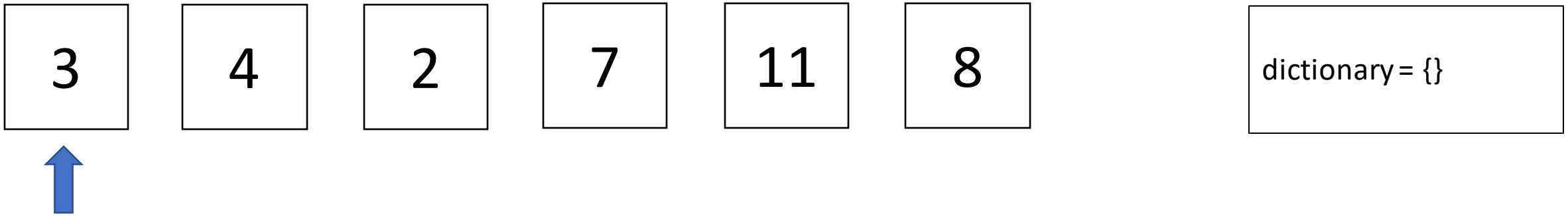
Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 3 (hashing)

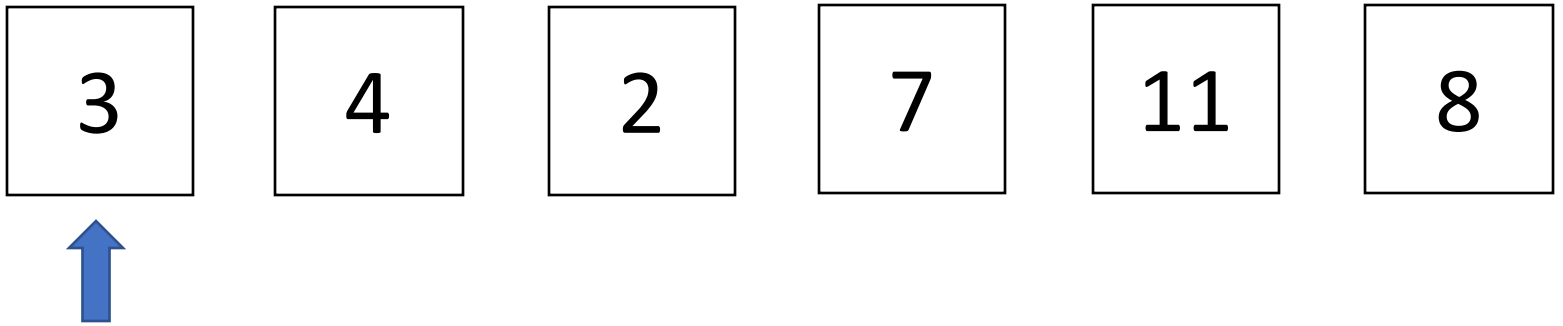
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



is 12 – 3 in dictionary?

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12

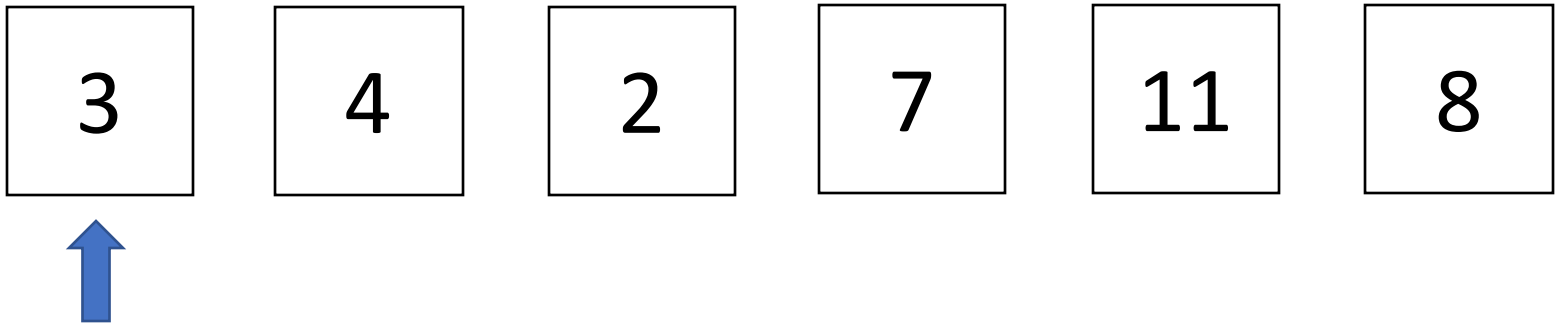


is 12 – 3 in dictionary?

False

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



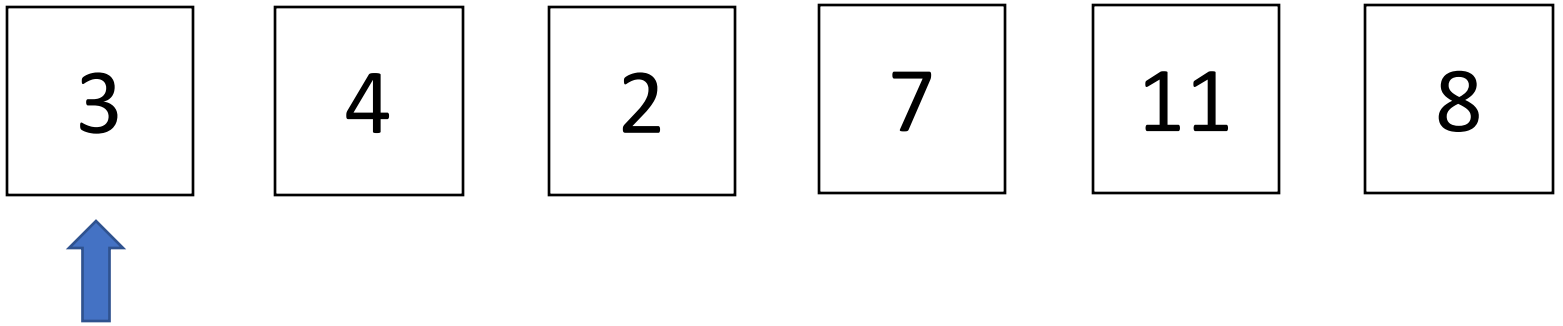
dictionary = {}

is 12 – 3 = 9 in dictionary?

False → **then** append 3 to dictionary

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



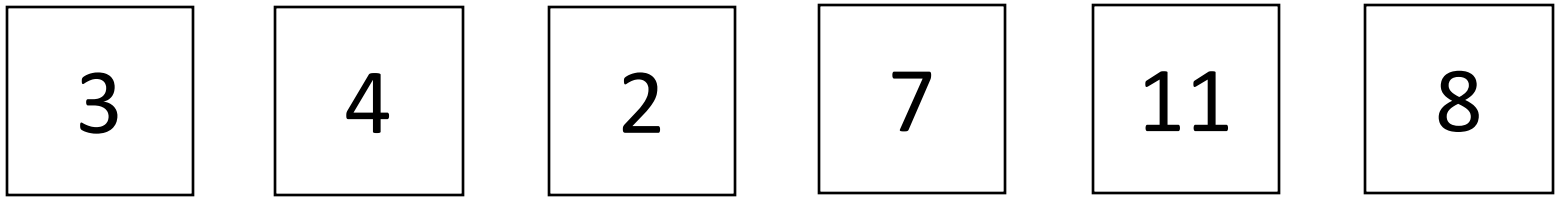
dictionary = {3}

is 12 – 3 = 9 in dictionary?

False → **then** append 3 to dictionary

Solution 3 (hashing)

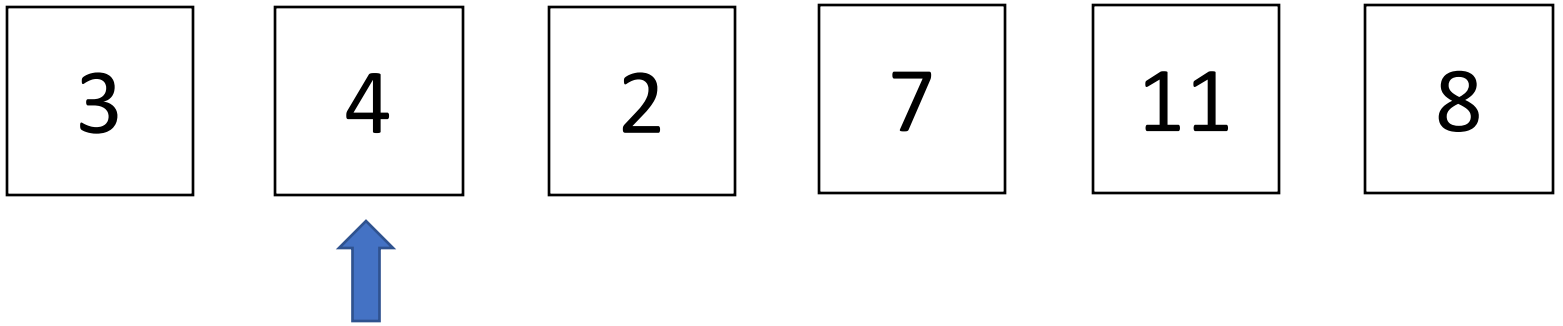
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



dictionary = {3}

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



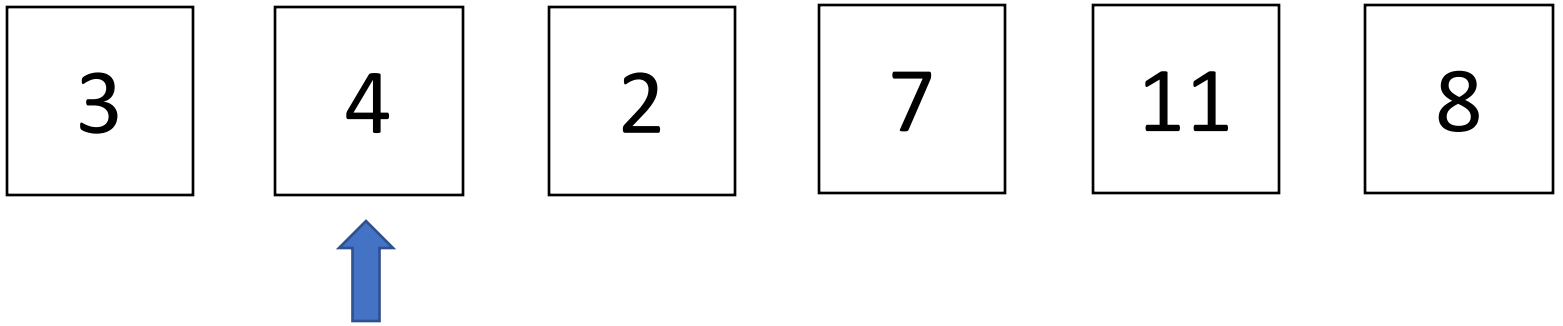
dictionary = {3}

is 12 – 4 = 8 in dictionary?

False

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



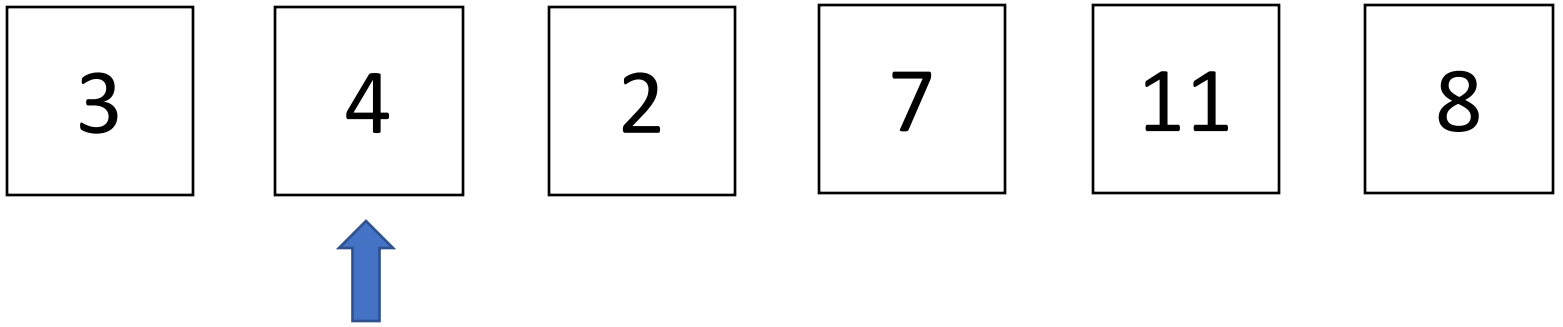
dictionary = {3}

is 12 – 4 = 8 in dictionary?

False → **then** append 4 to dictionary

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



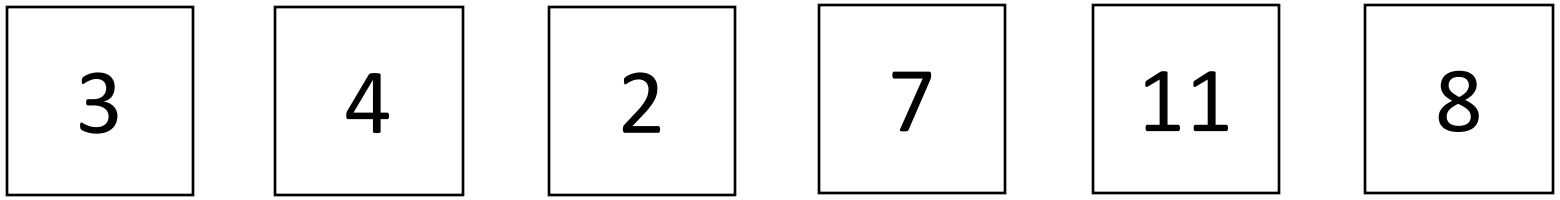
dictionary = {3, 4}

is 12 – 4 = 8 in dictionary?

False → **then** append 4 to dictionary

Solution 3 (hashing)

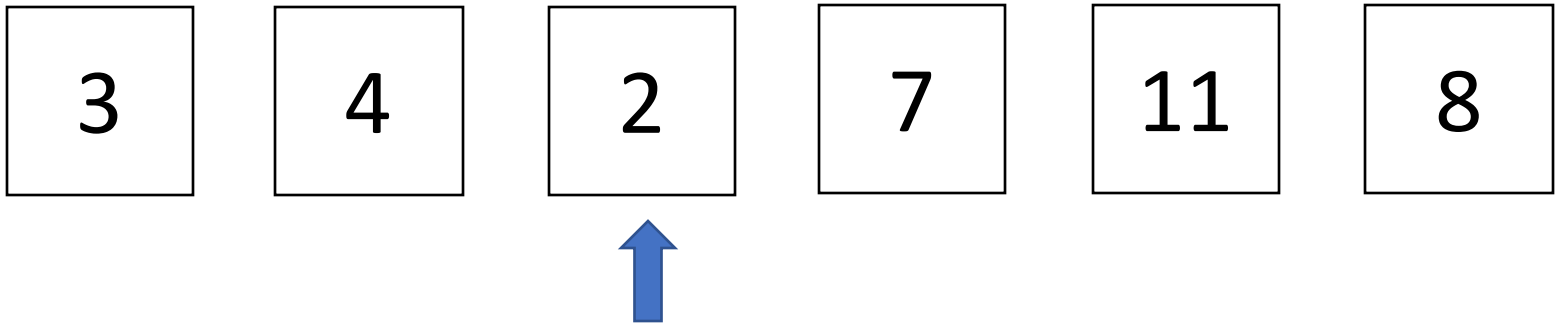
Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



dictionary = {3, 4}

Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



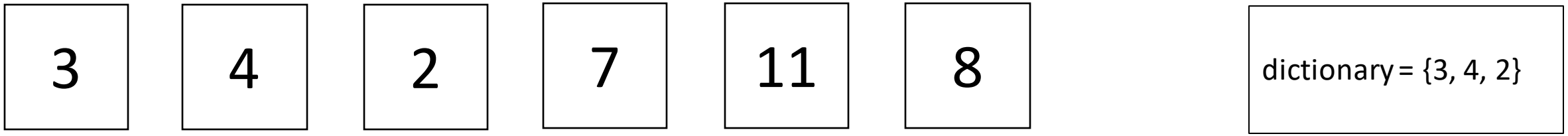
dictionary = {3, 4, 2}

is 12 – 2 = 10 in dictionary?

False → **then** append 2 to dictionary

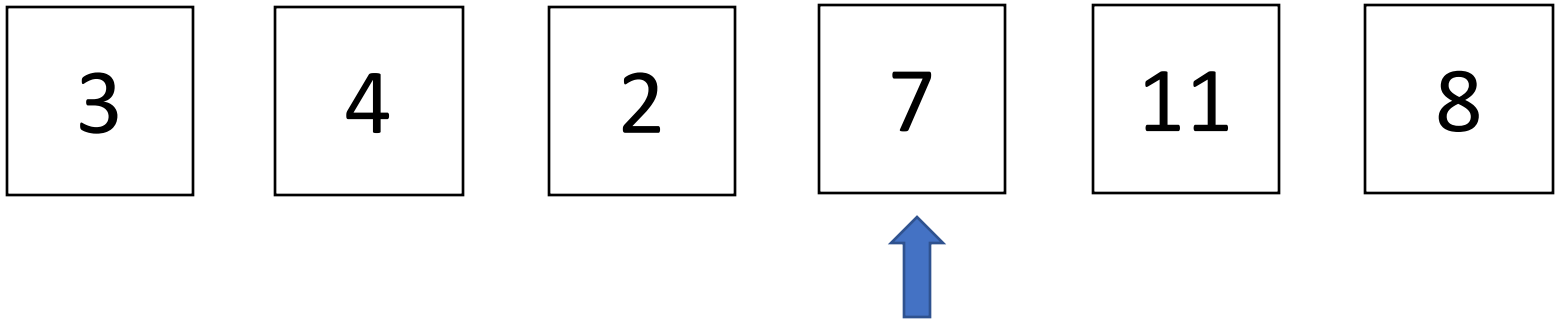
Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



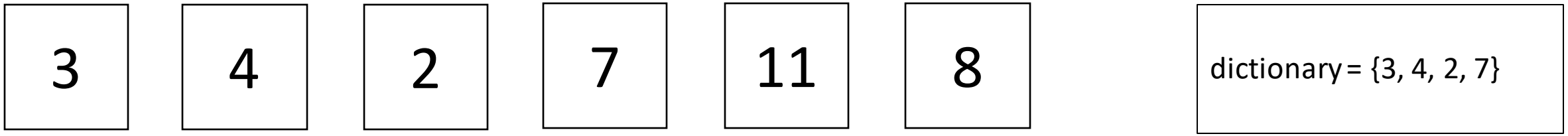
dictionary = {3, 4, 2, 7}

is 12 – 7 = 5 in dictionary?

False → **then** append 7 to dictionary

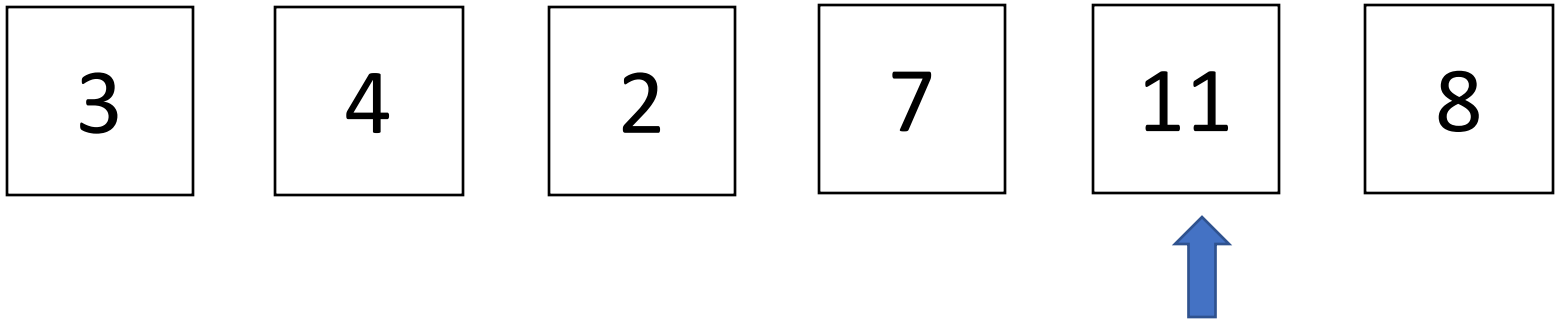
Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



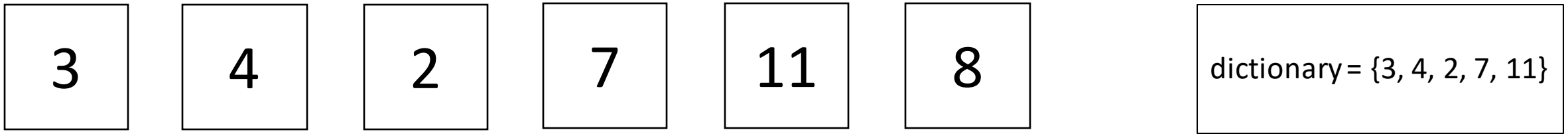
dictionary = {3, 4, 2, 7, 11}

is 12 – 11 = 1 in dictionary?

False → **then** append 11 to dictionary

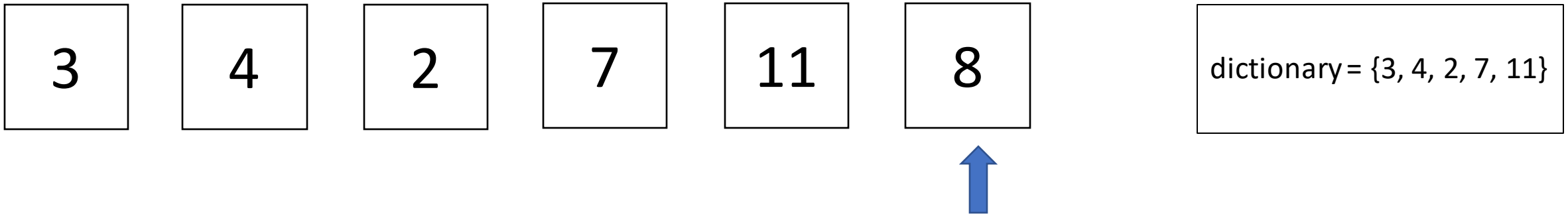
Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



Solution 3 (hashing)

Inputs:
array = {3, 4, 2, 7, 11, 8} and sum = 12



is 12 – 8 = 4 in dictionary?

True → **then** return (8, 4)

Solution 3 (hashing)

Time complexity?

- In the best case, the algorithm returns the firstly compared pair. That means the loop will iterate only once. Therefore, the best case complexity is $\Omega(1)$.

- In the worst case, there might not be such a pair in the array, or the pair might be the last one to be compared. Since we go over the array only once, we will make at most n comparisons and then return. Therefore, the worst time complexity is $O(n)$.

Comparison of the solutions

Solution	Best Case Time Complexity	Worst Case Time Complexity
Solution 1 (brute force)	Ω(1)	O(n^2)
Solution 2 (sorting)	Ω(nlogn)	O(nlogn)
Solution 3 (hashing)	Ω(1)	O(n)