

# 알튜비튜 투 포인터

두 개의 포인터로 배열을 빠르게 탐색하는 알고리즘입니다.  
코딩 테스트에선 주로 효율성을 보는 문제에 활용됩니다.

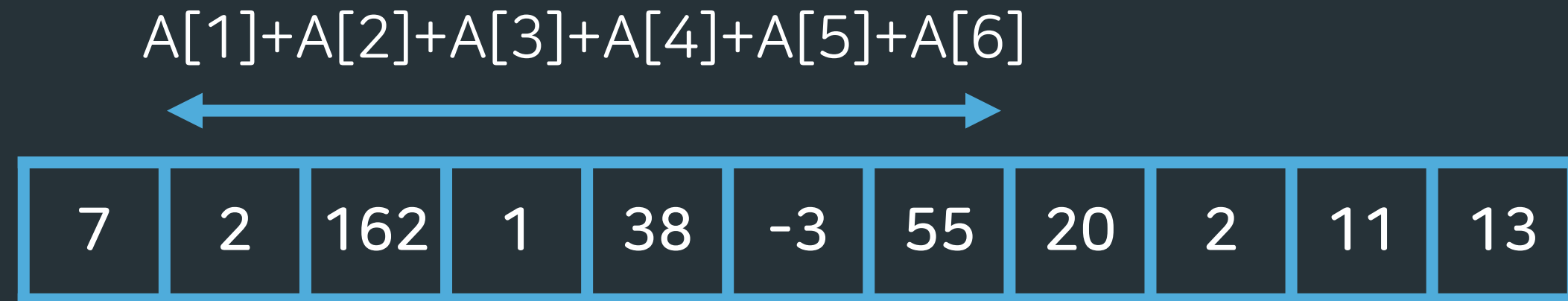
이와 더불어 투 포인터와 함께 자주 활용되는 누적 합, 슬라이딩 윈도우에 대해서도 알아봅니다.

이런 문제가 있다고 해봅시다.

7	2	162	1	38	-3	55	20	2	11	13
---	---	-----	---	----	----	----	----	---	----	----

“배열  $A$ 가 있을 때,  $A[i]$ 부터  $A[j]$ 까지의 합은?”  
( $i \leq j$ )

## 그냥 더해도 되지만...



더할 때마다 시간 복잡도  $O(n)$   
만약 배열의 크기가 10,000일 때,  
구하고자 하는 구간이 1,000,000개만 돼도...  
총 연산 횟수 100억

# 연산 횟수를 줄일 수 있는 방법은 없나?

sum

7	2	162	1	38	-3	55	20	2	11	13

# 연산 횟수를 줄일 수 있는 방법은 없나?

	7	2	162	1	38	-3	55	20	2	11	13
sum	7										
A[0]											

# 연산 횟수를 줄일 수 있는 방법은 없나?

	7	2	162	1	38	-3	55	20	2	11	13
sum	7	9									
	A[0]+A[1]										

# 연산 횟수를 줄일 수 있는 방법은 없나?

	7	2	162	1	38	-3	55	20	2	11	13
sum	7	9	171								

$A[0]+A[1]+A[2]$   
 $= \text{sum}[1]+A[2]$

# 연산 횟수를 줄일 수 있는 방법은 없나?

	7	2	162	1	38	-3	55	20	2	11	13
sum	7	9	171	172	210	207	262	282	284	295	308

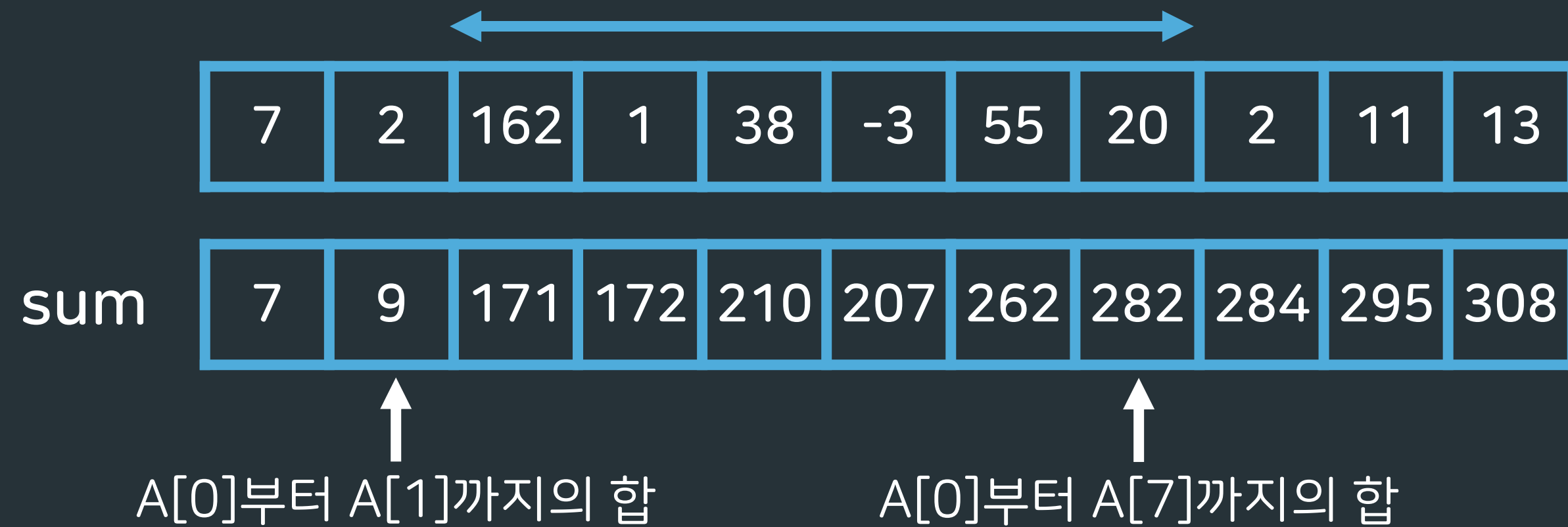


## A[0]부터 A[i]까지의 합

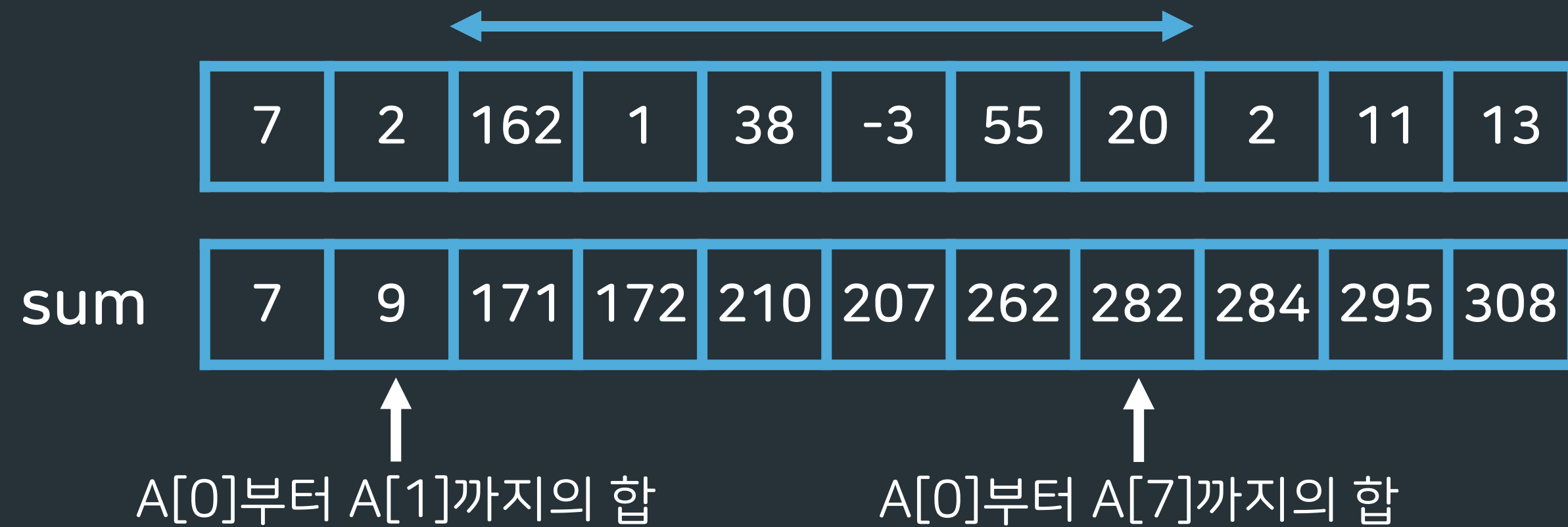
	7	2	162	1	38	-3	55	20	2	11	13
sum	7	9	171	172	210	207	262	282	284	295	308

↑  
A[0]부터 A[5]까지의 합

## A[i]부터 A[j]까지의 합

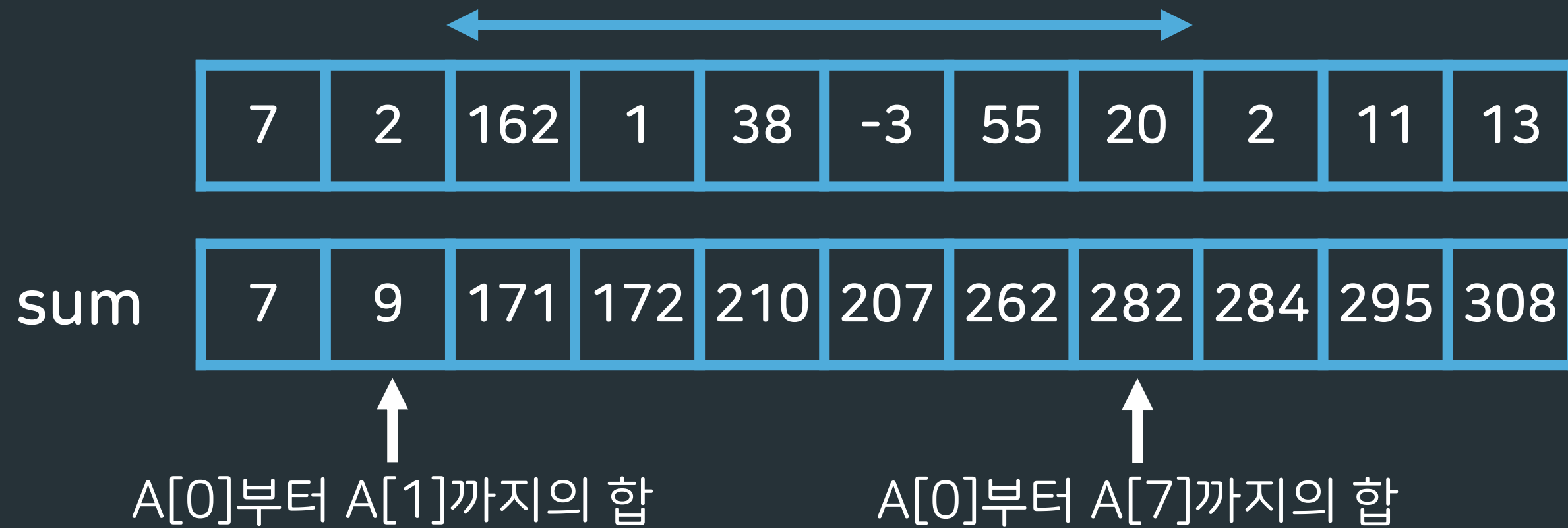


## A[i]부터 A[j]까지의 합



$\text{sum}[7] - \text{sum}[1] = A[2] \text{부터 } A[7] \text{까지의 합}$

# A[i]부터 A[j]까지의 합



$$\text{sum}[j] - \text{sum}[i-1] = A[i] \text{부터 } A[j] \text{까지의 합}$$

## /<> 11659번 : 구간 합 구하기 4 – Silver 3

### 문제

- 수 N개가 주어질 때, i번째 수부터 j번째 수까지의 합은?

### 제한 사항

- N, M은  $1 \leq N, M \leq 100,000$
- 입력되는 정수 k는  $1 \leq k \leq 1,000$

### 예제 입력

```
5 3
5 4 3 2 1
1 3
2 4
5 5
```

### 예제 출력

```
12
9
1
```

## /<> 11659번 : 구간 합 구하기 4 – Silver 3

### 문제

- 수 N개가 주어질 때, i번째 수부터 j번째 수까지의 합은?

### 제한 사항

- N, M은  $1 \leq N, M \leq 100,000$
- 입력되는 정수 k는  $1 \leq k \leq 1,000$

순차 탐색으로 구현하면  
최대 연산 횟수  $100,000 * 100,000 (=100\text{억})$ 으로 시간초과

### 예제 입력

```
5 3
5 4 3 2 1
1 3
2 4
5 5
```

### 예제 출력

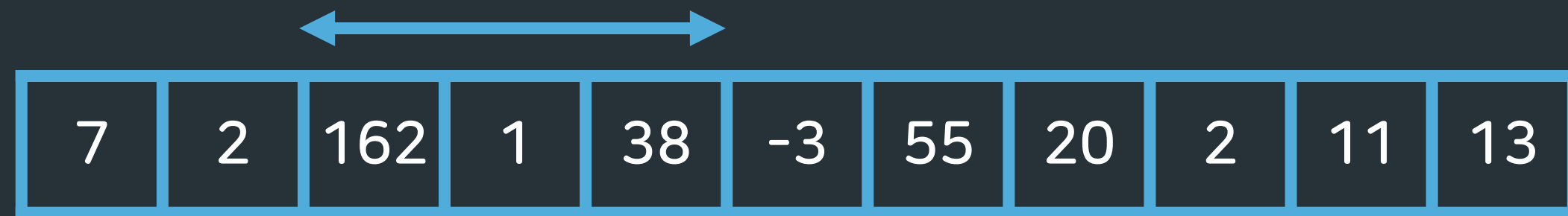
```
12
9
1
```

## 아까 봤던 문제를 살짝 바꿔볼게요

7	2	162	1	38	-3	55	20	2	11	13
---	---	-----	---	----	----	----	----	---	----	----

“배열  $A$ 가 있을 때,  $A[i]$ 부터  $A[j]$ 까지의 합은?”  
( $j - i = k$ )

## 아까 봤던 문제를 살짝 바꿔볼게요



“배열  $A$ 가 있을 때,  $A[i]$ 부터  $A[j]$ 까지의 합은?”  
( $j - i = 2$ )



## 아까 봤던 문제를 살짝 바꿔볼게요



“배열  $A$ 가 있을 때,  $A[i]$ 부터  $A[j]$ 까지의 합은?”  
( $j - i = 2$ )

## 아까 봤던 문제를 살짝 바꿔볼게요



“배열  $A$ 가 있을 때,  $A[i]$ 부터  $A[j]$ 까지의 합은?”  
( $j - i = 2$ )

누적 합은 아까 해봤으니까!



누적 합을 사용하지 않고 구간 합을 빠르게 구하는 방법은?

공통되는 부분이 보이시나요?



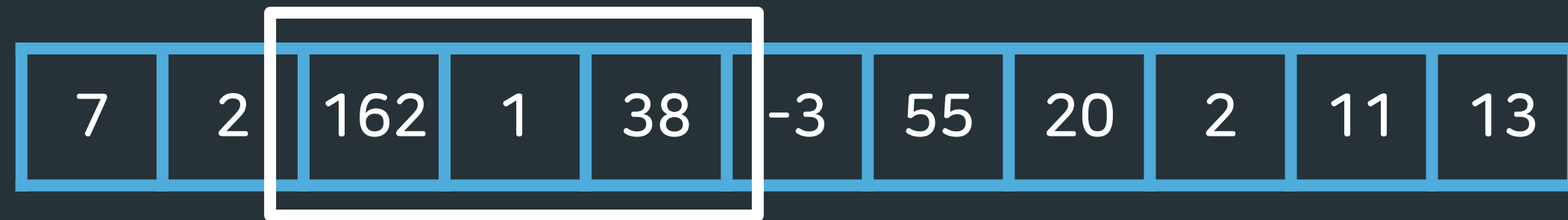
$A[0]+A[1]+A[2]$

공통되는 부분이 보이시나요?



$A[1]+A[2]+A[3]$

공통되는 부분이 보이시나요?



$A[2]+A[3]+A[4]$

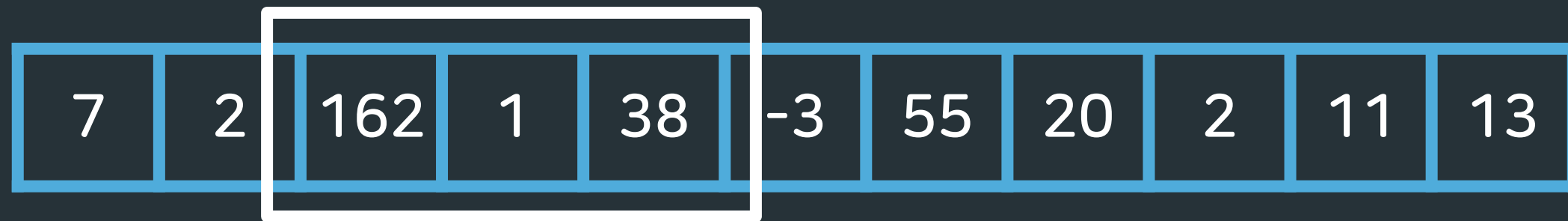


$$A[0] + A[1] + A[2] = k0$$



$$\begin{aligned} &A[1]+A[2]+A[3] \\ &= k1 = k0-A[0]+A[3] \end{aligned}$$





$$\begin{aligned} &A[2]+A[3]+A[4] \\ &= k2 = k1-A[1]+A[4] \end{aligned}$$

## /<> 21921번 : 블로그 - Silver 3

### 문제

- N일간의 방문자 수가 주어진다.
  - 연속된 X일 동안 가장 많이 들어온 방문자 수와 그 기간의 수는 몇 개인가?
- \* 최대 방문자 수가 0명이라면 SAD를 출력

### 제한 사항

- N, X는  $1 \leq X \leq N \leq 250,000$
- 방문자 수 k는  $0 \leq k \leq 8,000$

예제 입력 1

```
5 2
1 4 2 5 1
```

예제 입력 2

```
7 5
1 1 1 1 1 5 1
```

예제 입력 3

```
5 3
0 0 0 0 0
```

예제 출력 1

```
7
1
```

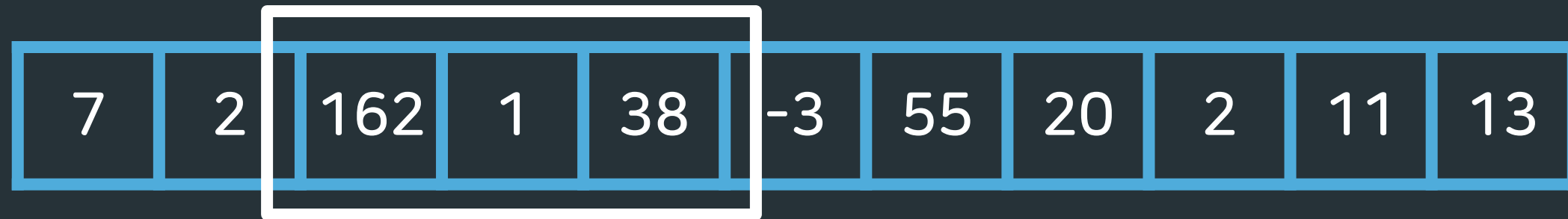
예제 출력 2

```
9
2
```

예제 출력 3

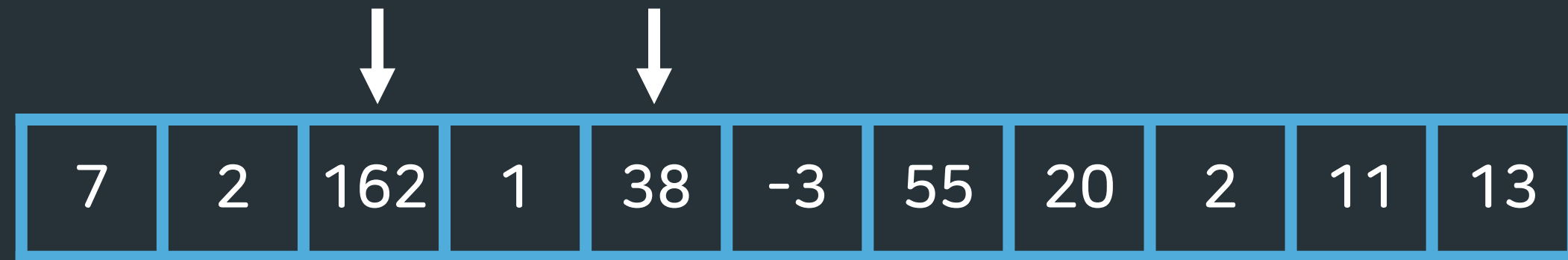
```
SAD
```

이제 본론에 들어가봅시다!



슬라이딩 윈도우

이제 본론에 들어가봅시다!



투 포인터

## Two Pointer

- 2개의 포인터로 배열을 탐색하며 빠르게 답을 찾는 알고리즘
- 주로 반복문(while)으로 구현
- 일반적으로 시간 복잡도  $O(n^2)$ 의 문제를 시간 복잡도  $O(n)$ 로 풀 수 있음
- 투 포인터 탐색 방법은 크게 2개로 나눌 수 있음
  1. 2개의 포인터가 다른 위치에서 시작하여 서로에게 다가가는 방향으로 탐색
  2. 2개의 포인터가 같은 위치에서 시작하여 같은 방향으로 이동하며 탐색
- 1번 방식은 일반적으로 배열이 정렬됐을 때에만 성립하는 경우가 많음
- 슬라이딩 윈도우는 2개의 포인터 사이의 거리를 고정하고, 2번 방식으로 탐색한 것과 같음

## /<> 2470번 : 두 용액 - Gold 5

### 문제

- 두 개의 서로 다른 용액을 혼합해, 합이 0에 가까운 용액을 만들어라

### 제한 사항

- 용액의 수 N은  $2 \leq N \leq 100,000$
- 용액의 특성값 k는  $-1e9 \leq k \leq 1e9$  (-10억 ~ 10억)

### 예제 입력

```
5
-2 4 -99 -1 98
```

### 예제 출력

```
-99 98
```

정렬이 됐다고 치면...  
맨 왼쪽에는 가장 작은 값이 존재  
맨 오른쪽에는 가장 큰 값이 존재  
오른쪽으로 갈 수록 값이 커지고  
왼쪽으로 갈 수록 값이 작아진다



## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = 76$$

Ans = 76

## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = 76$$

$$\text{Ans} = 76$$

0보다 크니까 숫자를 줄이자!

## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = 74$$

Ans = 74

## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = 37$$

Ans = 37

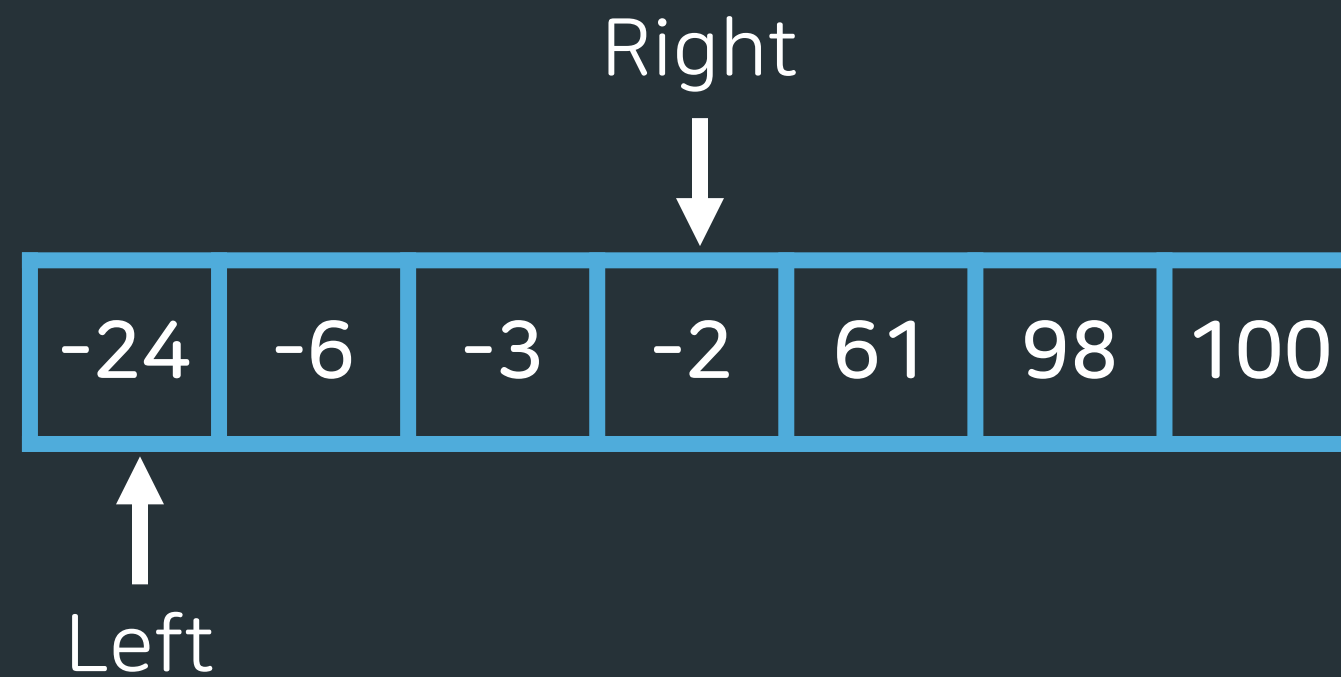
## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = -26$$

$$\text{Ans} = -26$$

## 다른 위치에서 시작하는 투 포인터

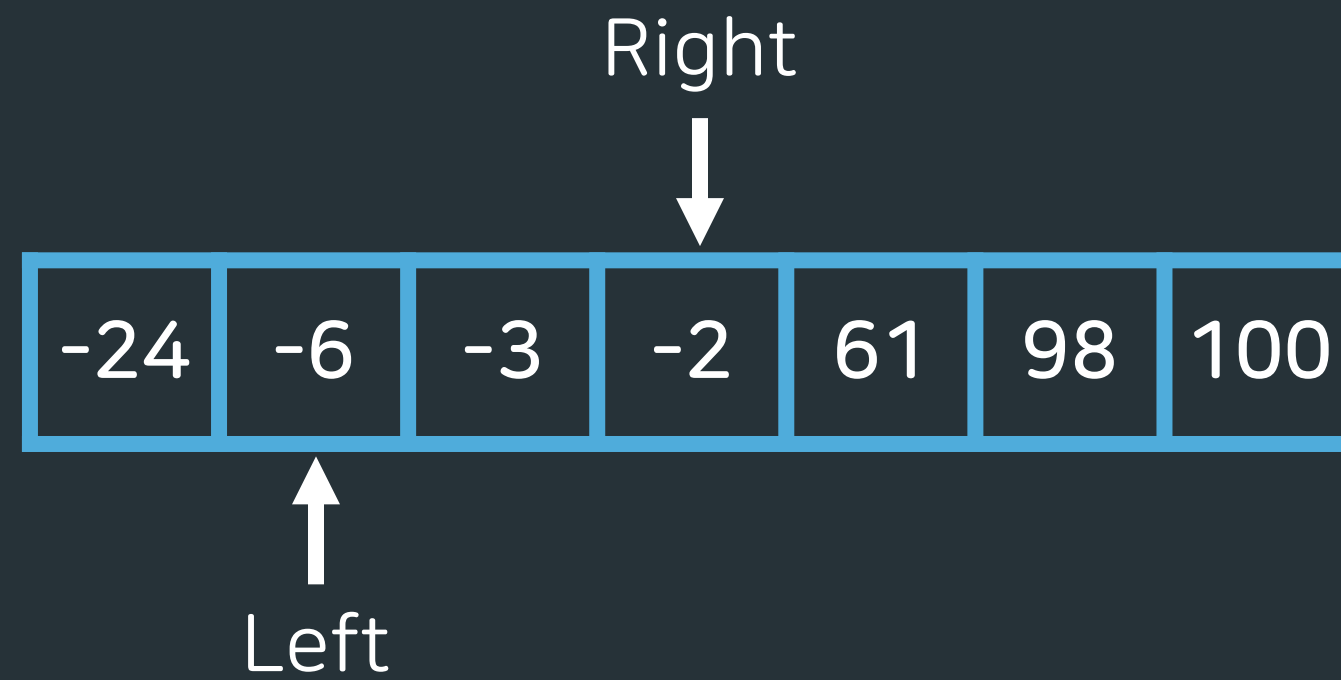


$$\text{Left} + \text{Right} = -26$$

$$\text{Ans} = -26$$

0보다 작으니까 숫자를 키우자!

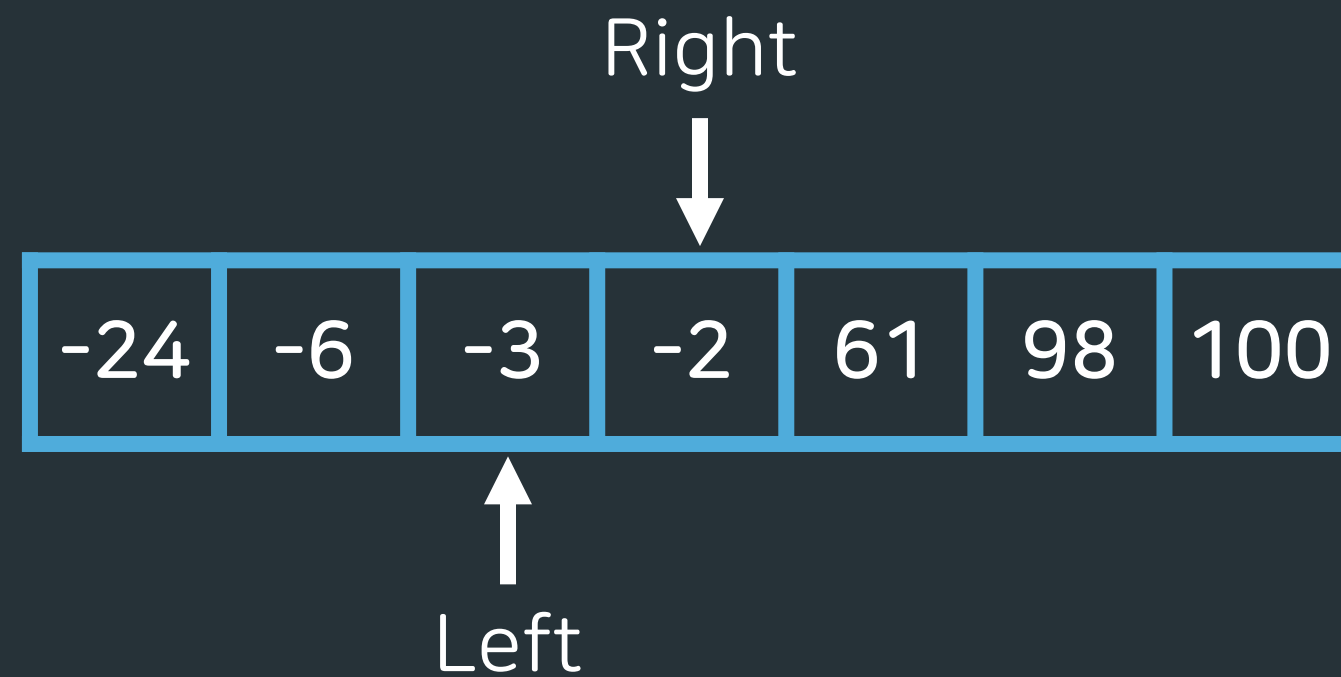
## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = -8$$

$$\text{Ans} = -8$$

## 다른 위치에서 시작하는 투 포인터

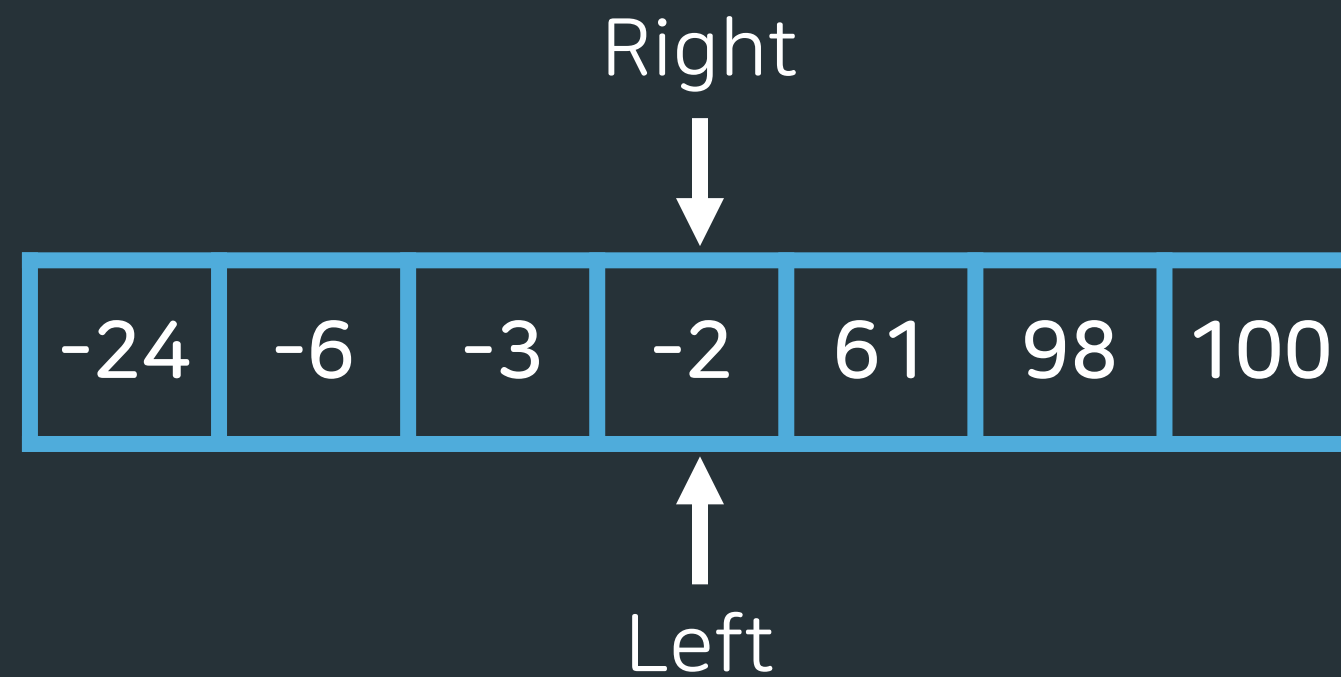


$$\text{Left} + \text{Right} = -5$$

$$\text{Ans} = -5$$



## 다른 위치에서 시작하는 투 포인터



$$\text{Left} + \text{Right} = -5$$

$$\text{Ans} = -5$$

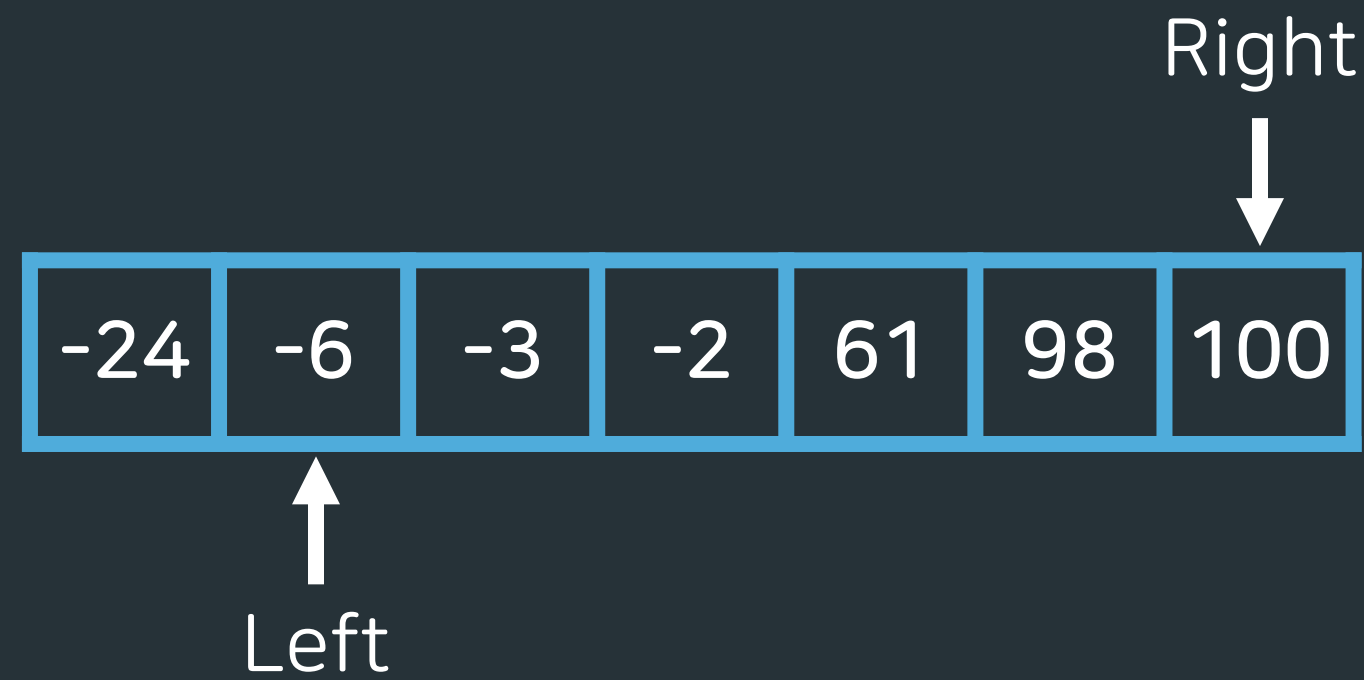
서로 다른 두 용액이어야 하므로 break

정말 모든 경우가 탐색된 건가요?



$$\text{Left} + \text{Right} = 76$$

정말 모든 경우가 탐색된 건가요?



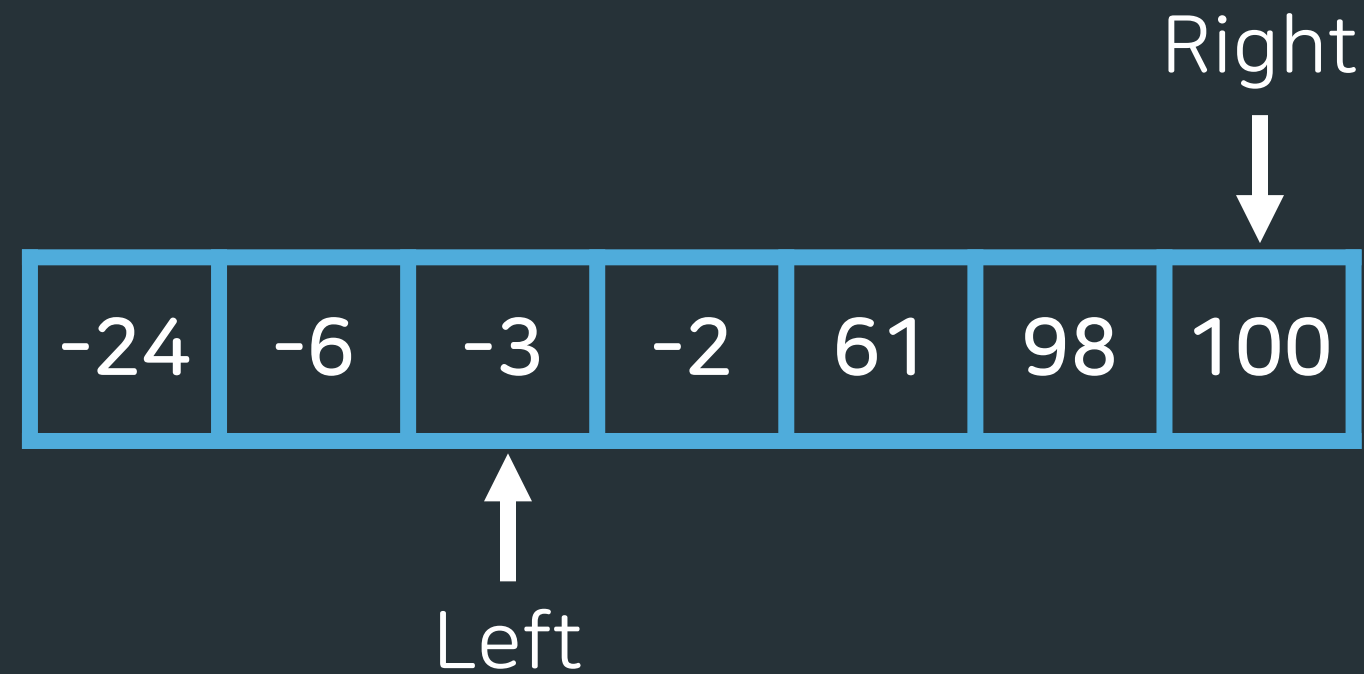
$$\text{Left} + \text{Right} = 94$$

정말 모든 경우가 탐색된 건가요?



$$\text{Left} + \text{Right} = 97$$

정말 모든 경우가 탐색된 건가요?



$$\text{Left} + \text{Right} = 97$$

76보다 큰 값임이 보장되기 때문에  
Left 포인터를 옮길 필요 없음

## /<> 1644번 : 소수의 연속합 - Gold 3

### 문제

- 자연수 N이 주어진다
- 연속된 소수의 합이 N이 되는 경우의 수는?

### 제한 사항

- N은  $1 \leq N \leq 4,000,000$

### 예제 입력 1

20

### 예제 입력 2

41

### 예제 출력 1

0

### 예제 출력 2

3

## /<> 1644번 : 소수의 연속합 - Gold 3

### 문제

- 자연수 N이 주어진다
- 연속된 소수의 합이 N이 되는 경우의 수는?

### 제한 사항

- N은  $1 \leq N \leq 4,000,000$

연속된 소수?  
연속된 배열의 원소!

### 예제 입력 1

20

### 예제 입력 2

41

### 예제 출력 1

0

### 예제 출력 2

3

## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41



Left ~ Right = 2



## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41



Left ~ Right = 2

41보다 작으니까 범위를 늘리자!

## 같은 위치에서 시작하는 두 포인터

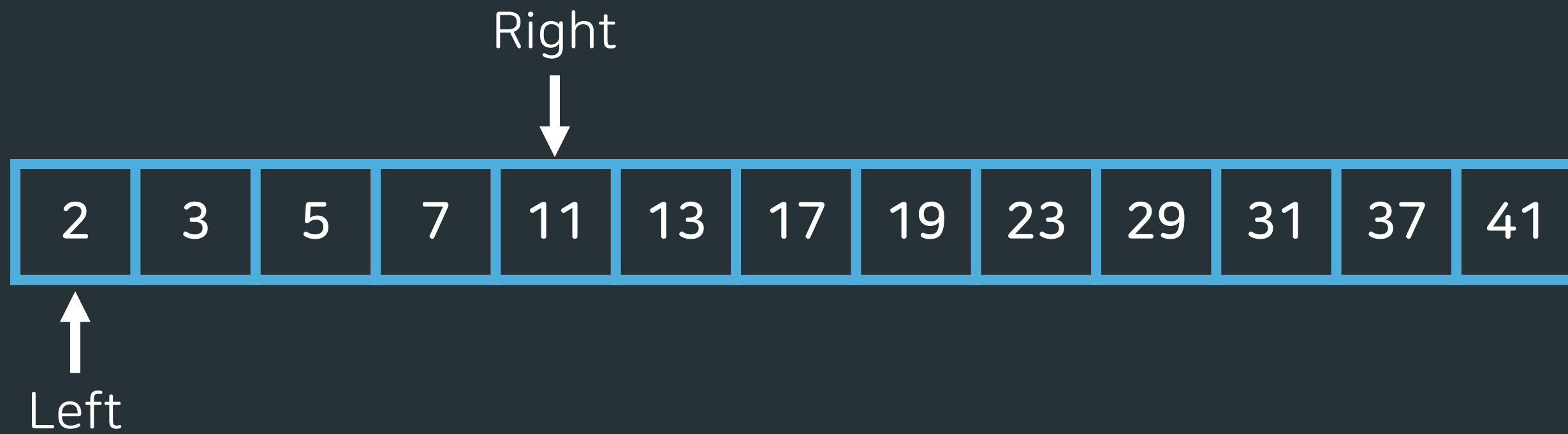
- (예제) 연속된 소수의 합이 41



Left ~ Right = 5

## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41



Left ~ Right = 28

## 같은 위치에서 시작하는 두 포인터

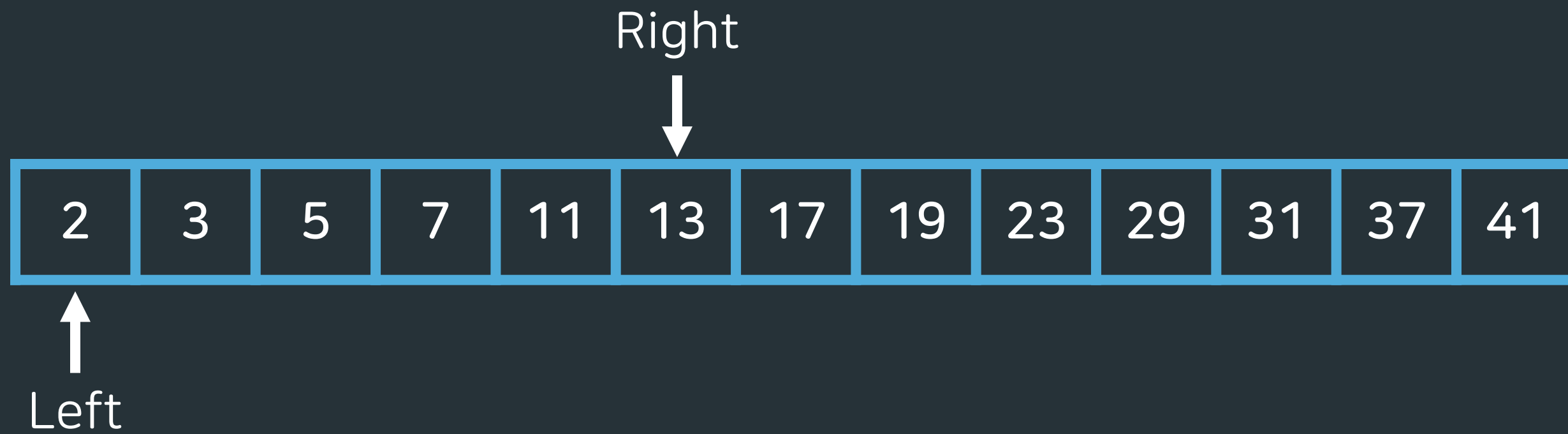
- (예제) 연속된 소수의 합이 41



Left ~ Right = 41

## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41

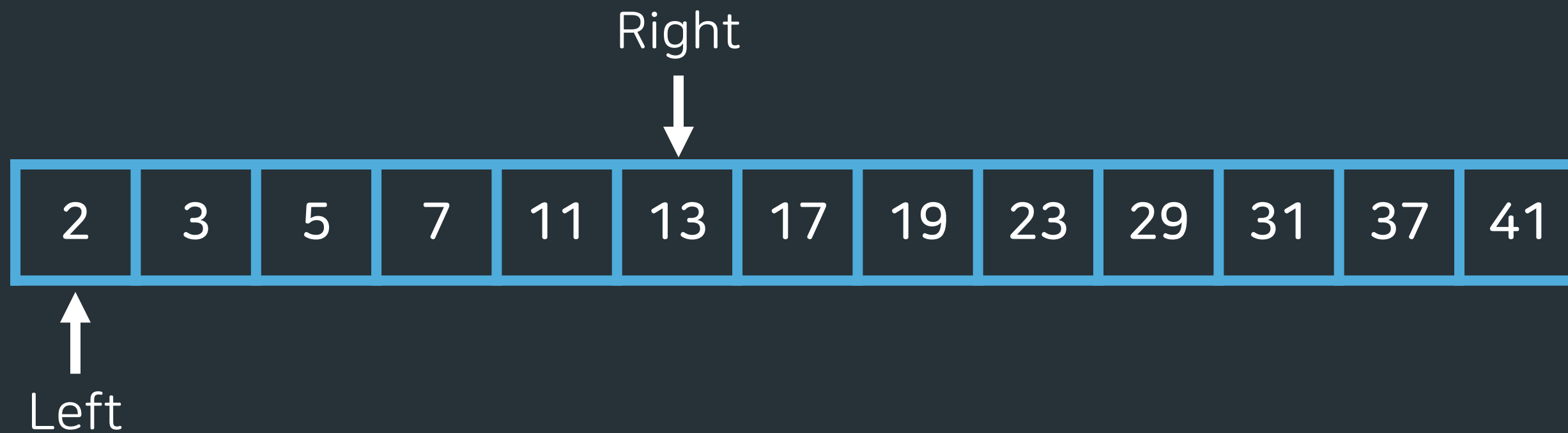


Left ~ Right = 41

이제 어디로 가지?

## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41

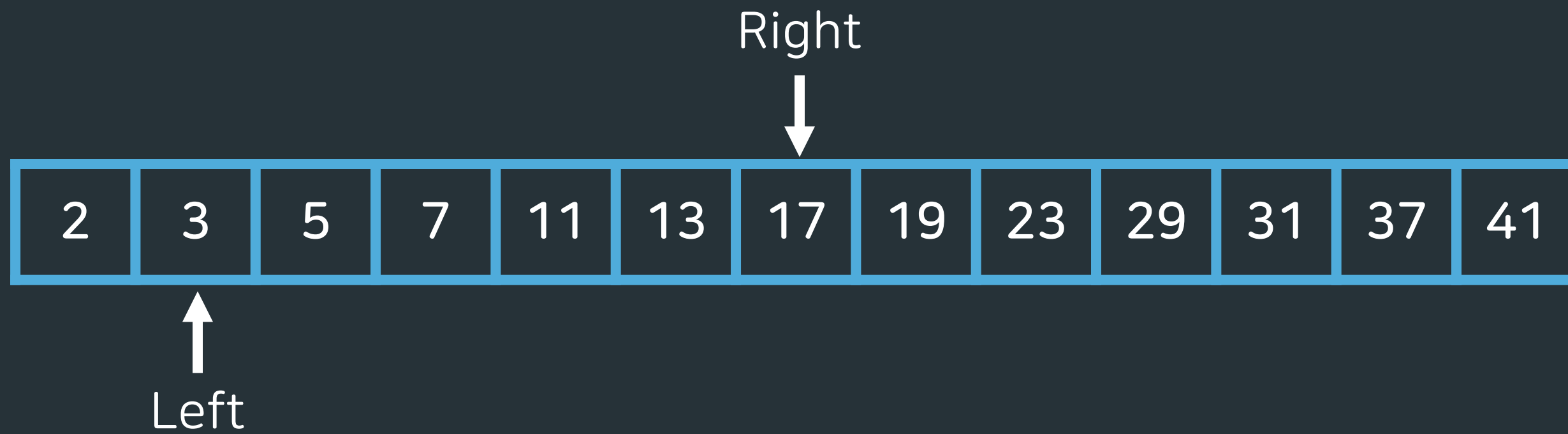


Left ~ Right = 41

이제 어디로 가지?  
둘 다 한 칸씩 옮기자!

## 같은 위치에서 시작하는 두 포인터

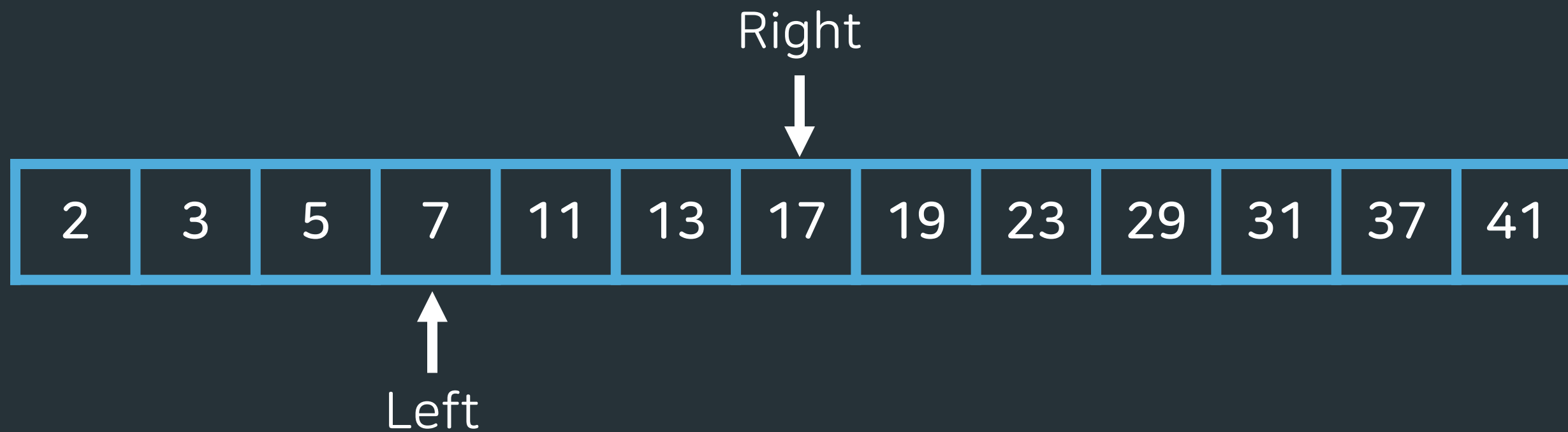
- (예제) 연속된 소수의 합이 41



Left ~ Right = 56

## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41

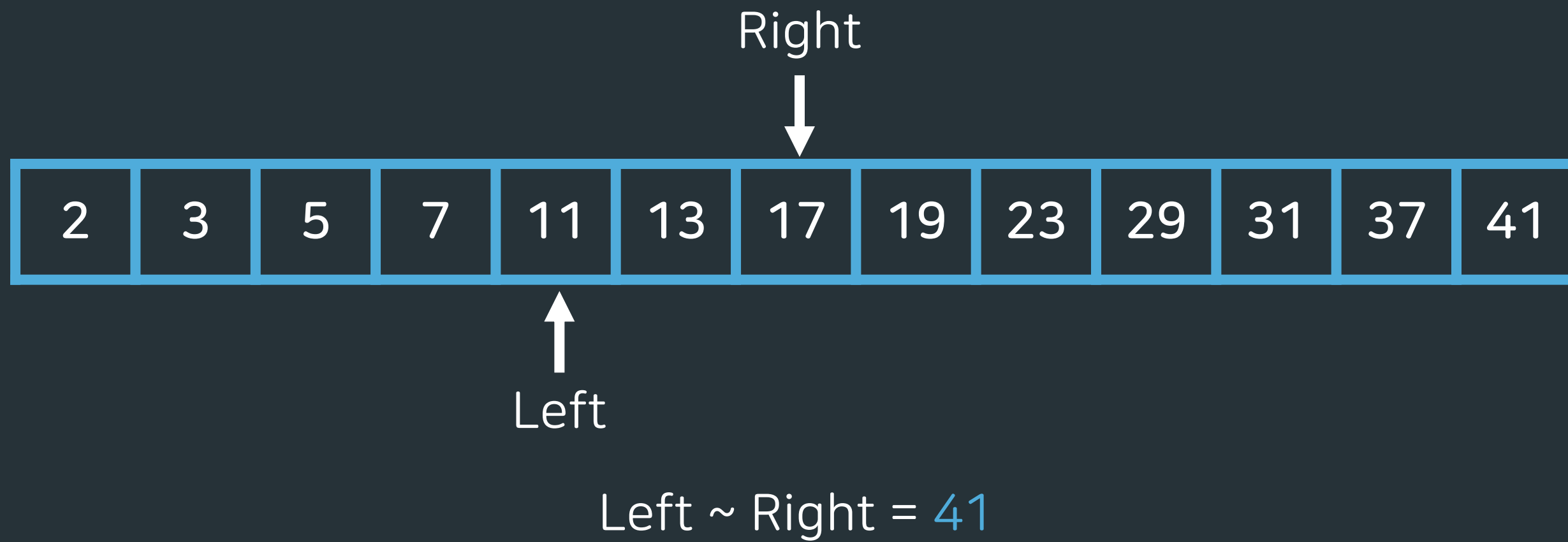


Left ~ Right = 48



## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41



## 같은 위치에서 시작하는 두 포인터

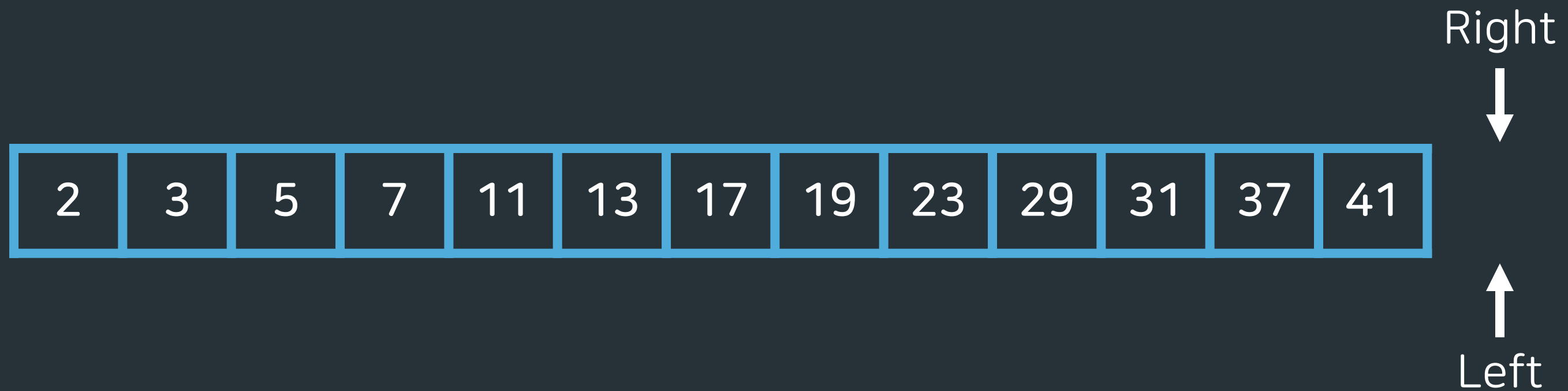
- (예제) 연속된 소수의 합이 41



Left ~ Right = 41

## 같은 위치에서 시작하는 두 포인터

- (예제) 연속된 소수의 합이 41



Left ~ Right =

break!

## /<> 10025번 : 게으른 백곰 - Silver 4

### 문제

- 얼음 양동자들이  $x$ 좌표마다 놓여 있고, 각 양동이 안에는  $g$ 씩의 얼음이 들어 있다
- 백곰 앨버트가 자리를 잡으면 그로부터 좌우로  $k$ 만큼 떨어진 양동일까지 닿을 수 있다
- 앨버트가 최적의 자리를 골랐을 때의 얼음의 합을 구하여라
- 즉, 얼음들의 합의 최댓값을 구하여라

### 제한 사항

- 얼음 양동이 개수  $N$ 은  $1 \leq N \leq 100,000$
- 얼음 양동이 좌표  $X$ 는  $0 \leq X \leq 1,000,000$
- 얼음 무게  $g$ 는  $1 \leq g \leq 10,000$
- 입력 범위  $K$ 는  $1 \leq K \leq 2,000,000$

## 예제 입력

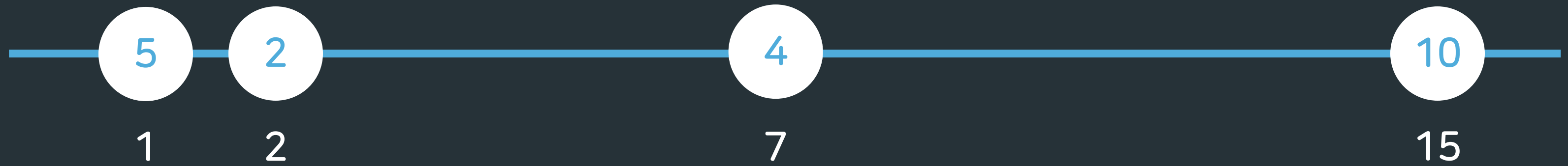
```
4 3
4 7
10 15
2 2
5 1
```

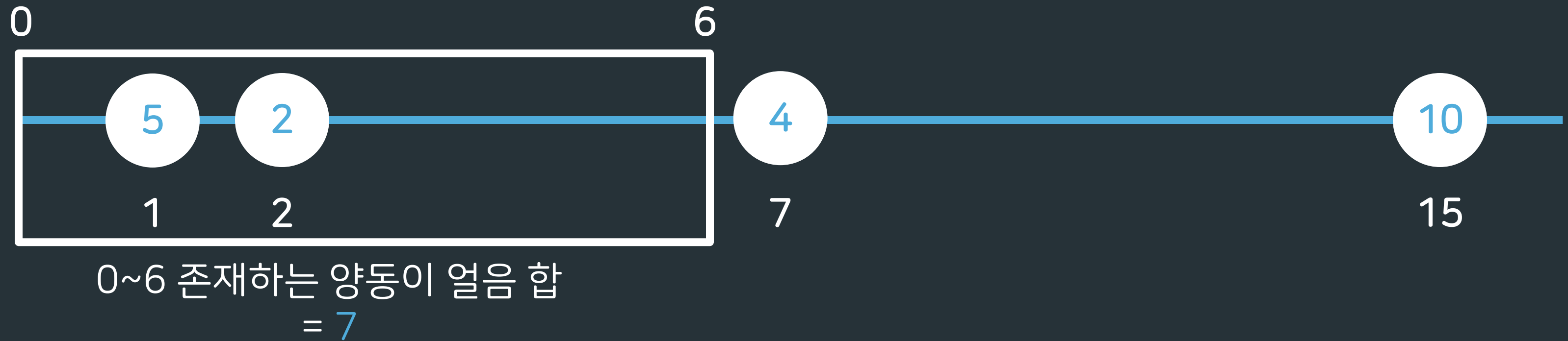
## 예제 출력

```
11
```

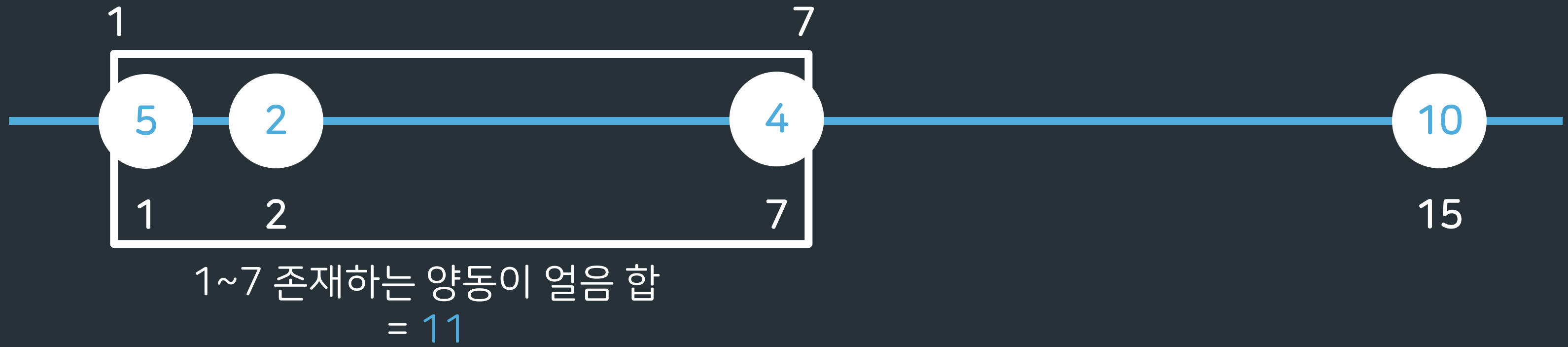
## Hint

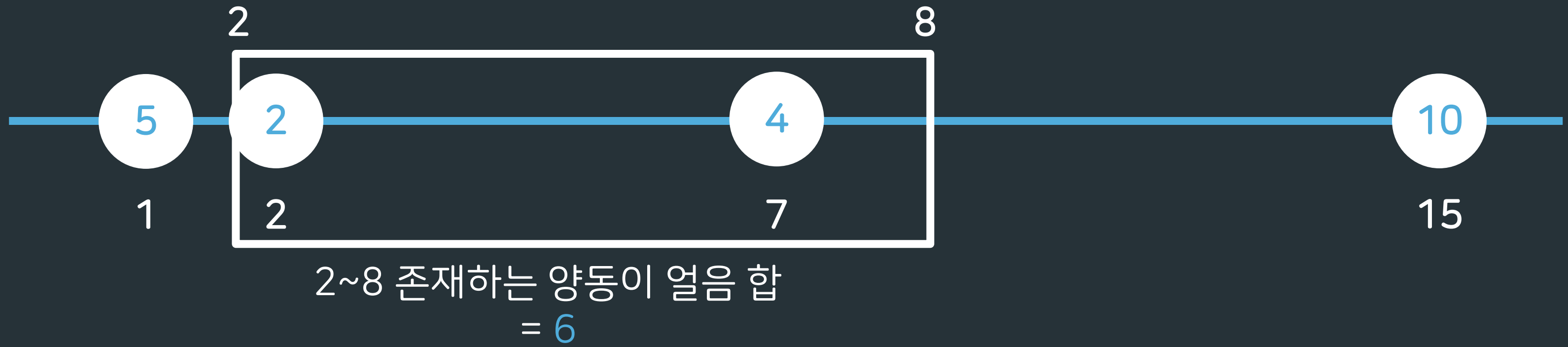
1. 오늘 다룬 누적합, 슬라이딩 윈도우, 투 포인터 중 무엇일까요?
2. K라는 구간은 변하지 않고, 정해져 있네요!
3. 그리고 구간 내의 모든 양동이를 고려해야 해요

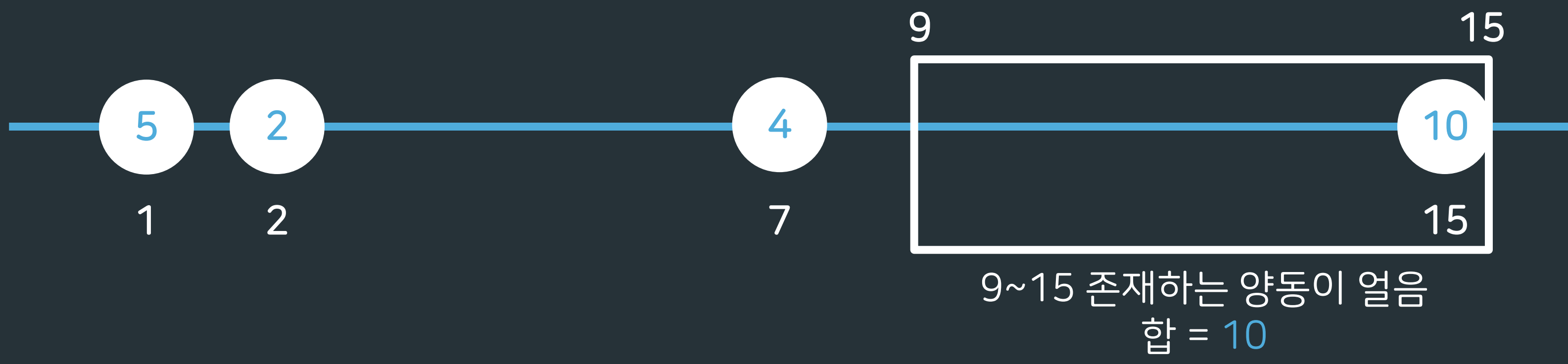













## 정리

- 다양한 경우에서 배열의 탐색 효율을 높이기 위해 사용되는 투 포인터 알고리즘!
- 두개의 포인터 사이의 거리가 고정된다면 슬라이딩 윈도우!
- 포인터가 가까워지는 방법( $left < right$ )과 멀어지는 방법( $left \leq right$ )이 있음
- 가까워지는 방법은 보통 중복이 없고, 정렬된 배열에만 사용 가능함. 두 개의 포인터가 가리키는 값만 고려
- 멀어지는 방법은 두 개의 포인터가 가리키는 값 사이의 모든 값을 고려
- 효율성 테스트 문제로 아주 많이 출제됨

## 3문제 이상 선택

- /<> 2473번 : 세 용액 - Gold 4
-  2020 카카오 인턴십 : 보석 쇼핑 - Level 3
- /<> 1253번 : 좋다 - Gold 4
- /<> 1484번 : 다이어트 - Gold 4
- /<> 2531번 : 회전 초밥 - Silver 1

**코드리뷰 O 마감** ~ 4월 30일 토요일 낮 12시

**코드리뷰 X 마감** ~ 4월 30일 토요일 밤 12시 (30일에서 5월 1일로 넘어가는 자정)

**추가제출 마감** ~ 5월 1일 일요일 밤 12시 (5월 1일에서 5월 2일로 넘어가는 자정)