

Chapter. 02

알고리즘

# | 두 포인터 Two Pointers

FAST CAMPUS  
ONLINE

알고리즘 공채 대비반 I

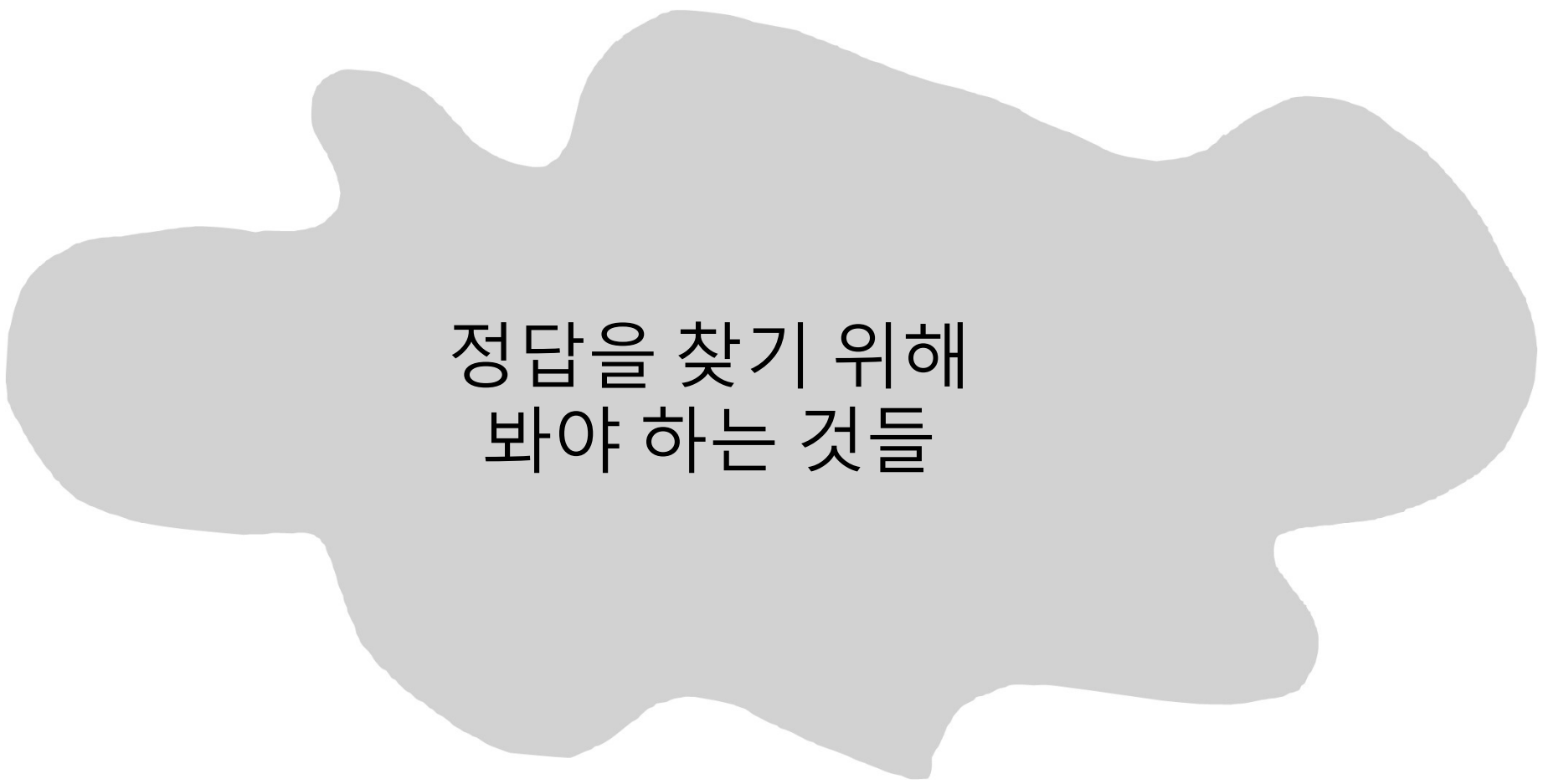
강사. 류호석

Chapter. 02

알고리즘  
두 포인터(Two Pointers)

Chapter. 02 알고리즘

# I 알고리즘의 핵심



정답을 찾기 위해  
보야 하는 것들

FAST CAMPUS  
ONLINE

류호석 강사.

Chapter. 02 알고리즘

# I 알고리즘의 핵심



볼 필요 없는 부분들

FAST CAMPUS  
ONLINE

류호석 강사.

Chapter. 02 알고리즘

# I 알고리즘의 핵심



**꼭 봐야 하는 애들**

FAST CAMPUS  
ONLINE

류호석 강사.

# I 접근 - 정답을 위해 봐야 하는 것들

L \ R	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

정답을 찾기 위해  
 봐야 하는 것들  
 모든  $(L, R)$

# I 접근 - 정답을 위해 봐야 하는 것들

L \ R	1	2	3	4	5	6	7	8	9	10
1		꼭	봐	야	하					
2		볼	필	요	는					
3			없	음	위					
4				!	치					
5					들					
6										
7										
8										
9										
10										

## I 두 포인터(Two Pointers)

### 화살표 두 개에 의미를 부여해서 탐색 범위를 압축하는 방법!

1. 1차원 배열 위에 2개의 포인터를 만드는 경우
  1. 2개의 포인터가 모두 왼쪽에서 시작해서 **같은 방향**으로 이동
  2. 2개의 포인터가 양 끝에서 **서로를 향해** 이동

1	2	3	4	5	6	7	8	9
10	11	18	19	38	58	72	87	92



## I 두 포인터(Two Pointers)

### 화살표 두 개에 의미를 부여해서 탐색 범위를 압축하는 방법!

- 1차원 배열 위에 2개의 포인터를 만드는 경우
  - 2개의 포인터가 모두 왼쪽에서 시작해서 **같은 방향**으로 이동
  - 2개의 포인터가 양 끝에서 **서로를 향해** 이동
- 관찰을 통해서 문제에 등장하는 **변수 2개의 값**을 두 포인터로 표현하는 경우

# I 두 포인터(Two Pointers) - 꿀팁

## <키워드>

- 1차원 배열에서의 “연속 부분 수열” or “순서를 지키며 차례대로”
- 곱의 최소

=> 이런 단어가 등장하면 Two Pointers 접근을 시도해 볼 가치가 있다!

I BOJ 1806-부분합

난이도: 3

 $10 \leq N < 100,000$  $0 < S \leq 10^8$  $1 \leq \text{각 원소} \leq 10,000$ 

원소의 개수,  $N = 10$   
 기준 합,  $S = 15$

1	2	3	4	5	6	7	8	9	10
5	1	3	5	10	7	4	9	2	8

이 수열에서 연속된 수들의 부분합 중에 그 합이  $S$  이상이 되는 것 중, 가장 짧은 것의 길이를 구하는 프로그램

## I 문제 파악하기 - 정답의 최대치

1	2	3	...	99,999	100,000
1	1	1	...	1	1

$$N = 100,000$$

$$S = 10^8$$

정답이  $N$  이하이므로 Integer 범위

모든 원소의 총합도  $10^9$ 이므로 Integer 범위

I 접근 – 가장 쉬운 방법  $O(N^2)$ 기준 합,  $S = 15$ 

1	2	3	4	5	6	7	8	9	10
5	1	3	5	10	7	4	9	2	8

1. 왼쪽 시작  $L$  결정  $\Rightarrow O(N)$
2. 오른쪽 끝을  $R$ 을  $L$ 부터 시작해서 이동  $\Rightarrow O(N)$
3.  $O(N^2)$

# I 접근 - 정답을 위해 봐야 하는 것들

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

# I 접근 - 정답을 위해 봐야 하는 것들

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2										
3										
4										
5										
6										
7										
8										
9										
10										

# I 접근 - 정답을 위해 봐야 하는 것들

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2		1	4	9	19					
3			3	8	18					
4				5	15					
5					10	17				
6						7	11	20		
7							4	13	15	
8								9	11	19
9									2	10
10										8



I 접근 - 볼 필요가 없는 부분!!!

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2		1	4	9	19					
3			3	8	18					
4				5	15					
5					10	17				
6						7	11	20		
7							4	13	15	
8								9	11	19
9									2	10
10										8

# I 접근 - 정답을 위해 봐야 하는 것들

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2										
3										
4										
5										
6										
7										
8										
9										
10										

## I 접근 - 정답을 위해 봐야 하는 것들

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2					19					
3					18					
4					15					
5										
6										
7										
8										
9										
10										

# I 접근 - 정답을 위해 봐야 하는 것들

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2					19					
3					18					
4					15					
5					10	17				
6										
7										
8										
9										
10										

# I 접근 - 볼 필요가 없는 부분!!!

	5	1	3	5	10	7	4	9	2	8
L \ R	1	2	3	4	5	6	7	8	9	10
1	5	6	9	14	24					
2					19					
3					18					
4					15					
5					10	17				
6						7	11	20		
7								13	15	
8									11	19
9										10
10										8

I 접근 – 투 포인터 방법  $O(N)$ 

↖ := L, 구간의 왼쪽 끝

↘ := R, 구간의 오른쪽 끝

$Sum$  := [L ... R] 구간의 합

기준 합,  $S = 15$

1	2	3	4	5	6	7	8	9	10
5	1	3	5	10	7	4	9	2	8

I 접근 – 투 포인터 방법  $O(N)$ 

↖ := L, 구간의 왼쪽 끝

↘ := R, 구간의 오른쪽 끝

Sum := [L ... R] 구간의 합

1	2	3	4	5	6	7	8	9	10
5	1	3	5	10	7	4	9	2	8

1. 왼쪽 시작 L의 이동 횟수  $N$  번
2. 오른쪽 끝을 R을 이전의 R부터 시작해서 이동
3. L, R이 각자 최대  $N$  번 이동하니까,  $O(N)$

## 구현

```
static void pro() {  
    int R = 0, sum = 0, ans = n + 1;  
    for (int L = 1; L <= n; L++) {  
        // L - 1 을 구간에서 제외하기  
  
        // R 을 옮길 수 있을 때 까지 옮기기  
  
        // [L ... R] 의 합, 즉 sum이 조건을 만족하면 정답 갱신하기  
    }  
  
    // ans 값을 보고 불가능 판단하기  
    System.out.println(ans);  
}
```



# I 연습 문제

- BOJ 2003 – 수들의 합 2
- BOJ 2559 – 수열
- BOJ 15565 – 귀여운 라이언
- BOJ 11728 – 배열 합치기
- BOJ 2230 – 수 고르기

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드

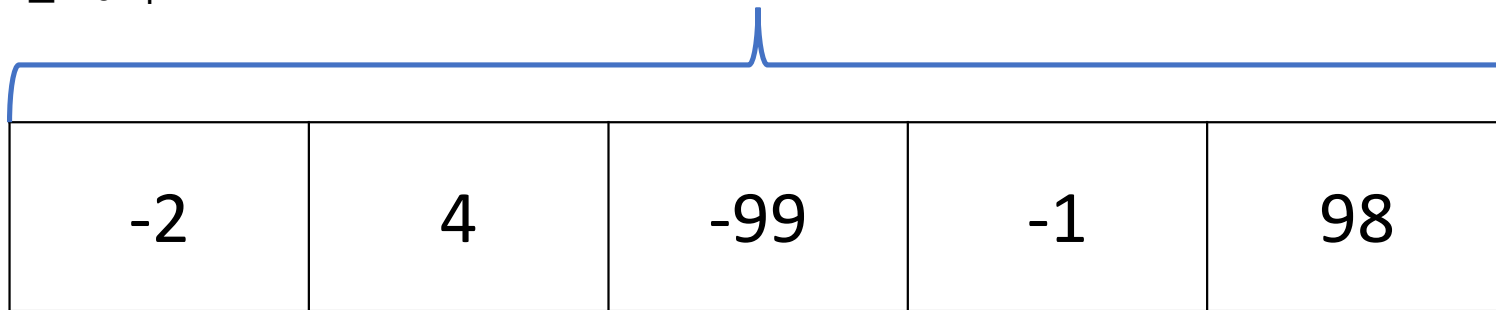
## I BOJ 2470 – 두 용액

난이도: 3

$2 \leq N \leq 100,000$

$-10억 \leq \text{원소} \leq 10억$

$N = 5$



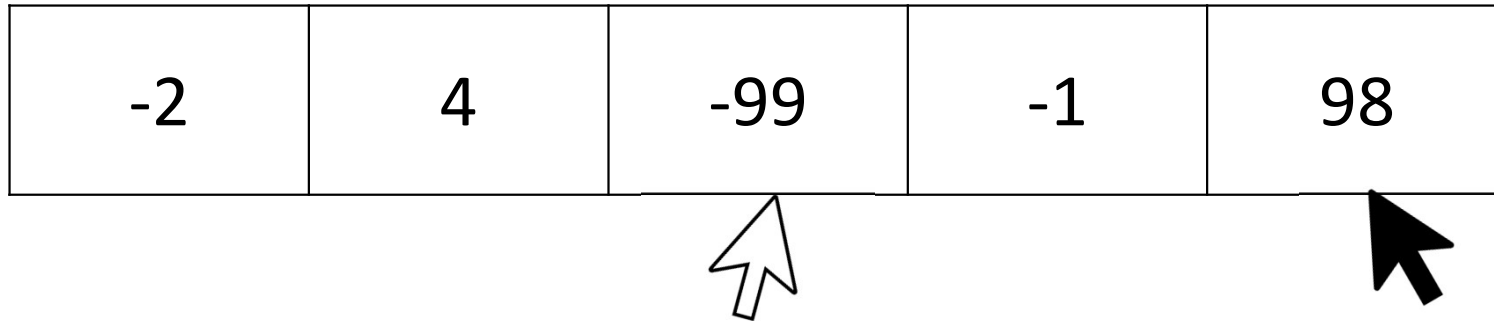
-2	4	-99	-1	98
----	---	-----	----	----

서로 다른 두 용액을 더해서 합이 최대한 0에 가깝게 만들기

I 접근 – 기존의 이분 검색 방법  $O(N \log N)$ 

-2	4	-99	-1	98
----	---	-----	----	----

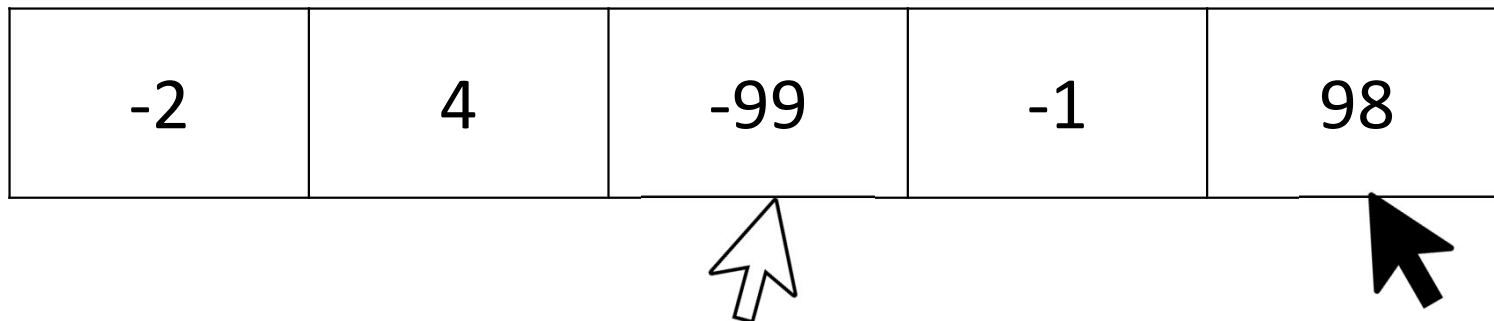
1. 정렬  $O(N \log N)$
2. 이분 검색  $O(N \log N)$

I 접근 – 빠른 방법  $O(N \log N)$ 

$L :=$  “남아 있는 것들 중” 제일 작은 원소

$R :=$  “남아 있는 것들 중” 제일 큰 원소

## I 접근 - 색다른 접근



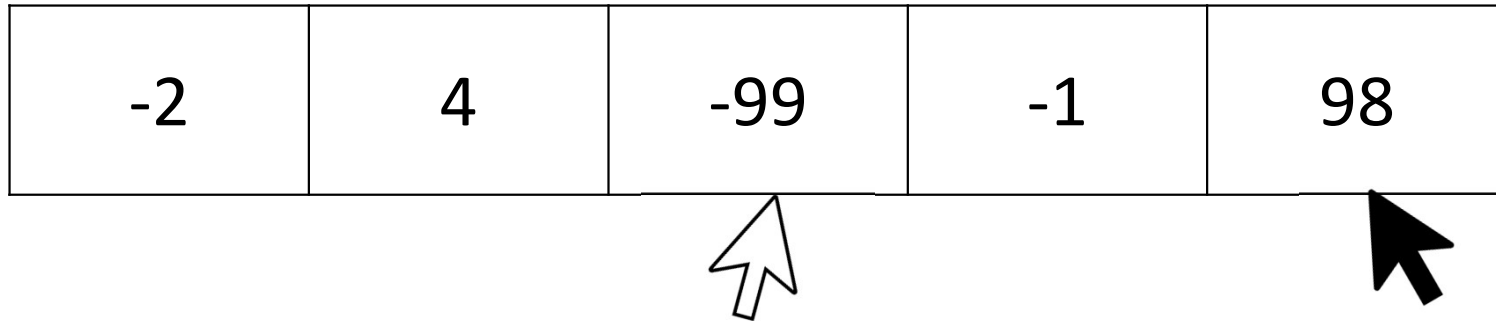
1. 최소 + 최대 < 0

→ ?

2. 최소 + 최대 > 0

→ ?

## I 접근 - 색다른 접근



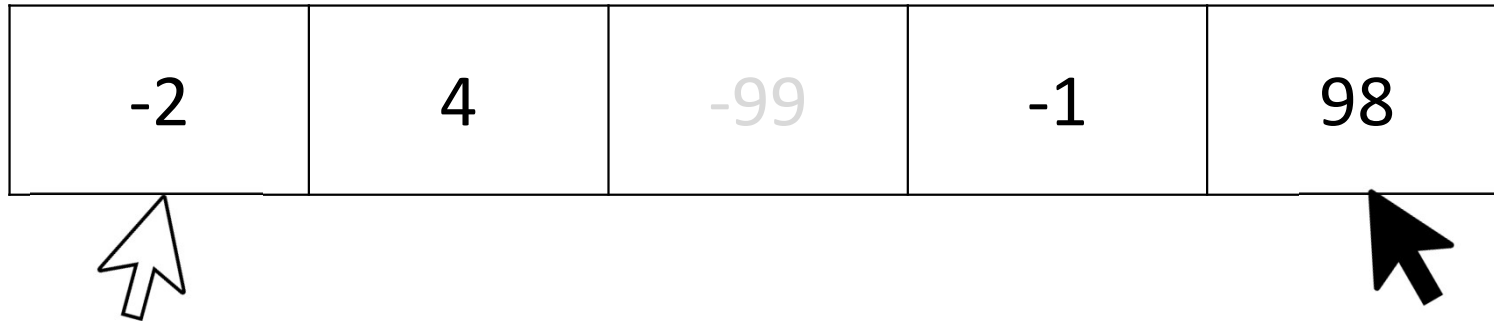
1. 최소 + 최대 < 0

→ 최소 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

2. 최소 + 최대 > 0

→ ?

## I 접근 - 색다른 접근



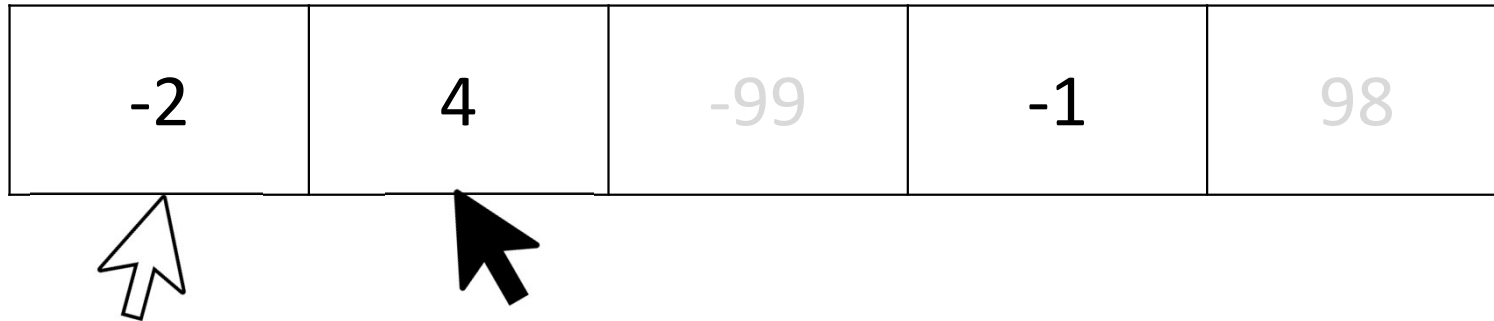
1. 최소 + 최대 < 0

➔ 최소 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

2. 최소 + 최대 > 0

➔ 최대 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

## I 접근 - 색다른 접근



1. 최소 + 최대 < 0

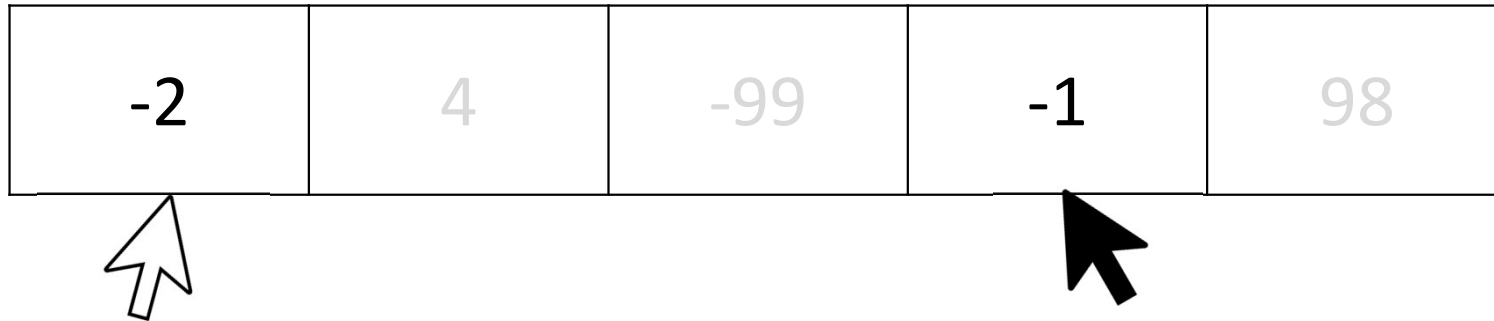
➔ 최소 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

2. 최소 + 최대 > 0

➔ 최대 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)



## I 접근 - 색다른 접근



1. 최소 + 최대 < 0

➔ 최소 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

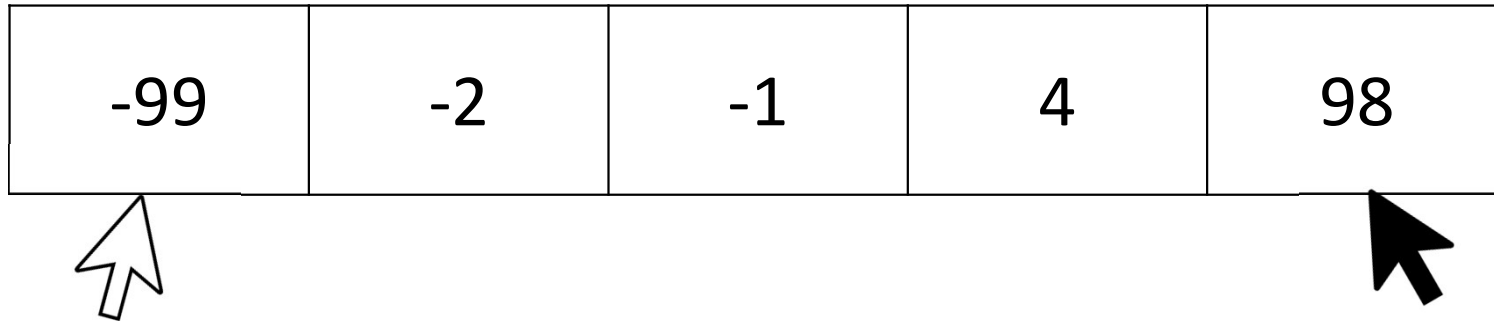
2. 최소 + 최대 > 0

➔ 최대 입장에서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

# I 시간, 공간 복잡도 계산하기

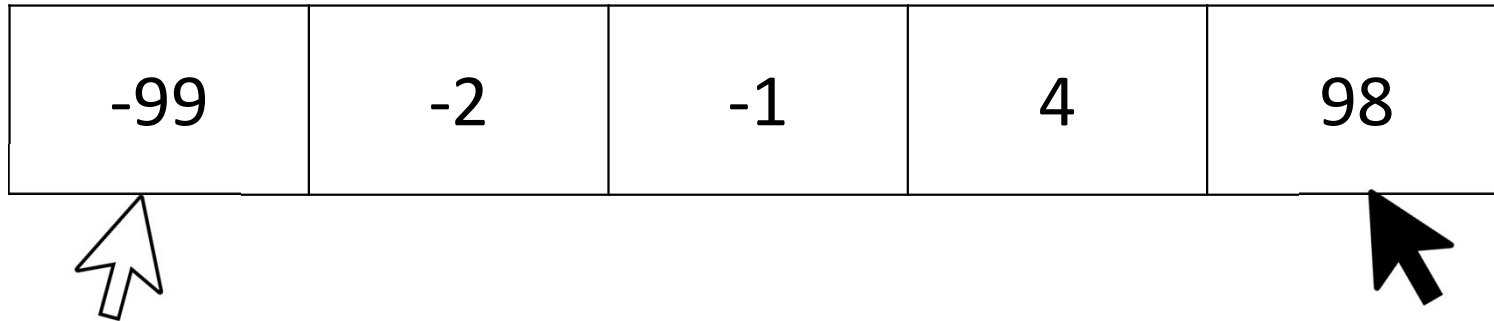
1. 매 순간 L, R 을 찾아야 한다  $\rightarrow O(N)$
2. 원소가 1개가 될 때까지 반복  $\rightarrow N$ 번 반복
3. 총  $O(N^2)$  의 시간이 걸린다.

# I 접근 – 빠른 방법 $O(N \log N)$



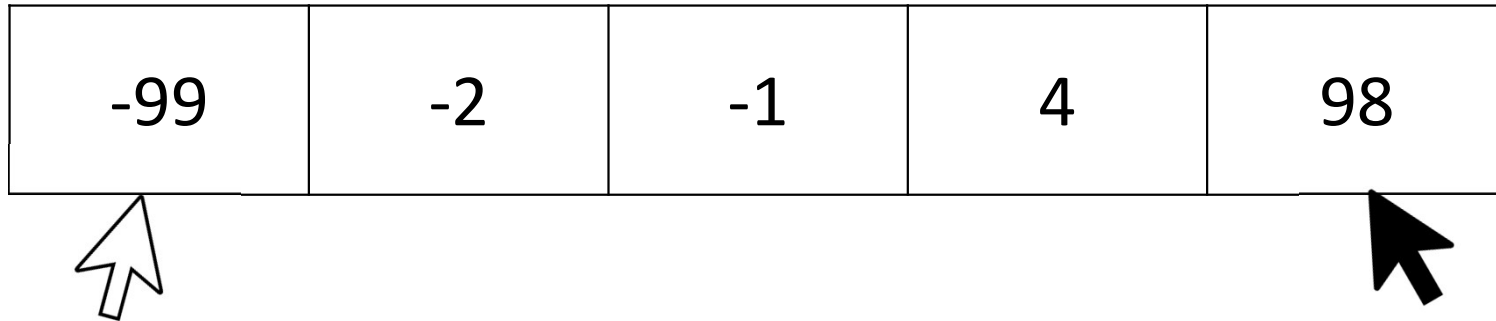
**정렬 해보기!  $O(N \log N)$**

제일 작은 원소, 제일 큰 원소를 **빠르게** 알 수 있다!

I 접근 – 빠른 방법  $O(N \log N)$ 

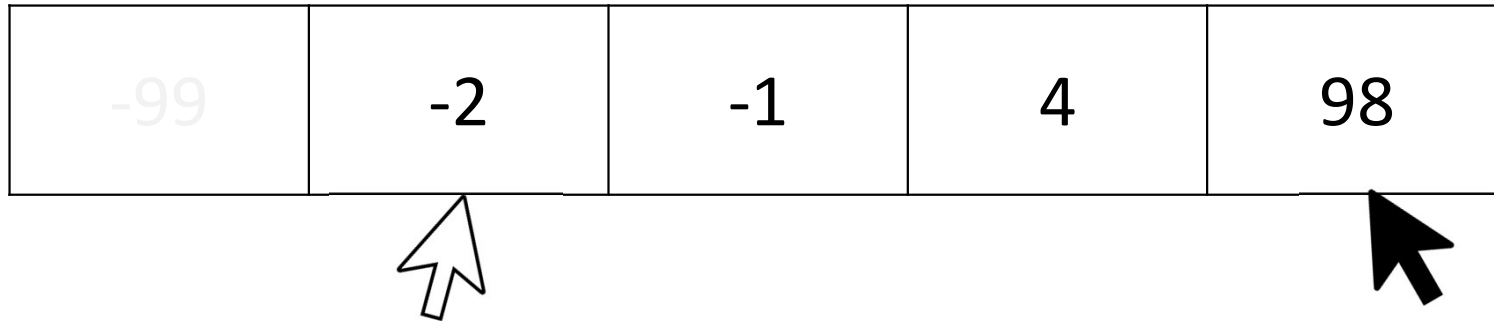
$L :=$  “남아 있는 것들 중” 제일 작은 원소

$R :=$  “남아 있는 것들 중” 제일 큰 원소

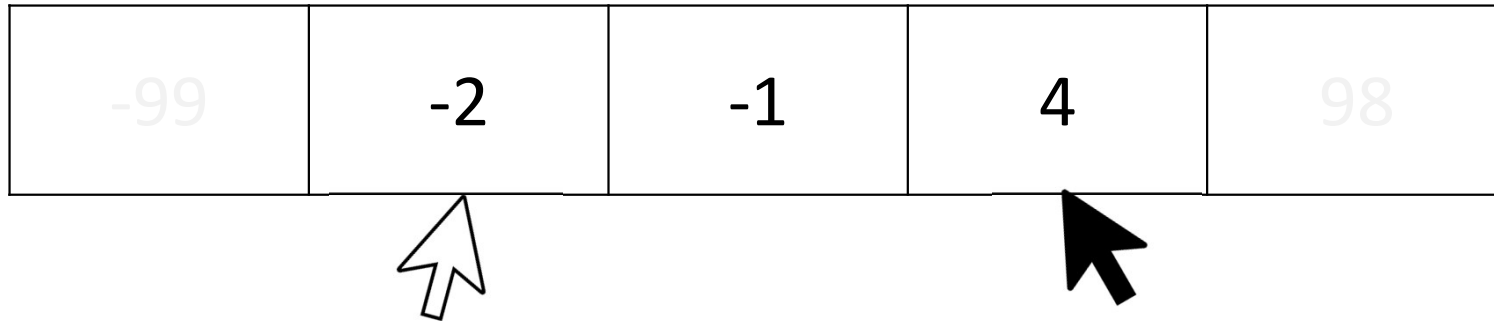
I 접근 - 빠른 방법  $O(N \log N)$ 

1. 최소 + 최대  $< 0 \rightarrow$  최소 입장에서서는 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

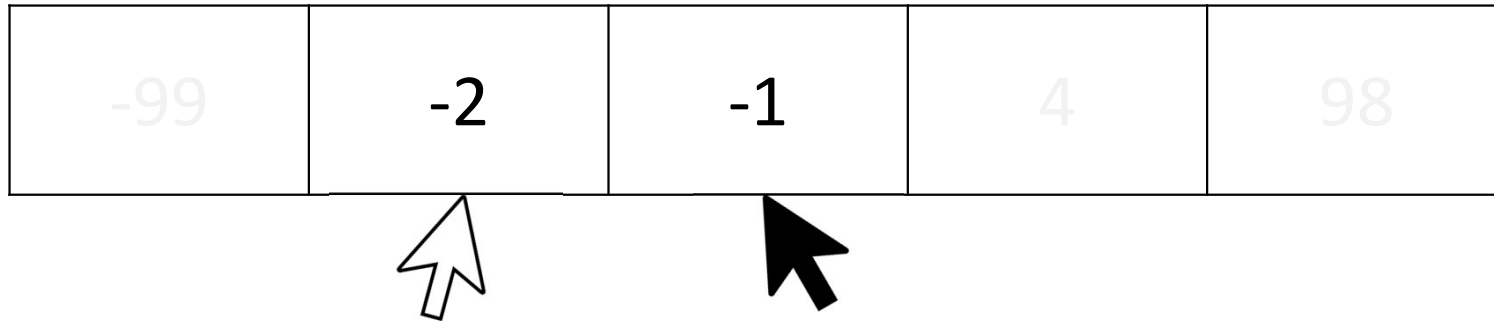
2. 최소 + 최대  $> 0 \rightarrow$

I 접근 - 빠른 방법  $O(N \log N)$ 

1. 최소 + 최대  $< 0 \rightarrow$  최소 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)
2. 최소 + 최대  $> 0 \rightarrow$  최대 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

I 접근 – 빠른 방법  $O(N \log N)$ 

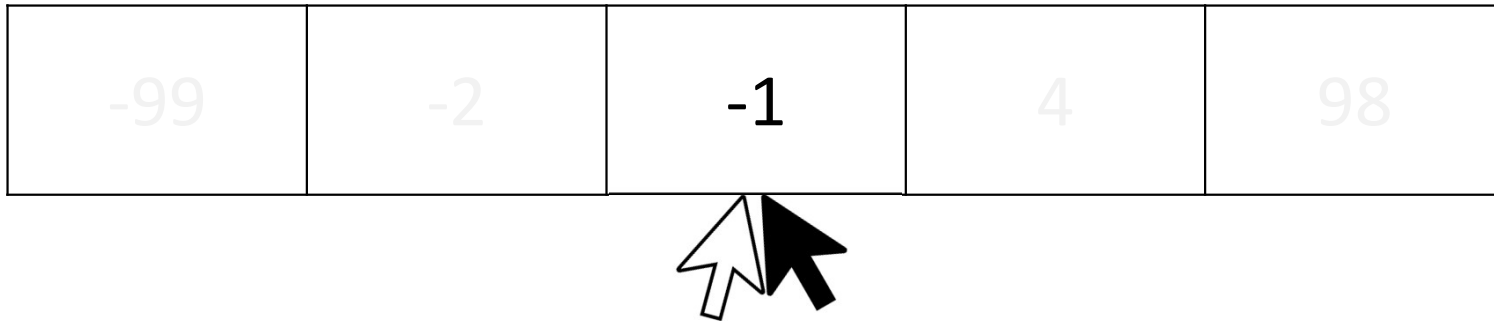
1. 최소 + 최대  $< 0 \rightarrow$  최소 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)
2. 최소 + 최대  $> 0 \rightarrow$  최대 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)

I 접근 – 빠른 방법  $O(N \log N)$ 

1. 최소 + 최대  $< 0 \rightarrow$  최소 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)
2. 최소 + 최대  $> 0 \rightarrow$  최대 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)



## I 접근 - 빠른 방법 $O(N \log N)$



1. 최소 + 최대  $< 0 \rightarrow$  최소 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)
2. 최소 + 최대  $> 0 \rightarrow$  최대 입장에서선 최선책을 만난 상태! 짝을 찾았으니 삭제(더 고려 x)
3.  $L = R \rightarrow$  서로 다른 두 용액을 고를 수 없는 상태이므로 종료!

## I 접근 - 정답을 위해 봐야 하는 것들

L \ R	1	2	3	4	5
1					
2					
3					
4					
5					

# I 시간, 공간 복잡도 계산하기

1. 배열 정렬 한 번  $\Rightarrow O(N \log N)$
2. 매 순간 L, R 로 계산한 후에, 이동시키기  $\Rightarrow O(N)$
3. 총 시간 복잡도:  $O(N \log N)$

## 구현

```
static void pro() {  
    // A 에 대해 이분 탐색을 할 예정이니까, 정렬을 미리 해주자.  
    Arrays.sort(A, fromIndex: 1, toIndex: N + 1);  
  
    int best_sum = Integer.MAX_VALUE;  
    int v1 = 0, v2 = 0, L = 1, R = N;  
  
    while (L < R){ // L == R 인 상황이면 용액이 한 개 뿐인 것이므로, L < R 일 때까지만 반복한다.  
        /* TODO */  
    }  
    sb.append(v1).append(' ').append(v2);  
    System.out.println(sb);  
}
```

# I 연습 문제

- BOJ 3273 – 두 수의 합

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드