

Chapter. 02

알고리즘

# 이분 탐색 Binary Search

FAST CAMPUS  
ONLINE

알고리즘 공채 대비반 I

강사. 류호석

Chapter. 02

# 알고리즘 이분 탐색 (Binary Search)



## I 탐색(Search)

## (수열에서의) 탐색

수열과 탐색 대상  $x$ 가 주어졌을 때,

$x = 63$	72	19	38	58	10	92	18	11	87
----------	----	----	----	----	----	----	----	----	----

- $x$ 가 존재하는 지?
- $x$ [이하, 미만, 이상, 초과]의 원소는 몇 개가 있는 지?
- $x$ 랑 가장 가까운 원소는 무엇인지?

모두  $O(N)$

## I 탐색(Search)

(수열에서의) 탐색 => 만약! 정렬이 되어 있다면...?

정렬된 수열과 탐색 대상  $x$ 가 주어졌을 때,

$x = 63$	10	11	18	19	38	58	72	87	92
----------	----	----	----	----	----	----	----	----	----

- $x$ 가 존재하는 지?
- $x$ [이하, 미만, 이상, 초과]의 원소는 몇 개가 있는 지?
- $x$ 랑 가장 가까운 원소는 무엇인지?

더 빠르게...?

# I 이분 탐색(Binary Search)

## 이분 탐색(Binary Search)

**무엇: 정렬이 보장되는 배열**에서 기준  $x$  를 가지고 범위를 “이분” 하면서 탐색하는 방법

$x = 63$

10	11	18	19	38	58	72	87	92
----	----	----	----	----	----	----	----	----

$x = 63$

?	?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---	---



- $x$  가 존재하는 지?
- $x$  [이하, 미만, 이상, 초과]의 원소는 몇 개가 있는 지?
- $x$  랑 가장 가까운 원소는 무엇인지?

⇒ 시간 복잡도:  $O(\log N)$

# I 이분 탐색(Binary Search)

※ 오름차순 정렬이 된 배열의 특성

	1	2	3	4	5	6	7	8	9
$x = 63$	10	11	18	19	38	58	72	87	92

1. 임의의 M번 인덱스에 있는  $A[M]$  이  $x$  보다 **크다면**,  $A[M...N]$  은 전부  $x$  보다 **크다**.
2. 임의의 M번 인덱스에 있는  $A[M]$  이  $x$  보다 **작다면**,  $A[1...M]$  은 전부  $x$  보다 **작다**.

※주의※ 오름차순 정렬이기 때문에 생기는 성질! 정렬이 아니라면 불가능하다.

# I 이분 탐색(Binary Search)

**L** := 탐색할 가치가 있는 범위의 **왼쪽** 끝 인덱스

**R** := 탐색할 가치가 있는 범위의 **오른쪽** 끝 인덱스

**Result** := 탐색한  $x$ 의 위치

탐색 목표 :=  $x$  이하의 원소 중에 가장 오른쪽에 있는 원소

**X = 63**

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

**Result**

$$M = (L+R)/2 = 5$$

**$A[M]$  과  $x$  비교!**

# I 이분 탐색(Binary Search)

**L** :=  $x$ 를 탐색할 가치가 있는 범위의 **왼쪽** 끝 인덱스

**R** :=  $x$ 를 탐색할 가치가 있는 범위의 **오른쪽** 끝 인덱스

**Result** := 탐색한  $x$ 의 위치

탐색 목표 :=  $x$  이하의 원소 중에 가장 오른쪽에 있는 원소

**Result**

	1	2	3	4	5	6	7	8	9
<b>X = 63</b>	10	11	18	19	38	58	72	87	92

[
]



$$M = (L+R)/2 = 7$$

**$A[M]$  과  $x$  비교!**



# I 이분 탐색(Binary Search)

**L** :=  $x$ 를 탐색할 가치가 있는 범위의 **왼쪽** 끝 인덱스

**R** :=  $x$ 를 탐색할 가치가 있는 범위의 **오른쪽** 끝 인덱스

**Result** := 탐색한  $x$ 의 위치

탐색 목표 :=  $x$  이하의 원소 중에 가장 오른쪽에 있는 원소

**Result**

	1	2	3	4	5	6	7	8	9
<b>X = 63</b>	10	11	18	19	38	58	72	87	92

$$M = (L+R)/2 = 6$$

**$A[M]$  과  $x$  비교!**

# I 이분 탐색(Binary Search)

**L** :=  $x$ 를 탐색할 가치가 있는 범위의 **왼쪽** 끝 인덱스

**R** :=  $x$ 를 탐색할 가치가 있는 범위의 **오른쪽** 끝 인덱스

**Result** := 탐색한  $x$ 의 위치

탐색 목표 :=  $x$  이하의 원소 중에 가장 오른쪽에 있는 원소

	1	2	3	4	5	6	7	8	9
$x = 63$	10	11	18	19	38	58	72	87	92

Result  
] [

$L > R$  : 탐색할 가치가 있는 구간이 없다!

**Result=6** 이므로

- $A[6]$  은  $x$  이하 중 제일 큰 값이고
- $A[7]$ 은  $x$ 보다 큰 값일 것이다!
- 또한  $x$  이하의 숫자가 6개 인 것도 알 수 있다!

# I 이분 탐색(Binary Search) – 비교 과정 정리

X = 63

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

순서	L	R	M	A[M]과 X 비교	Result
초기값	1	9			0
			5	A[5]=38 <= X	
1	6	9			5
			7	A[7]=72 > X	
2	6	6			5
			6	A[6]=58 <= X	
3	7	6			6

# I 이분 탐색(Binary Search) – 시간 복잡도

	1	2	3	4	5	6	7	8	9
<b>X = 63</b>	10	11	18	19	38	58	72	87	92

A[M]과 X를 한 번 비교할 때마다 [L, R] 구간이 절반씩 좁아집니다.

구간의 길이:  $N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \rightarrow \dots \rightarrow 1$  의 순서로 구간이 점점 좁아집니다.

즉, “총 비교 횟수”는 “구간의 변화 횟수”인  $O(\log N)$  만에 원하는 값을 탐색합니다.

# I 이분 탐색(Binary Search) – 시간 복잡도

	1	2	3		...			10만
<b>X = 63</b>	10	11	18					92

만약 N=10만 이면?

$$10\text{만} \gg \gg \gg \log(10\text{만}) \cong 16$$

# I 이분 탐색(Binary Search) – 자주 하는 실수

**1위** 입력된 배열에 바로 이분 탐색을 하는데, 정렬을 하지 않는 경우!

**2위** L, R, M, Result 변수의 정의를 헷갈려서 부등호 등을 잘못 쓰는 경우!

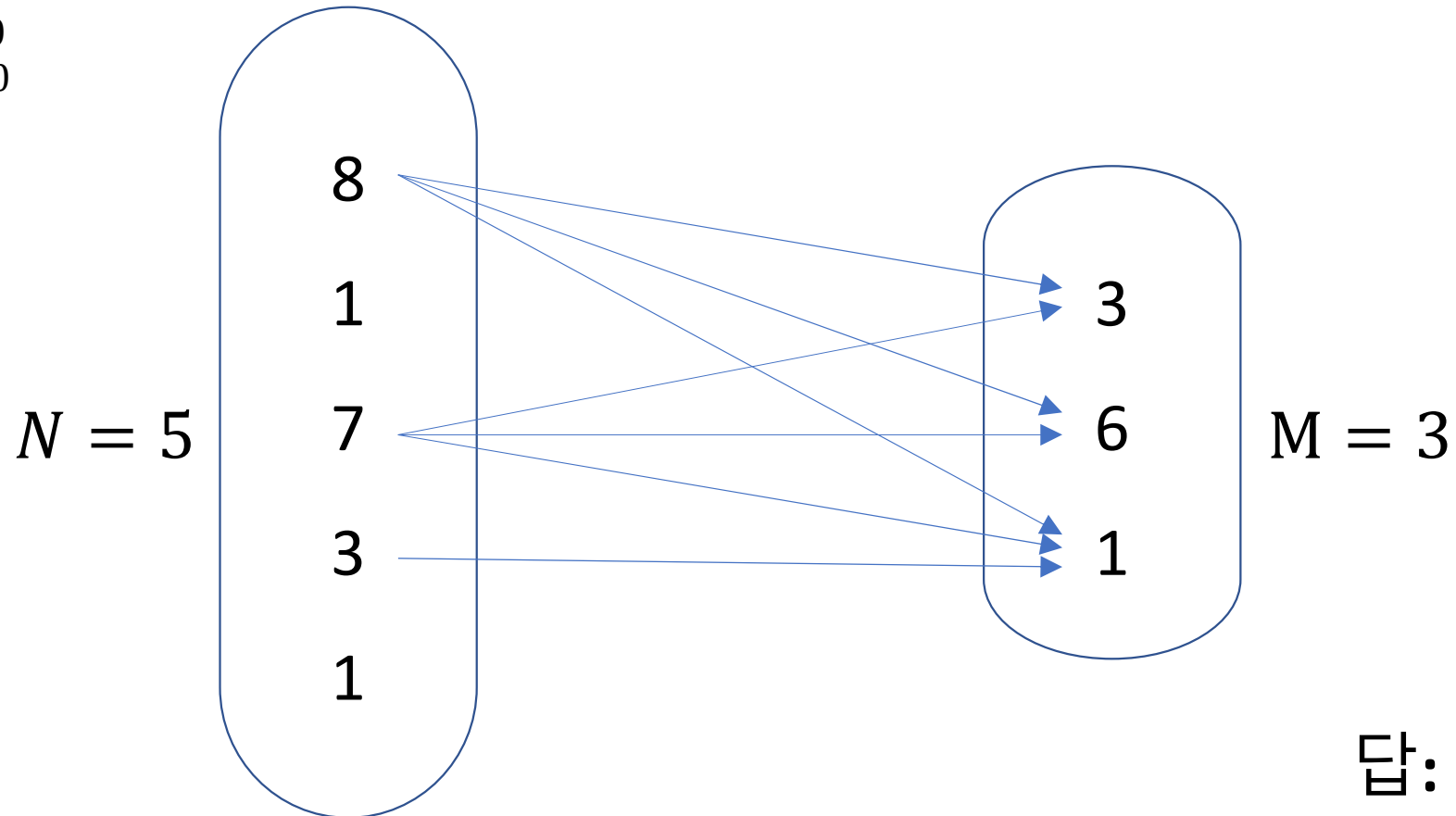
**3위** L, R 범위를 잘못 설정하거나 Result의 초기값을 잘못 설정하는 경우!

# I BOJ 7795-먹을 것인가 먹힐 것인가

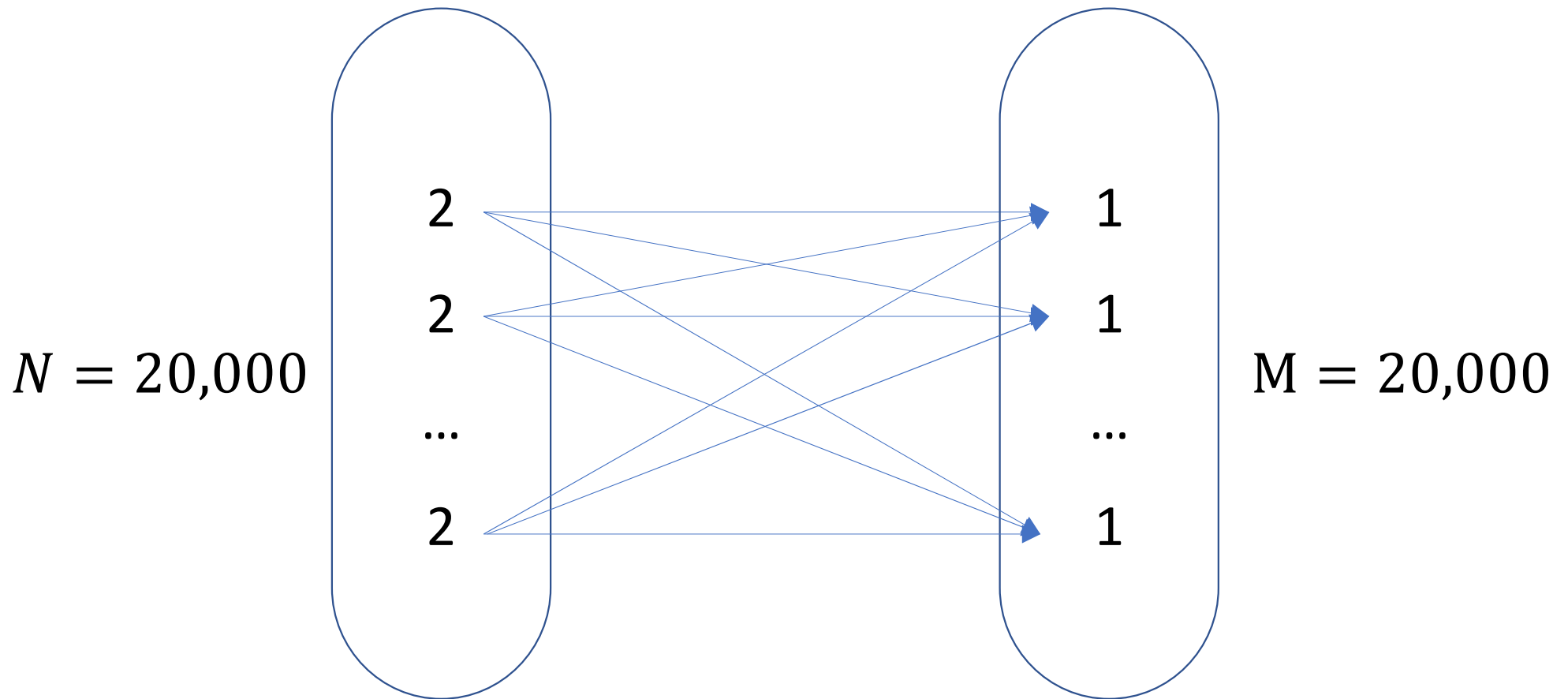
난이도: 2

$N \leq 20,000$

$M \leq 20,000$



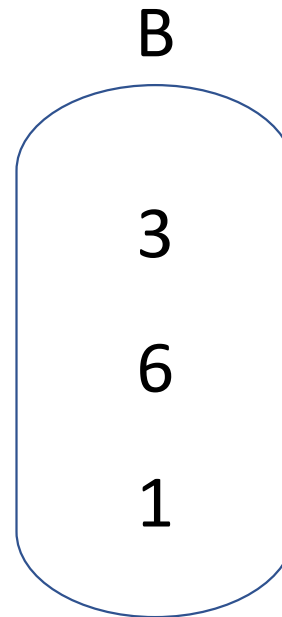
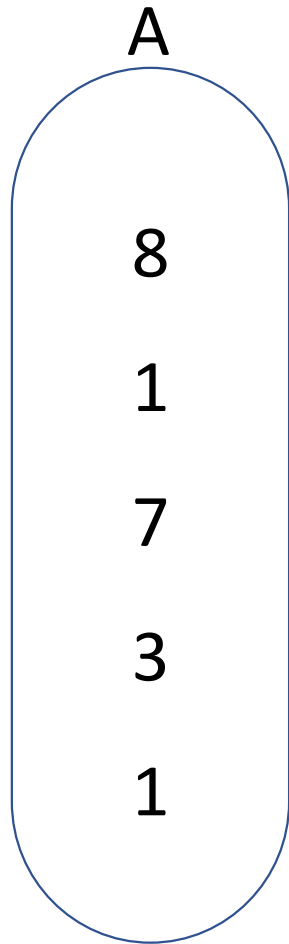
# I 문제 파악하기 - 정답의 최대치



모든 쌍이 정답인 경우!  $\Rightarrow N * M = 4억 \Rightarrow \text{Integer!}$



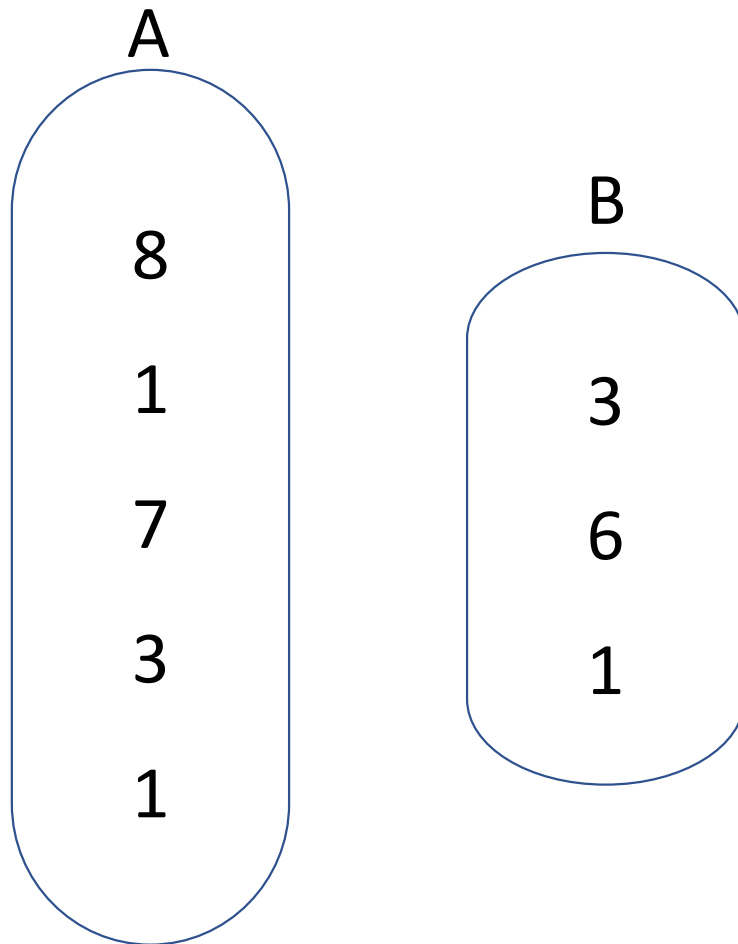
## I 접근 - 가장 쉬운 방법 $O(NM)$



**해야 하는 일:**

B 에서 A[i] 미만 원소들 찾기

# I 접근 - 탐색을 빨리 하기! $O((N + M) \log M)$



## <생각의 흐름>

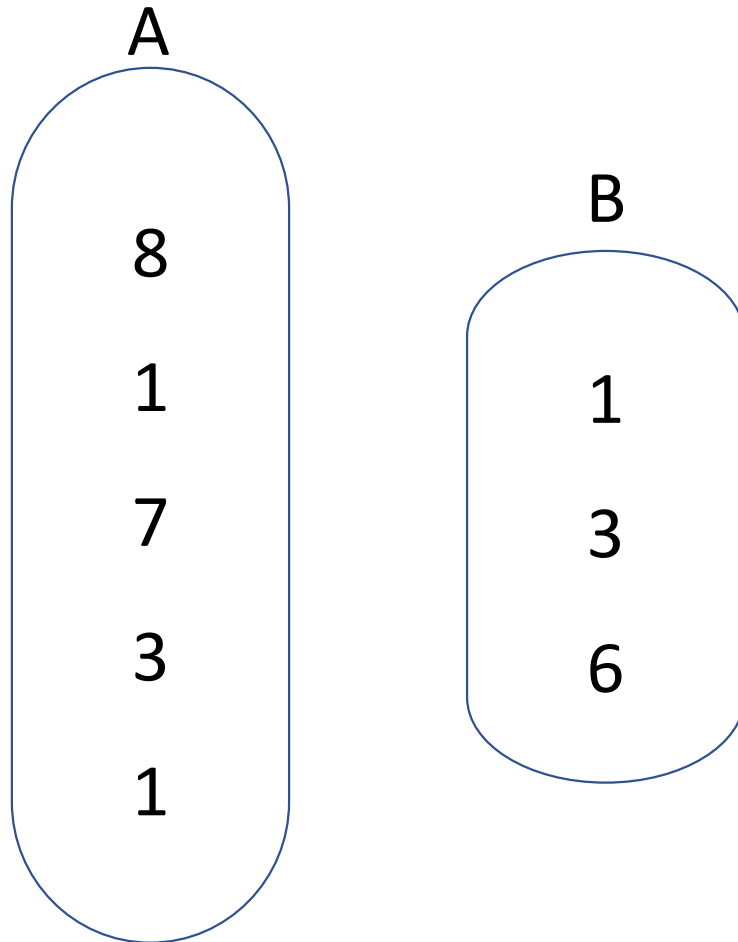
B 에서 A[i] 이하 원소들을 **빨리** 찾기

이걸 빨리 해주는 게 뭐가 있을까?

“이분 탐색”을 쓰자!

이를 위해서 전제 조건(정렬된 상태) 만족!

## I 접근 - 탐색을 빨리 하기! $O((N + M) \log M)$



이를 위해서 전제 조건(정렬된 상태) 만족

이제 B가 정렬되어 있으니까 원하는 원소를 빠르게( $\log M$ ) 탐색이 가능하네!

A의 원소마다 이분 탐색  $\Rightarrow O(N \log M)$

# I 시간, 공간 복잡도 계산하기

1. B 배열 정렬 한 번  $\Rightarrow O(M \log M)$
2. 모든 A의 원소마다, B 배열에 대해 이분 탐색 필요  
 $\Rightarrow O(N \log M)$
3. 총 시간 복잡도:  $O((N + M) \log M)$

## 구현

```
static int lower_bound(int[] A, int L, int R, int X) {  
    // A[L...R] 에서 X 미만의 수(X 보다 작은 수) 중 제일 오른쪽 인덱스를 return 하는 함수  
    // 그런 게 없다면 L - 1 을 return 한다  
}  
  
static void pro() {  
    // B 배열에 대해 이분탐색을 할 예정이니까, 정렬을 해주자!  
    // TODO  
  
    int ans = 0;  
    for (int i = 1; i <= N; i++) {  
        // A[i] 를 선택했을 때, B 에서는 A[i]보다 작은 게 몇 개나 있는 지 count하기  
        ans += /* TODO */;  
    }  
    System.out.println(ans);  
}
```

# I 연습 문제

- BOJ 1920 – 수 찾기
- BOJ 1764 – 듣보잡
- BOJ 3273 – 두 수의 합
- BOJ 10816 – 숫자 카드 2

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드

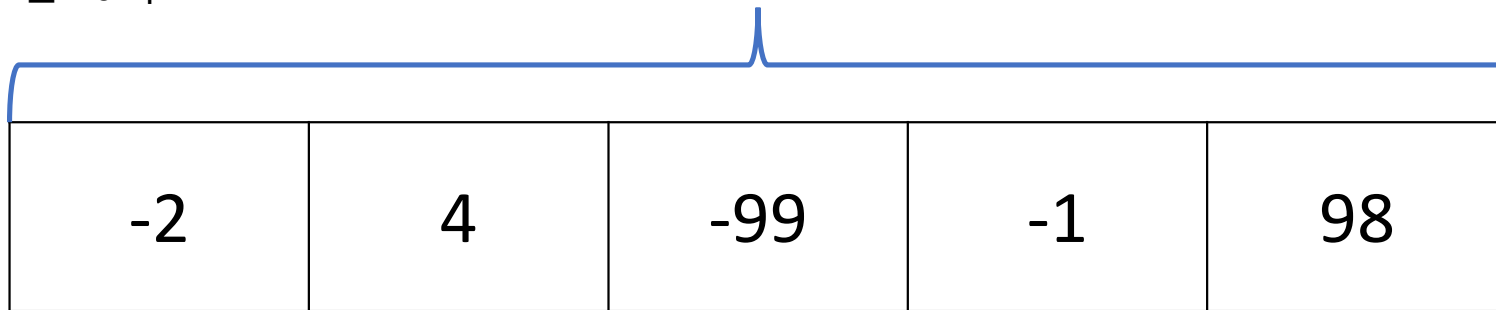
## I BOJ 2470 – 두 용액

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 문제 파악하기 – 정답의 최대치

10억	10억
-10억	-10억

두 수의 합으로 가능한 범위:  $-20\text{억} \sim 20\text{억}$

Integer의 범위:  $-21\text{억}_{4748\text{만 } 3648} \sim 21\text{억}_{4748\text{만 } 3647}$



# I 접근 – 가장 쉬운 방법 $O(N^2)$

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

**해야 하는 일:**

두 용액을 선택해보기! 전부 다!

I 접근 – 빠른 방법  $O(N \log N)$ 

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

**해야 하는 일:**

왼쪽 용액( $A[\text{left}]$ )을 골랐을 때, 오른쪽 용액은?

$A[\text{left}]$  와 더해서 0에 가장 가까우려면?

$-A[\text{left}]$  와 가까울수록 좋습니다!

I 접근 – 빠른 방법  $O(N \log N)$ 

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

즉,  $A[\text{Left}]$ 를 정했을 때,  $-A[\text{Left}]$  랑 가장 가까운 걸 빨리 찾자!

어디서?  $A[(\text{Left}+1) \dots N]$  에서!

정렬의 특성을 되새겨보자

## 특성



각 원소마다, 가장 가까운 원소는 자신의 양 옆 중에 있다.

## I 접근 – 빠른 방법 $O(N \log N)$

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

### 정렬 해보기! $O(N \log N)$

두 가지 이득을 취할 수 있습니다.

1. 이분 탐색 사용 가능!
2. 가장 가까운 원소를 빠르게 찾기 가능!

I 접근 – 빠른 방법  $O(N \log N)$ 

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

**Result** :=  $A[Left+1 \dots N]$  에서  $X = -A[Left]$  이상의 원소 중 가장 왼쪽 위치  
만약 그런 게 없다면  $N+1$

$A[Result-1]$  와  $A[Result]$  중에  $X$  랑 가장 가까운 원소가 있다!

대신  $Result-1$ 과  $Result$  중에  $Left+1$  이상  $N$  이하인 것만 가능한 원소이다.

# I 시간, 공간 복잡도 계산하기

1. 배열 정렬 한 번  $\Rightarrow O(N \log N)$
2. 모든 원소마다 Left로 정하고,  $-A[\text{Left}]$  를 이분탐색하기  $\Rightarrow O(N \log N)$
3. 총 시간 복잡도:  $O(N \log N)$

## 구현

```

static int lower_bound(int[] A, int L, int R, int X) {
    // A[L...R] 에서 X 이상의 수 중 제일 왼쪽 인덱스를 return 하는 함수
    // 그런 게 없다면 R + 1 을 return 한다
}

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;
    for (int left = 1; left <= N - 1; left++) {
        // A[left] 용액을 쓸 것이다. 고로 -A[left] 와 가장 가까운 용액을 자신의 오른쪽 구간에서 찾자.
        int candidate = lower_bound(A, L: left+1, N, -A[left]);

        // A[candidate - 1] 와 A[candidate] 중에 A[left] 와 섞었을 때의 정보를 정답에 갱신시킨다.
    }
    sb.append(v1).append(' ').append(v2);
    System.out.println(sb);
}

```