

Chapter. 02

알고리즘

| 두 포인터 Two Pointers

FAST CAMPUS
ONLINE

알고리즘 공채 대비반 I

강사. 류호석

Chapter. 02

알고리즘

↙ 두 포인터(Two Pointers) - 응용편 ↘

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!

많은 연습을 필요로 합니다!

I [BOJ 13144 - List of Unique Numbers](#)

난이도: 4

$1 \leq N < 100,000$

$1 \leq \text{각 원소} \leq 100,000$

원소의 개수, $N = 5$

1	2	3	4	5
1	2	3	1	2

수열에서 연속한 1개 이상의 수를 뽑았을 때 같은 수가 여러 번 등장하지 않는 경우의 수

I 문제 파악하기 - 정답의 최대치

1	2	3	...	99,999	100,000
1	2	3	...	99999	100000

$$N = 100,000$$

모든 연속 구간이 모두 정답에 카운트 된다!

$$\text{그러한 개수} = N + (N - 1) + \dots + 2 + 1 \cong 50\text{억} \rightarrow \text{Long}$$

I 접근 – 가장 쉬운 방법 $O(N^3)$

1	2	3	4	5
1	2	3	1	2

1. 왼쪽 시작 L 결정 $\rightarrow O(N)$
2. 오른쪽 끝을 R 을 L 부터 시작해서 이동 $\rightarrow O(N)$
3. R 을 이동해서 추가된 원소가 $[L, R - 1]$ 안에 있는지 확인 $\rightarrow O(N)$
4. 총 $O(N^3)$

I 접근 – 개선된 방법 $O(N^2)$


1	2	3	4	5
1	2	3	1	2

3. R 을 이동해서 추가된 원소가 $[L, R - 1]$ 안에 있는 지 확인

숫자마다 $[L, R]$ 안에 몇 개나 있는 지를 직접 세자!

I 접근 - 개선된 방법 $O(N^2)$

1	2	3	4	5
1	2	3	1	2



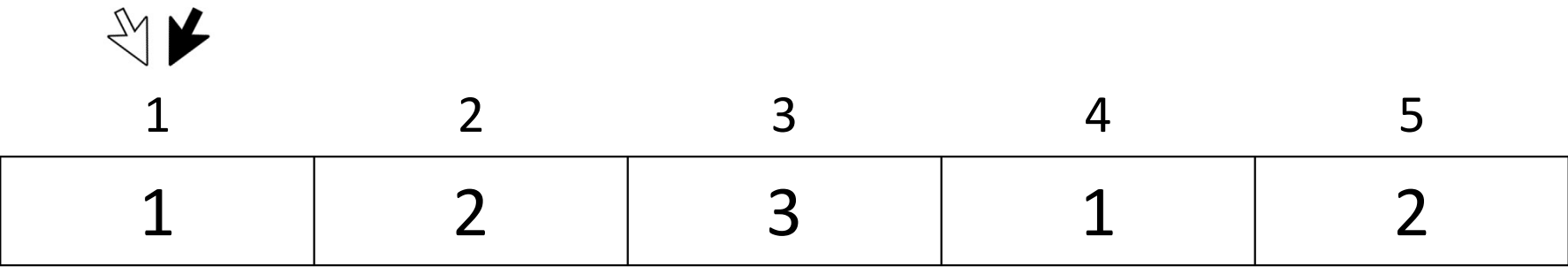
	1	2	3	4	5	...	99,999	100,000
Count	1	1	1	0	0	...	0	0

I 접근 – 개선된 방법 $O(N^2)$

1	2	3	4	5
1	2	3	1	2

1. 왼쪽 시작 L 결정 $\rightarrow O(N)$
2. 오른쪽 끝을 R 을 L 부터 시작해서 이동 $\rightarrow O(N)$
3. R 을 이동해서 추가된 원소가 $[L, R - 1]$ 안에 있는지 확인 $\rightarrow O(1)$
4. 총 $O(N^2)$

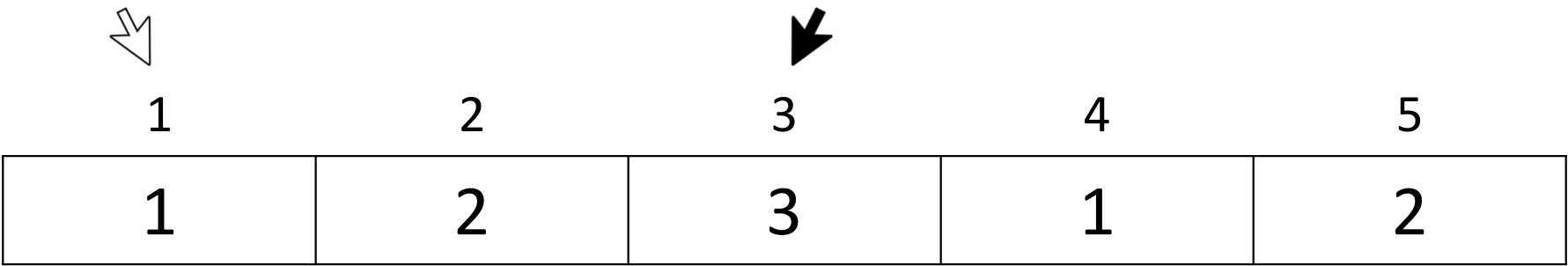
I 접근 – 투 포인터 방법 $O(N)$



	1	2	3	4	5	...	99,999	100,000	
Count				0	0	...		0	0

정답: 0

I 접근 – 투 포인터 방법 $O(N)$



	1	2	3	4	5	...	99,999	100,000
Count	1	1	1	0	0	...	0	0

정답: +3

I 접근 – 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2

1	2	3	4	5	...	99,999	100,000
0	1	1	0	0	...	0	0

정답: 3

I 접근 - 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2

	1	2	3	4	5	...	99,999	100,000
Count	1	1	1	0	0	...	0	0

정답: $3 + 3 = 6$

I 접근 – 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2

1	2	3	4	5	...	99,999	100,000
1	0	1	0	0	...	0	0

정답: 6

I 접근 – 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2

1	2	3	4	5	...	99,999	100,000
1	1	1	0	0	...	0	0

정답: $6 + 3 = 9$

I 접근 – 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2

1	2	3	4	5	...	99,999	100,000
1	1	0	0	0	...	0	0

정답: $9 + 2 = 11$

I 접근 – 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2



	1	2	3	4	5	...	99,999	100,000
Count	0	1	0	0	0	...	0	0

정답: $11 + 1 = 12$

I 접근 – 투 포인터 방법 $O(N)$

↖ := L, 구간의 왼쪽 끝

↘ := R, 구간의 오른쪽 끝

1	2	3	4	5
1	2	3	1	2

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

I 접근 – 투 포인터 방법 $O(N)$

1	2	3	4	5
1	2	3	1	2

1. 왼쪽 시작 L 결정 $\rightarrow O(N)$
2. 오른쪽 끝을 R 을 **이전의 R** 부터 시작해서 이동
3. R 을 이동해서 추가된 원소가 $[L, R - 1]$ 안에 있는 지
확인 $\rightarrow O(1)$
4. 총 $O(N)$

구현

```
static void pro() {  
    long ans = 0;  
  
    for (int L=1, R=0; L<=N; L++){ // L 마다 R 을 최대한 옮겨 줄 계획이다.  
        // R 을 옮길 수 있는 만큼 옮긴다.  
        /* TODO */  
  
        // 정답을 갱신한다  
        /* TODO */  
  
        // L 을 옮겨주면서 A[L] 의 개수를 감소시킨다.  
        /* TODO */  
    }  
  
    System.out.println(ans);  
}
```

I [BOJ 1253-좋다](#)

난이도: 2

$1 \leq N < 2,000$

$-10^9 \leq \text{각 원소} \leq 10^9$

원소의 개수, $N = 6$

1	2	3	4	5	6
5	1	-3	6	4	-5

I 문제 파악하기 – 정답의 최대치

1	2	3	...	99,999	100,000
0	0	0	...	0	0

$$N = 100,000$$

정답이 N 이하이므로 Integer 범위

원소 두 개의 합도 최대 10^9 이므로 Integer 범위

I 접근 – 가장 쉬운 방법 $O(N^3)$

1	2	3	4	5	6
5	1	-3	6	4	-5

1. 타겟 수 결정 $\Rightarrow O(N)$
2. 다른 수 2개 결정해서 만들어지나 확인 $\Rightarrow O(N^2)$
3. $O(N^3)$

I 복기 – 두 용액 복기

두 용액

스페셜 저지

출처

분류

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초 (추가 시간 없음)	128 MB	9472	2454	1824	29.784%

문제

KOI 부설 과학연구소에서는 많은 종류의 산성 용액과 알칼리성 용액을 보유하고 있다. 각 용액에는 그 용액의 특성을 나타내는 하나의 정수가 주어져있다. 산성 용액의 특성값은 1부터 1,000,000,000까지의 양의 정수로 나타내고, 알칼리성 용액의 특성값은 -1부터 -1,000,000,000까지의 음의 정수로 나타낸다.

같은 양의 두 용액을 혼합한 용액의 특성값은 혼합에 사용된 각 용액의 특성값의 합으로 정의한다. 이 연구소에서는 같은 양의 두 용액을 혼합하여 특성값이 0에 가장 가까운 용액을 만들려고 한다.

예를 들어, 주어진 용액들의 특성값이 [-2, 4, -99, -1, 98]인 경우에는 특성값이 -99인 용액과 특성값이 98인 용액을 혼합하면 특성값이 -1인 용액을 만들 수 있고, 이 용액이 특성값이 0에 가장 가까운 용액이다. 참고로, 두 종류의 알칼리성 용액만으로도 혹은 두 종류의 산성 용액만으로 특성값이 0에 가장 가까운 혼합 용액을 만드는 경우도 존재할 수 있다.

산성 용액과 알칼리성 용액의 특성값이 주어졌을 때, 이 중 두 개의 서로 다른 용액을 혼합하여 특성값이 0에 가장 가까운 용액을 만들어내는 두 용액을 찾는 프로그램을 작성하시오.

I 접근 – 가장 쉬운 방법 $O(N^2)$

1	2	3	4	5	6
5	1	-3	6	4	-5

1. Step 3. 를 위해 정렬 한 번 하기 $\rightarrow O(N \log N)$
2. 타겟 수 결정 $\rightarrow O(N)$
3. 다른 수 2개 결정해서 만들어지나 확인 $\rightarrow O(N)$
4. 최종적으로 $O(N^2)$

구현

```
// target_idx 번째 원소가 서로 다른 두 수의 합으로 표현이 되는가?
static boolean func(int target_idx) {
    int L = 1, R = N;
    int target = A[target_idx];
    while (L < R) {
        /* TODO */
    }
    return false;
}

static void pro() {
    // 최소, 최대를 빠르게 알기 위한 정렬
    /* TODO */

    int ans = 0;
    for (int i = 1; i <= N; i++) {
        // i 번째 원소가 서로 다른 두 수의 합으로 표현이 되는가?
        /* TODO */
    }
    System.out.println(ans);
}
```

I 연습 문제

- BOJ 2473 – 세 용액

이외의 추천 문제가 추가되면 Github 자료에 코드 업로드

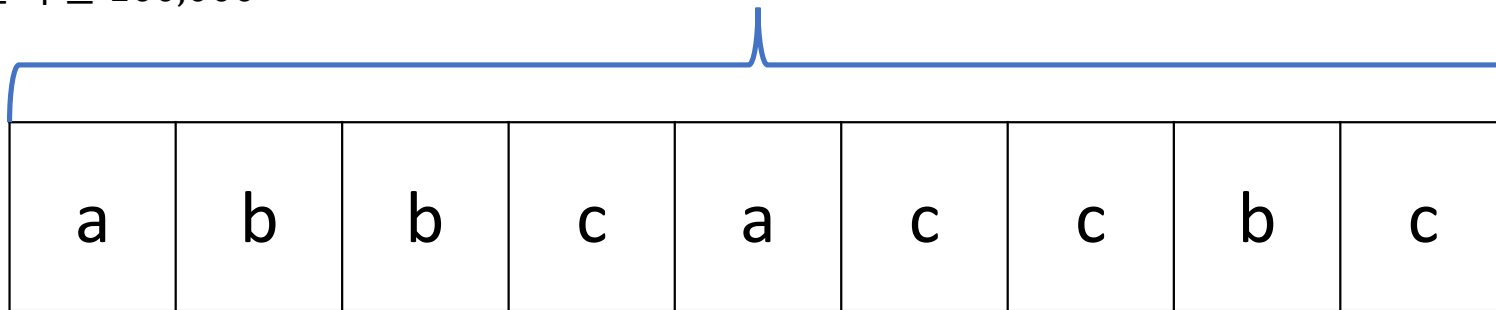
| BOJ 16472 - 고양이

난이도: 3

$1 \leq \text{알파벳 종류}, N \leq 26$

$1 \leq \text{문자열 길이} \leq 100,000$

문자열



최대 N개의 종류의 알파벳을 가진 연속된 문자열밖에 인식하지 못한다.

인식할 수 있는 최대 문자열의 길이는 얼마인지가 궁금해졌다.




I 문제 파악하기 – 정답의 최대치

$N = 26$ 이라면 문자열 전체를 인식하므로,
최대 길이인 10만이 정답이다.

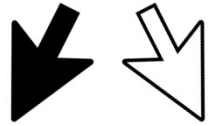
I 접근 - 바로 투 포인터 접근으로!

a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z
cnt	0	0	0	0		...	0	0

 R:= 인식하고 싶은 구간의 오른쪽 끝 
 L:= 인식 가능한 가장 왼쪽 위치

I 접근 - 바로 투 포인터 접근으로!

 $N = 2$

a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z
cnt	0	0	0	0		...	0	0

Kind := [L, R] 사이의 알파벳 종류

= cnt 배열에서 0이 아닌 것의 개수!

I 접근 - 바로 투 포인터 접근으로!

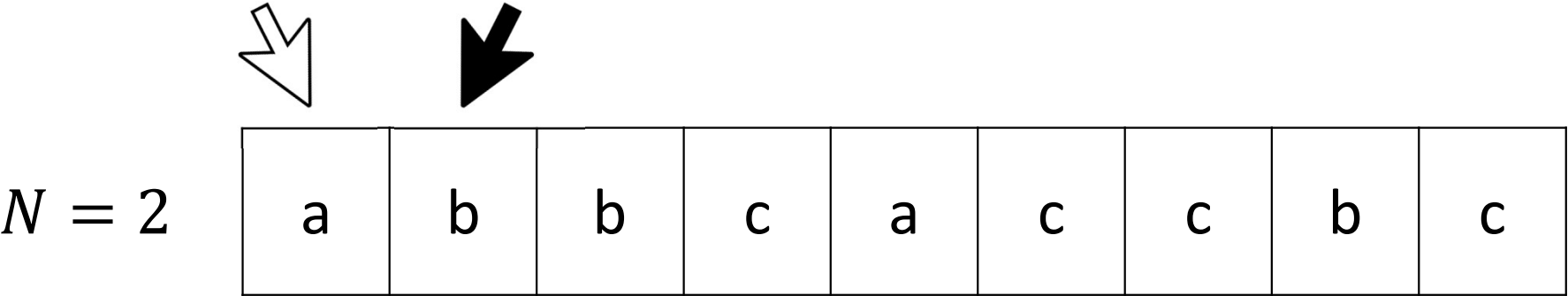
 $N = 2$

a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z
cnt	1	0	0	0		...	0	0

Kind := 1

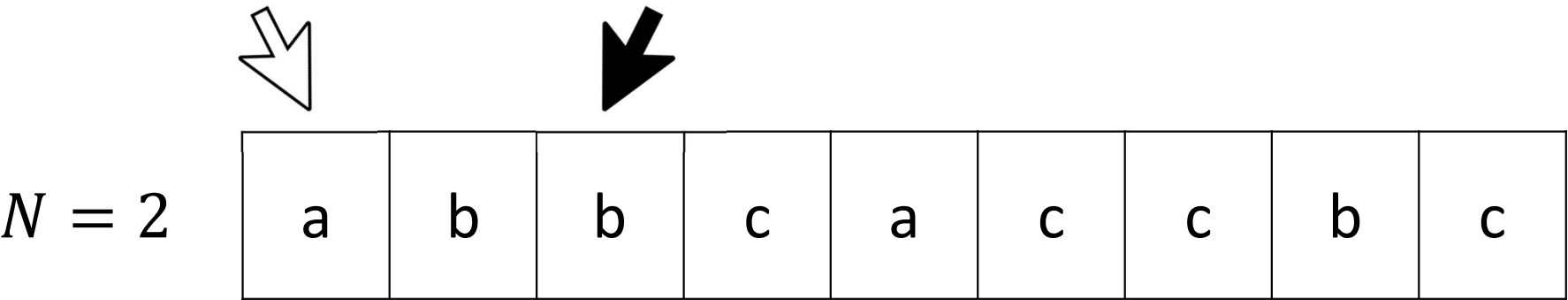
I 접근 - 바로 투 포인터 접근으로!



	a	b	c	d	...	y	z
cnt	1	1	0	0	...	0	0

Kind := 2

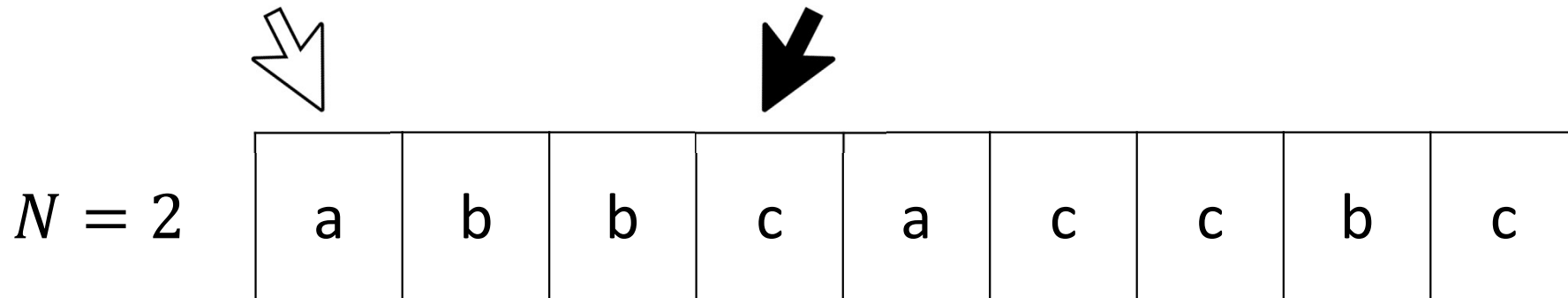
I 접근 - 바로 투 포인터 접근으로!



	a	b	c	d	...	y	z
cnt	1	2	0	0	...	0	0

Kind := 2

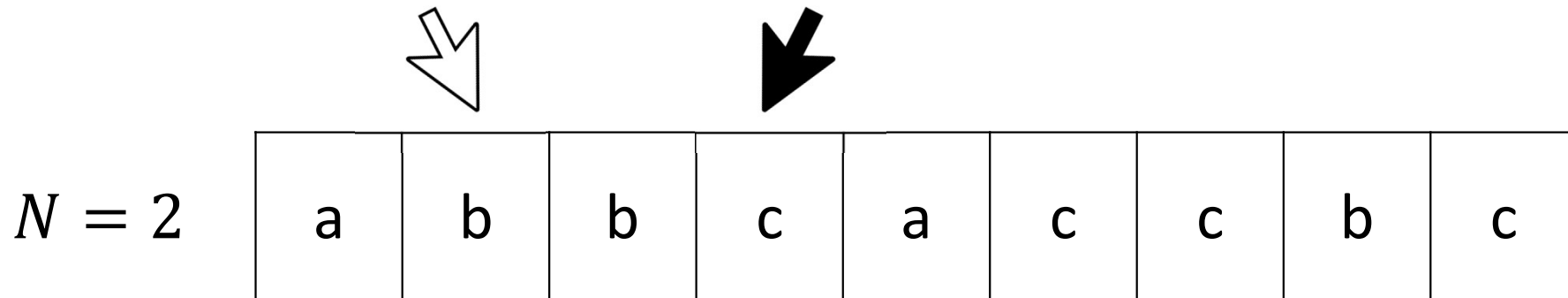
I 접근 - 바로 투 포인터 접근으로!



	a	b	c	d	...	y	z
cnt	1	2	1	0	...	0	0

Kind := 3!!!!

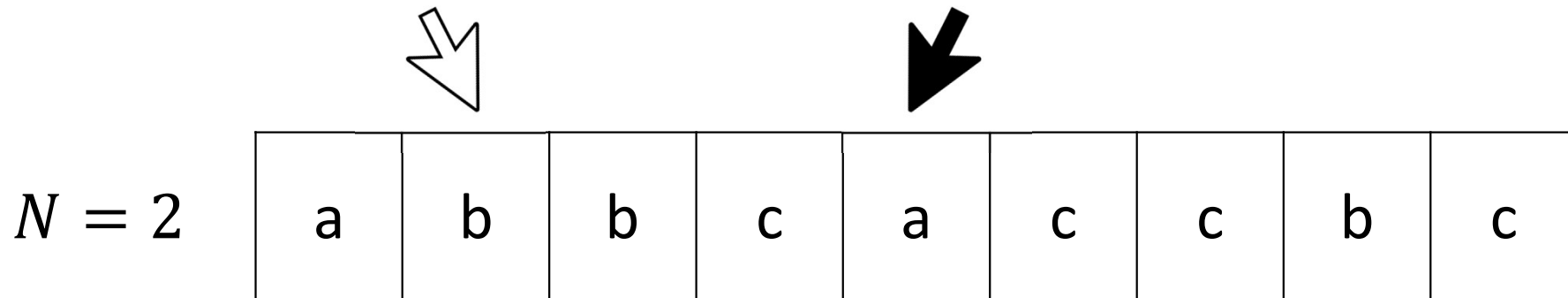
I 접근 - 바로 투 포인터 접근으로!



	a	b	c	d	...	y	z
cnt	0	2	1	0	...	0	0

Kind := 2

I 접근 - 바로 투 포인터 접근으로!



	a	b	c	d	...	y	z
cnt	1	2	1	0	...	0	0

Kind := 3!!!

I 접근 - 바로 투 포인터 접근으로!



$N = 2$

a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...	y	z
cnt	1	1	1	0	...	0	0

Kind := 3!!!

I 접근 - 바로 투 포인터 접근으로!


 $N = 2$

a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...	y	z
cnt	1	0	1	0	...	0	0

Kind := 2

I 접근 - 바로 투 포인터 접근으로!

 $N = 2$


a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z	
cnt	1	0	2	0		...		0	0

Kind := 2

I 접근 - 바로 투 포인터 접근으로!

 $N = 2$


a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z	
cnt	1	0	3	0		...		0	0

Kind := 2

I 접근 - 바로 투 포인터 접근으로!

$N = 2$




a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...	y	z
cnt	1	1	3	0	...	0	0

Kind := 3!!!

I 접근 - 바로 투 포인터 접근으로!

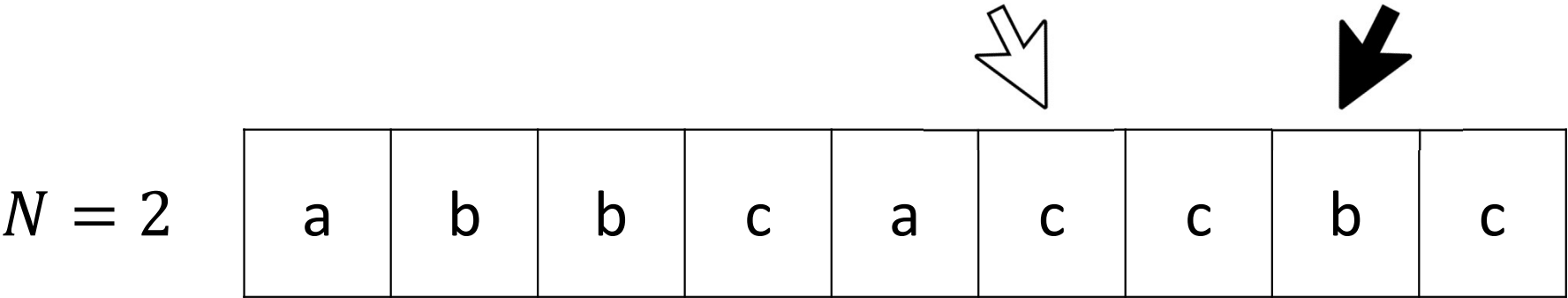
 $N = 2$


a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z
cnt	1	1	2	0		...	0	0

Kind := 3!!!


I 접근 - 바로 투 포인터 접근으로!



	a	b	c	d	...	y	z
cnt	0	1	2	0	...	0	0

Kind := 2

I 접근 - 바로 투 포인터 접근으로!

 $N = 2$


a	b	b	c	a	c	c	b	c
---	---	---	---	---	---	---	---	---

	a	b	c	d	...		y	z	
cnt	0	1	3	0		...		0	0

Kind := 2

I 시간, 공간 복잡도 계산하기

1. R을 하나씩 이동시키면서 L을 조절하기 $\Rightarrow O(N)$
2. Kind 를 $O(26)$ 에 계산하거나 $O(1)$ 에 계산 가능!
3. 총 시간 복잡도: $O(N)$

I 구현

```
static void pro() {  
    int len = A.length(), ans = 0;  
    for (int R = 0, L = 0; R < len; R++) {  
        // R 번째 문자를 오른쪽에 추가  
        /* TODO */  
  
        // 불가능하면, 가능할 때까지 L을 이동  
        /* TODO */  
  
        // 정답 갱신  
        /* TODO */  
    }  
    System.out.println(ans);  
}
```