

#중급반 5주차

스위핑, 슬라이딩 윈도우, 투 포인터

<Sweeping, Sliding window, Two pointer>

2p /<스위핑>

14p /<슬라이딩 윈도우>

9p /<투 포인터>

27p /<연습 문제>

#스위핑

<Sweeping Algorithm> - 소개

스위핑 알고리즘 : 정렬된 데이터를 쓸고 가면서 문제를 푸는 알고리즘

스위핑 알고리즘은 Greedy나 DP처럼 개념 자체는 매우 간단한 알고리즘입니다.

쓸고 간다는 표현이 조금 어색할 수 있는데, 다르게 말하면 모든 데이터를 한번만 확인하는 것입니다.

즉, 데이터를 앞에서부터 훑고 지나가면서 문제를 푸는데 필요한 처리를 하면 문제가 풀리게 됩니다.

처리에 관한 내용은 문제에 따라 많이 달라지게 됩니다.

따라서, 예를 통해서 감을 잡아보겠습니다.



#스윙핑

<Sweeping Algorithm> - 연습문제

선 긋기 (BOJ #2170)

<문제 설명>

- 선을 그은 횟수와 양 끝점이 주어진다.
- 그은 선의 총 길이를 출력하는 문제
- 단, 여러 번 그은 선과 한번 그은 선을 비교할 수 없다.

<제약 조건>

- $1 \leq N \leq 1,000,000$
- $-10^9 \leq x, y \leq 10^9$
- 메모리 제한 192MB

#스위핑

<Sweeping Algorithm> - 연습문제

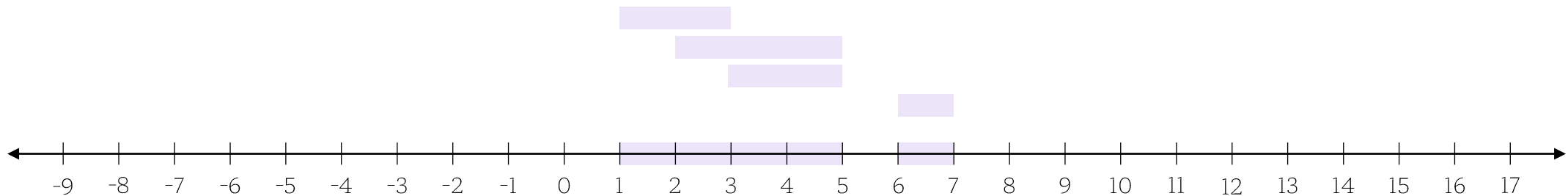
5 선긋기 (BOJ #2170)

<문제 해설>

먼저, 예제를 한번 살펴보겠습니다.

예제를 그림으로 그려보면 다음과 같습니다.

1부터 5까지 그어진 길이 4의 직선과 6부터 7까지 그어진 길이 1의 직선이 합쳐져 답은 5가 됩니다.



#스위핑

<Sweeping Algorithm> - 연습문제

5 선긋기(BOJ #2170)

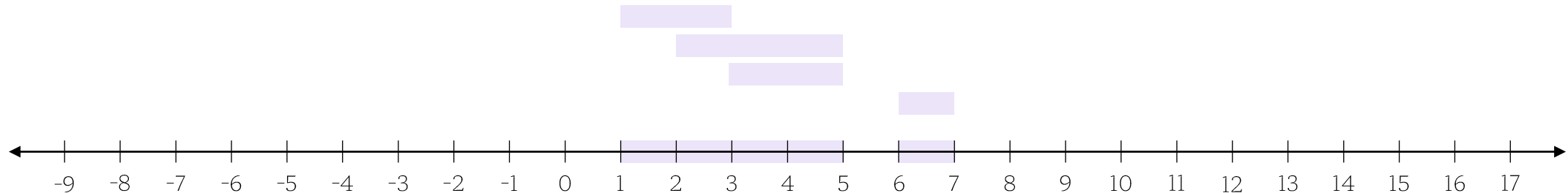
<문제 해설>

각 구간의 상태를 배열로 저장하면 어떻게 될까요?

즉, -10^9 부터 $-10^9 + 1$ 까지의 구간을 $arr[0]$ 으로 두고,

각각의 구간이 칠해질 때마다 $arr[x]$ 의 값을 바꾸는 방식을 생각해볼 수 있습니다.

하지만, 보통 10^9 이상의 큰 배열을 만들면 메모리 제한에 걸려 MLE를 받게 됩니다.



#스위핑

<Sweeping Algorithm> - 연습문제

5 선긋기(BOJ #2170)

<문제 해설>

따라서, 정렬을 한 후 스위핑을 통해 문제를 풀 것입니다.

정확한 순서는 다음과 같습니다.

1. 구간을 시작하는 점이 작은 순서대로 정렬
 2. 구간을 확인할 때마다 각각의 끝 점을 저장
 - 3-1. 만약 구간이 이미 색칠한 곳부터 시작한다면 시작 점을 그대로 하고 끝 점만 저장
 - 3-2. 만약 구간이 새로운 곳부터 시작한다면 지금까지 체크한 구간을 답에 더한 후 시작 점을 갱신
- 예제를 보면서 위의 순서를 따라가보겠습니다.

#스위핑

<Sweeping Algorithm> - 연습문제

5 선긋기(BOJ #2170)

<문제 해설>

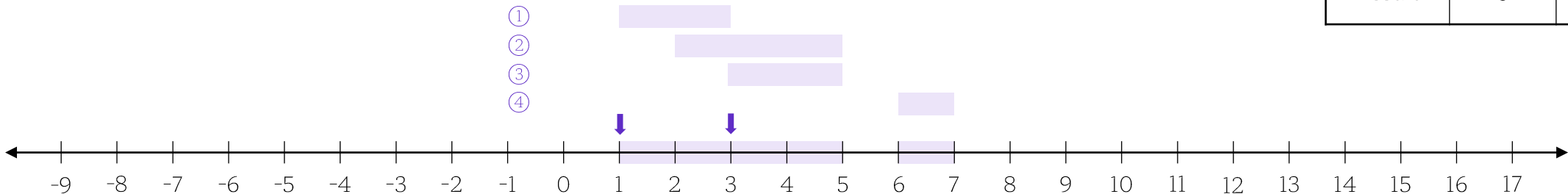
예제는 이미 구간이 정렬되어 있는 상태이므로, 위에서부터 번호를 붙이겠습니다. (① ~ ④)

첫 번째로, 1번 구간을 확인합니다. 새로운 구간이므로, 시작과 끝을 모두 저장합니다.

Start와 End의 첫 값은 임의의 아주 작은 값으로 저장하면 됩니다. 두 값이 같아야 첫 구간을 볼 때 Result가 변하지 않습니다.

<변수들의 이름과 값>

Name	Before	After
Start	-MAX	1
End	-MAX	3
Result	0	0



#스위핑

<Sweeping Algorithm> - 연습문제

5 선긋기(BOJ #2170)

<문제 해설>

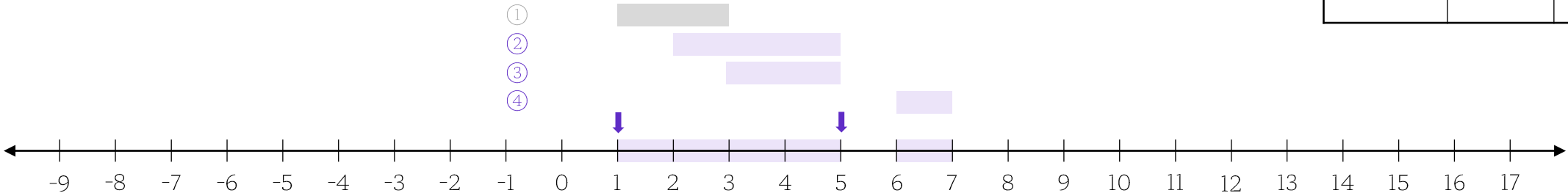
두 번째 구간은 Start와 End 사이의 값에서 시작합니다.

따라서, End만 더 큰 값으로 갱신해줍니다.

max 함수를 이용해서 무조건 더 큰 값으로 만들어줘야 합니다. 두 번째 구간의 값이 더 작을 수도 있습니다.

<변수들의 이름과 값>

Name	Before	After
Start	1	1
End	3	5
Result	0	0



#스위핑

<Sweeping Algorithm> - 연습문제

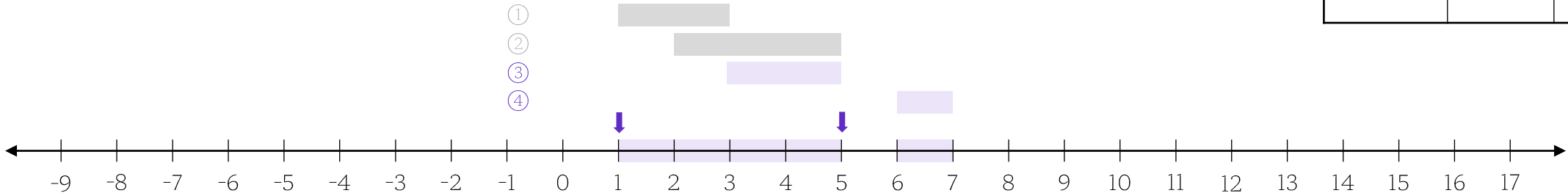
5 선긋기(BOJ #2170)

<문제 해설>

세 번째 구간도 Start와 End 사이의 값에서 시작합니다.
따라서, End만 더 큰 값으로 갱신해줍니다.
이미 After가 5이기 때문에 바뀌지 않습니다.

<변수들의 이름과 값>

Name	Before	After
Start	1	1
End	5	5
Result	0	0



#스위핑

<Sweeping Algorithm> - 연습문제

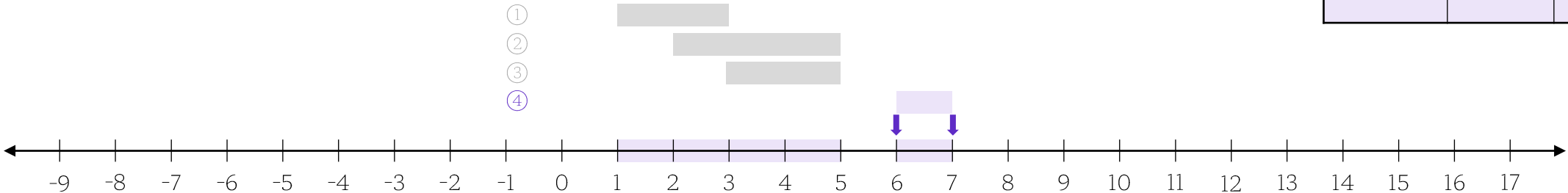
5 선긋기(BOJ #2170)

<문제 해설>

네 번째 구간은 End보다 더 큰 값에서 시작됩니다.
따라서, 원래 구간의 크기를 Result에 더해준 후, Before, After 모두 바꿔줍니다.

<변수들의 이름과 값>

Name	Before	After
Start	1	6
End	5	7
Result	0	4



#스위핑

<Sweeping Algorithm> - 연습문제

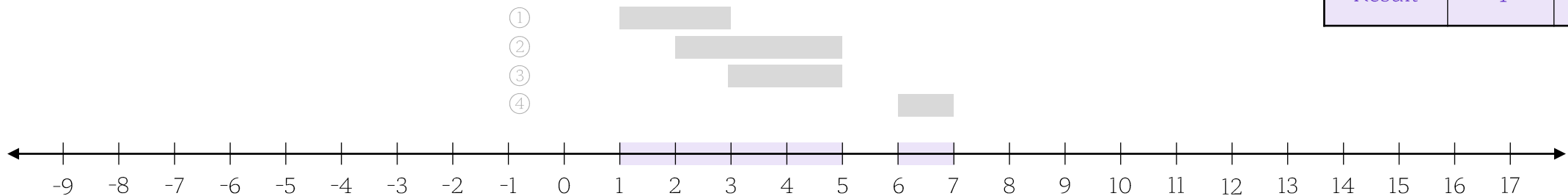
5 선긋기(BOJ #2170)

<문제 해설>

마지막으로, 모든 구간을 확인한 후에는
현재 저장되어 있는 Start, End가 나타내는 구간을 추가로 더해줍니다.
모든 과정을 거치면 Result가 5가 됩니다.

<변수들의 이름과 값>

Name	Before	After
Start	6	6
End	7	7
Result	4	5



#슬라이딩 윈도우

<Sliding window> - 소개

슬라이딩 윈도우 : 일정한 크기의 윈도우를 움직이면서 문제를 푸는 알고리즘

슬라이딩 윈도우는 스�핑 알고리즘과 비슷하지만, 움직이는 윈도우의 크기가 일정하다는 점이 다릅니다.

슬라이딩 윈도우 알고리즘은 다음과 같은 순서로 진행됩니다.

먼저, 윈도우의 크기를 잡습니다. 이때, 윈도우의 크기는 일정해야 합니다.

두 번째로, 윈도우를 움직입니다. 이때, 남길 수 있는 데이터는 남깁니다.

이 남기는 데이터를 통해 시간복잡도를 줄일 수 있고, 일반적으로 시간복잡도는 $O(N)$ 이 됩니다.

슬라이딩 알고리즘도 예제를 통해 자세히 알아보겠습니다.

#슬라이딩 윈도우

<Sliding window> - 연습문제

3 수열 (BOJ #2559)

<문제 설명>

- 온도의 정보를 담은 길이 N 의 수열이 주어진다.
- 이 수열에서 연속된 K 개의 수를 합한다.
- 이때의 합의 최댓값을 구하는 문제

<제약 조건>

- $2 \leq N \leq 100,000$
- $1 \leq K \leq N$
- $-100 \leq t_i \leq 100$

#슬라이딩 윈도우

<Sliding window> - 연습문제

3 수열 (BOJ #2559)

<문제 해설>

첫 번째로, 맨 앞에서부터 K개의 원소를 선택합니다.

5개의 원소의 합은 -12가 됩니다.

answer : -12

3	-2	-4	-9	0	3	7	13	8	-3
---	----	----	----	---	---	---	----	---	----

#슬라이딩 윈도우

<Sliding window> - 연습문제

3 수열 (BOJ #2559)

<문제 해설>

두 번째로, 윈도우를 한 칸 움직입니다.

윈도우 내부의 숫자의 합은 다시 새로 계산할 필요가 없습니다.

대신, 맨 뒤의 원소를 빼고 맨 앞의 원소만 더해주면 됩니다.

answer : $\max(-12, -12)$

3	-2	-4	-9	0	3	7	13	8	-3
---	----	----	----	---	---	---	----	---	----

#슬라이딩 윈도우

<Sliding window> - 연습문제

3 수열 (BOJ #2559)

<문제 해설>

이 과정을 계속 반복하면서 최대가 되는 값만 저장해주면 됩니다.

answer : $\max(-15, 14)$

3	-2	-4	-9	0	3	7	13	8	-3
---	----	----	----	---	---	---	----	---	----

#슬라이딩 윈도우

<Sliding window> - 연습문제

3 수열 (BOJ #2559)

<문제 해설>

끝까지 확인했을 때, 최댓값은 31이 됩니다.

따라서, 최댓값인 31을 출력합니다.

시간복잡도는 $O(N)$ 입니다.

answer : max(31, 28)

3	-2	-4	-9	0	3	7	13	8	-3
---	----	----	----	---	---	---	----	---	----

#투 포인터

<Two pointers> - 소개

투 포인터 : 두 개의 포인터를 움직이면서 문제를 푸는 방법

투 포인터 알고리즘은 두 개의 포인터를 사용하여 시간복잡도를 줄일 수 있습니다.

n 개의 데이터에서 특정 구간에 대해 알아보고 싶을 때, 보통 $O(n^2)$ 의 시간이 소요되지만, 특정한 상황에서는 다음의 두 가지 연산만을 활용할 수 있게 됩니다.

1. 뒤에 있는 포인터를 밀어서 구간을 줄이기
2. 앞에 있는 포인터를 밀어서 구간을 늘리기

따라서, 투 포인터를 사용할 수 있는 문제는 $O(n)$ 시간에 문제를 풀 수 있게 됩니다.



#투 포인터

<Two pointers> - 연습문제

용액 (BOJ #2467)

<문제 설명>

- 용액의 특성값들이 주어진다.
- 두 용액의 특성값의 합을 구한다.
- 이때의 합이 0에 가장 가깝게 하는 두 용액의 특성값을 구하는 문제

<제약 조건>

- $2 \leq N \leq 100,000$
- $-10^9 \leq x_i \leq 10^9$
- x_i 의 값은 서로 다르다

#투 포인터

<Two pointers> - 연습문제

5 용액 (BOJ #2467)

<문제 해설>

첫 번째로, 모든 특성값들을 정렬합니다.

변수 “value”(int) 를 만들고, 지금까지 본 경우 중 최선의 합을 저장합니다.

변수 “answer”(pair<int, int>) 를 만들고, 최선일 때의 두 용액의 특성값을 저장합니다.

value : MAX, answer : {MAX, MAX}

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

#투 포인터

<Two pointers> - 연습문제

5 용액 (BOJ #2467)

<문제 해설>

변수 “st”, “ed”를 만듭니다.

각각 확인할 용액을 가리키는 변수입니다.


처음 보고 있는 st, ed의 정보를 value와 answer에 저장합니다.

value : -1, answer : {-99, 98}

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

#투 포인터

<Two pointers> - 연습문제

 용액 (BOJ #2467)

<문제 해설>

각각의 경우에 따라 움직일 변수를 정합니다.


1. 지금 보고 있는 두 변수의 합이 음수인 경우 : 왼쪽 변수를 한칸 오른쪽으로 움직입니다.
2. 지금 보고 있는 두 변수의 합이 양수인 경우 : 오른쪽 변수를 한칸 왼쪽으로 움직입니다.
3. 지금 보고 있는 두 변수의 합이 0인 경우 : 아무거나 움직여도 이미 답이 정해졌으므로 상관 없습니다.

value : -1, answer : {-99, 98}

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

#투 포인터

<Two pointers> - 연습문제

 용액 (BOJ #2467)

<문제 해설>


만약 저장된 value보다 절댓값이 더 작은 합을 찾았다면 새롭게 저장합니다.
아니면, 포인터를 움직이는 과정을 반복합니다.

value : -1, answer : {-99, 98}

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

#투 포인터

<Two pointers> - 연습문제

 용액 (BOJ #2467)

<문제 해설>

두 포인터가 서로 만나게 되었을 때, 답을 출력합니다.

답은 answer에 저장한 두 값을 출력하면 됩니다.

value : -1, answer : {-99, 98}

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



#연습 문제

정수론

5 아우으 우아으이야!! (BOJ #15922)

== 선 긋기

4 부분합 (BOJ #1806)

고인물 전용 문제 2(Platinum)

3 소수의 연속합 (BOJ #1644)

4주차 정수론을 다시 떠올려봅시다

5 회문 (BOJ #17609)

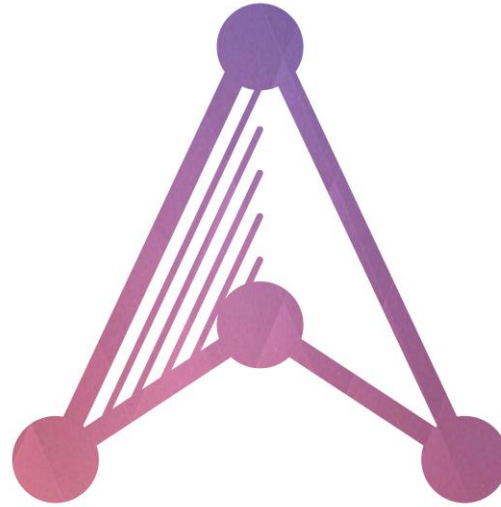
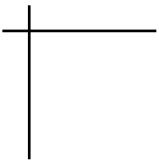
회문 판별의 강화판

3 세 용액 (BOJ #2473)

용액의 강화 버전

4 수 고르기 (BOJ #3988)

앞의 문제들이 시시하다면?



A L O H A
The algorithm club.

