

분할 정보

최백준 choi@startlink.io

분할 정보

분할 정복

Divide & Conquer

3

- 분할 정복은 문제를 2개 또는 그 이상의 작은 부분 문제로 나눈 다음 푸는 것(분할)
- 푸는 다음에는 다시 합쳐서 정답을 구할 때도 있음 (정복)
- 대표적인 분할 정복 알고리즘
- 퀵 소트
- 머지 소트
- 큰 수 곱셈 (카라츠키 알고리즘)
- FFT

분할 정복

Divide & Conquer

- 분할 정복과 다이나믹은
- 문제를 작은 부분 문제로 나눈다는 점은 동일하다
- 분할 정복: 문제가 겹치지 않음
- 다이나믹: 문제가 겹쳐서 겹치는 것을 Memoization으로 해결

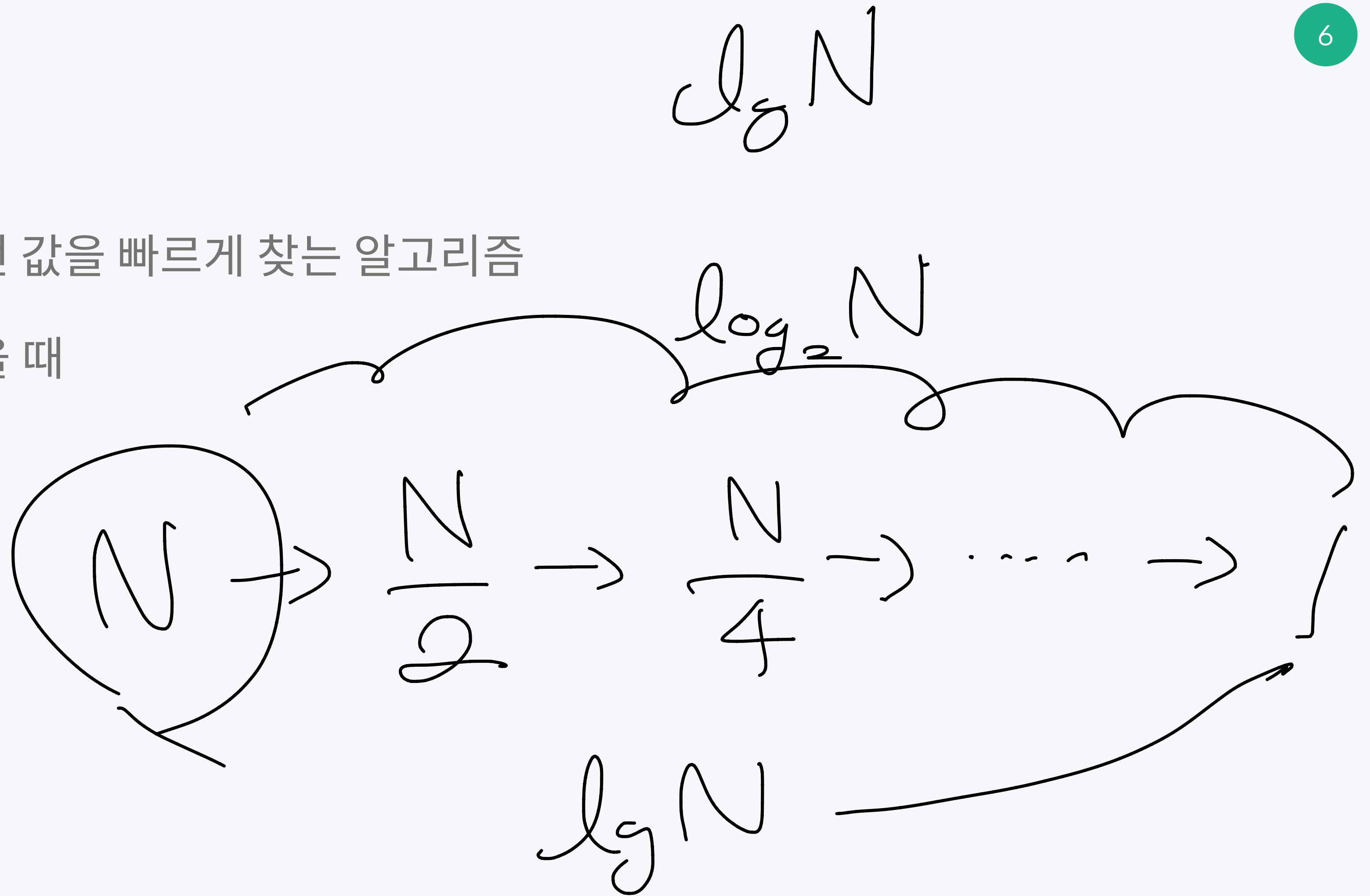
이분 탐색

이분 탐색

Binary Search

6

- 정렬되어 있는 리스트에서 어떤 값을 빠르게 찾는 알고리즘
- 리스트의 크기를 N 이라고 했을 때
- $\lg N$ 의 시간이 걸린다.

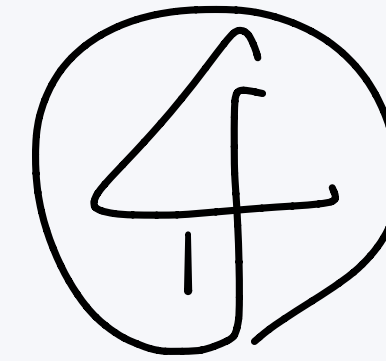


$$2^K = N$$

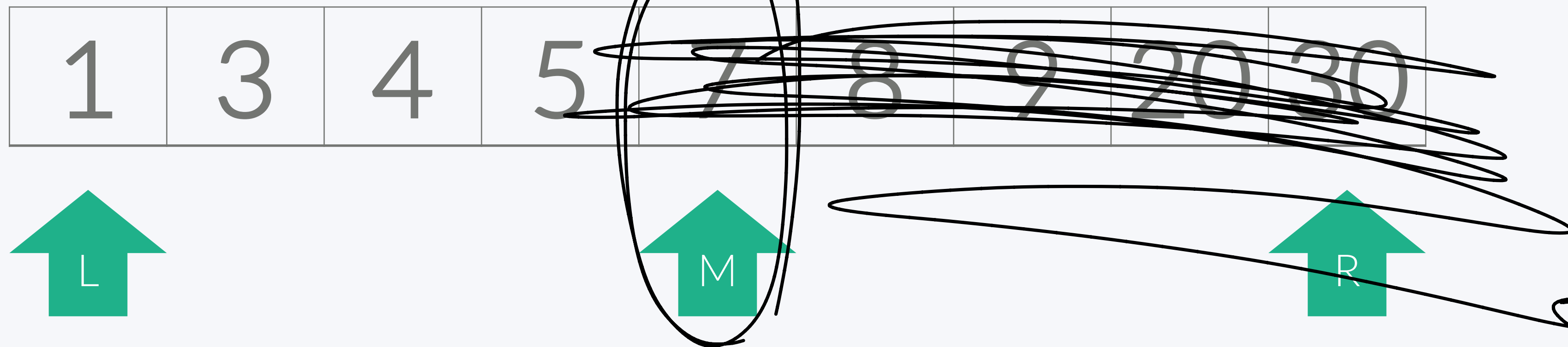
이분 탐색

Binary Search

7



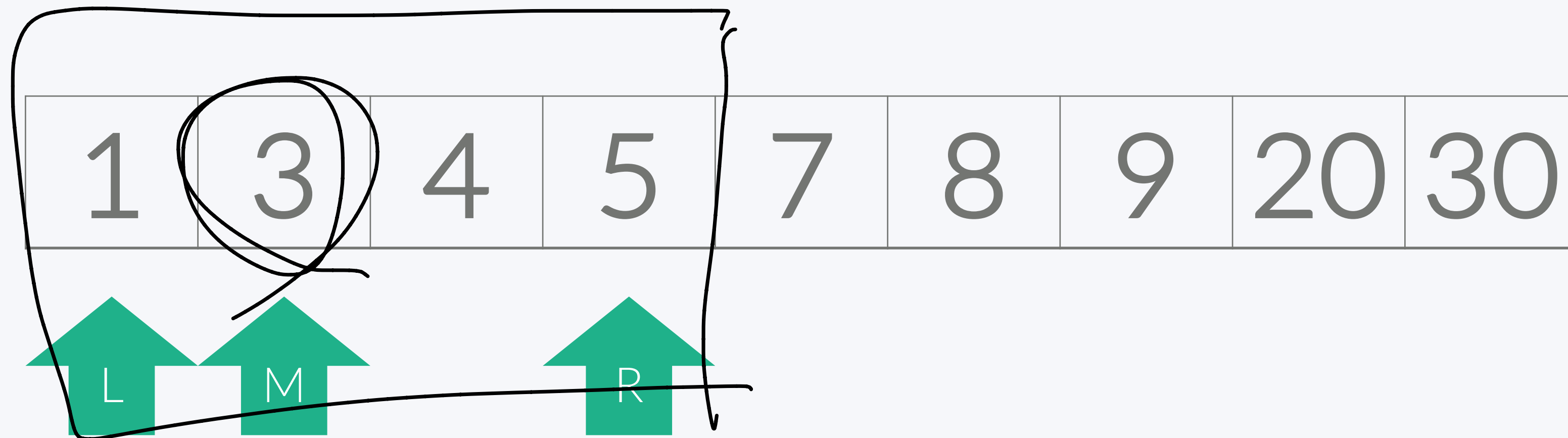
- 4를 찾아보자
- $L = 0, R = 8, M = 4$
- $4 < 7$ 이기 때문에 왼쪽 ($L \sim M-1$)에 4가 있을 수 있다.



이분 탐색

Binary Search

- 4를 찾아보자
- $L = 0, R = 3, M = 1$
- $4 > 3$ 이기 때문에 오른쪽 ($M+1 \sim R$)에 4가 있을 수 있다.

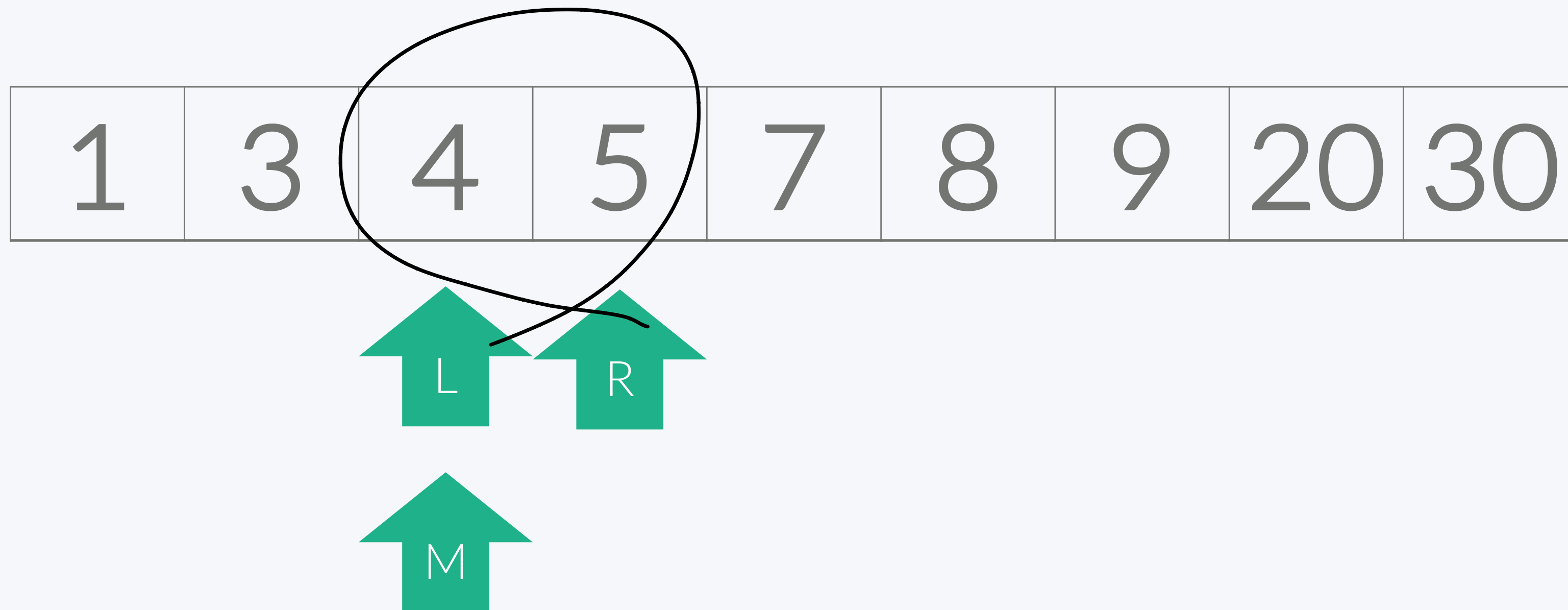


이분 탐색

Binary Search

9

- 4를 찾아보자
- $L = 2, R = 3, M = 2$
- $4 == 4$ 이다. 4를 찾았다.






이분 탐색

10

Binary Search

- 2를 찾아보자
- $L = 0, R = 8, M = 4$
- $2 < 7$ 이기 때문에 왼쪽 ($L \sim M-1$)에 4가 있을 수 있다.

1	3	4	5	7	8	9	20	30
								
L		M			R			

이분 탐색

Binary Search

- 2를 찾아보자
- $L = 0, R = 3, M = 1$
- $2 < 3$ 이기 때문에 왼쪽 ($L \sim M-1$)에 2가 있을 수 있다.

1	3	4	5	7	8	9	20	30
---	---	---	---	---	---	---	----	----



left right
[left, right]
[left, right)

[x, y] \leftrightarrow (y, x)
(r, c)

이분 탐색

12

Binary Search

- 2를 찾아보자
- $L = 0, R = 0, M = 0$
- $2 > 1$ 이기 때문에 오른쪽 ($M+1 \sim R$)에 2가 있을 수 있다.

1	3	4	5	7	8	9	20	30
---	---	---	---	---	---	---	----	----



이분 탐색

Binary Search

13

- 2를 찾아보자
- $L = 1, R = 0, M = 0$
- $L < R$ 이기 때문에, 이분 탐색을 종료한다. 2는 리스트에 없다.

1	3	4	5	7	8	9	20	30
---	---	---	---	---	---	---	----	----



이분 탐색

Binary Search

14

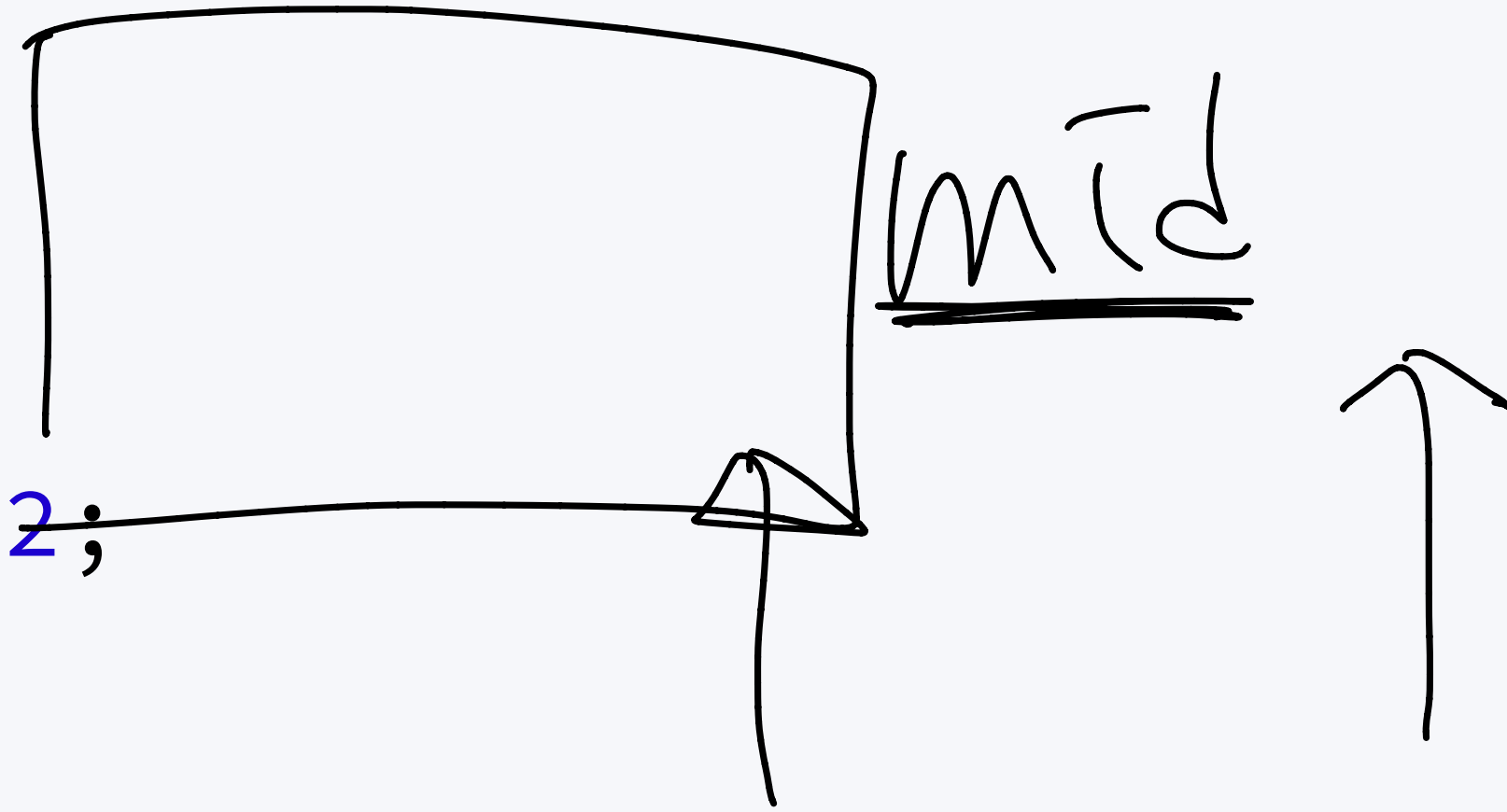
- 정렬되어 있는 리스트에서 어떤 값을 빠르게 찾는 알고리즘
- 리스트의 크기를 N 이라고 했을 때
- $\lg N$ 의 시간이 걸린다.
- 시간 복잡도가 $\lg N$ 인 이유는
- 크기가 N 인 리스트를 계속해서 절반으로 나누기 때문이다.
- $2^k = N$ 일 때, $k = \lg N$

이분 탐색

Binary Search

15

```
while (left <= right) {  
    int mid = (left + right) / 2;  
    if (a[mid] == x) {  
        position = mid;  
        break;  
    } else if (a[mid] > x) {  
        right = mid - 1;  
    } else {  
        left = mid + 1;  
    }  
}
```



숫자 카드

<https://www.acmicpc.net/problem/10815>

- 이분 탐색을 이용해 풀 수 있다.

숫자 카드

17

<https://www.acmicpc.net/problem/10815>

- 스스

숫자 카드 2

<https://www.acmicpc.net/problem/10816>

- 이분 탐색을 이용해 풀 수 있다.

숫자 카드 2

<https://www.acmicpc.net/problem/10816>

- 소스: <http://boj.kr/dc85be1048db4a2880edb42be1971a92>

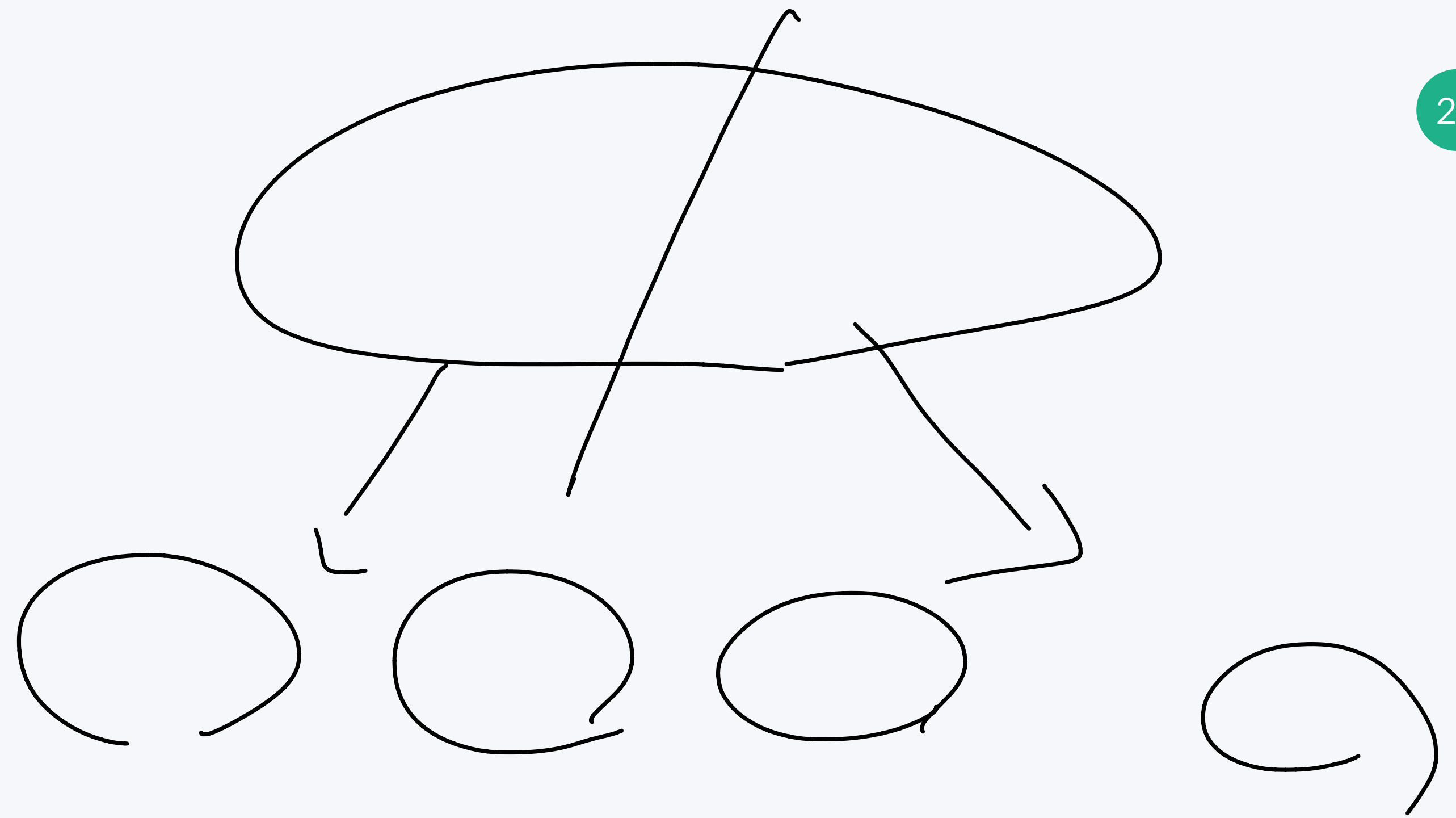
$N \lg N$

머지 소트

머지 소트

Merge Sort

- N개를 정렬하는 알고리즘
- N개를 $N/2$, $N/2$ 개로 나눈다.
- 왼쪽 $N/2$ 와 오른쪽 $N/2$ 를 정렬한다.
- 두 정렬한 결과를 하나로 합친다.

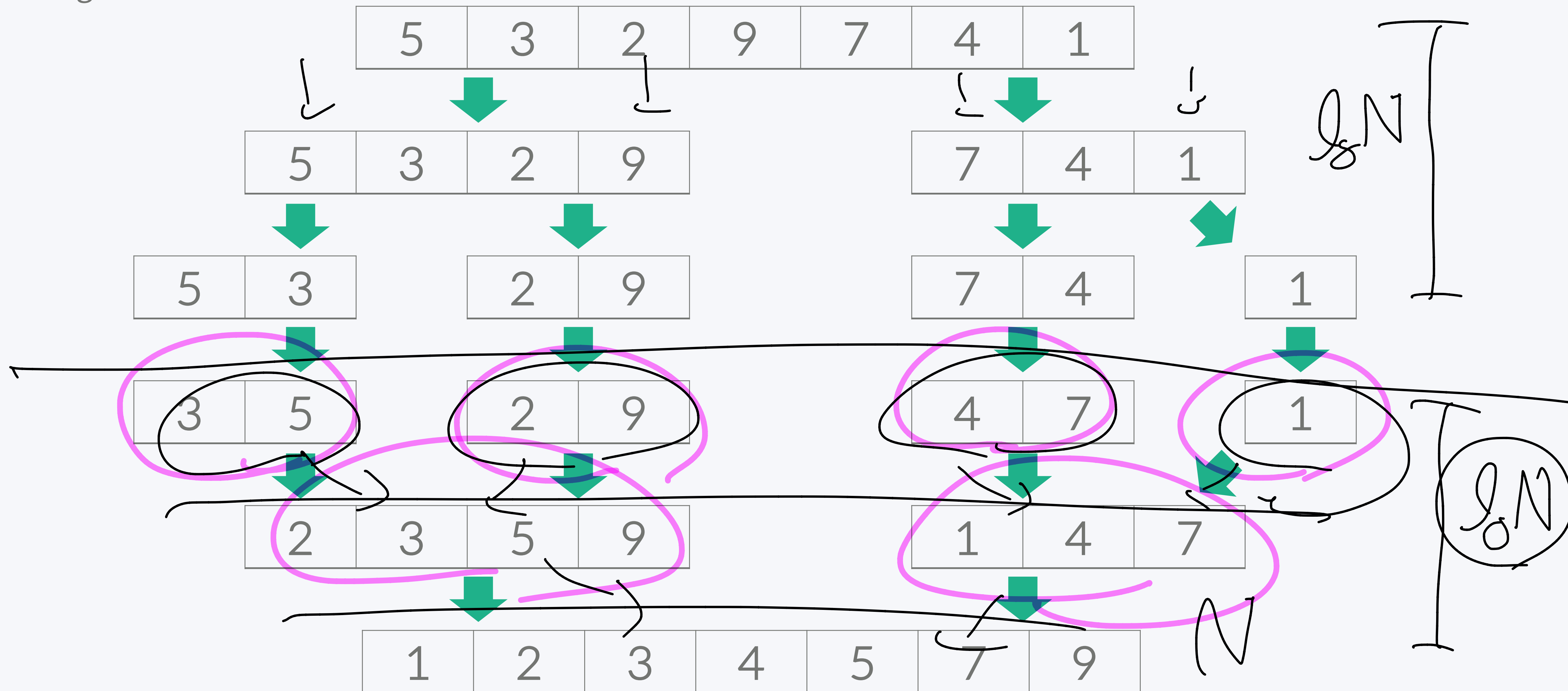


머지 소트

Merge Sort

$O(N \log N)$

22



머지 소트

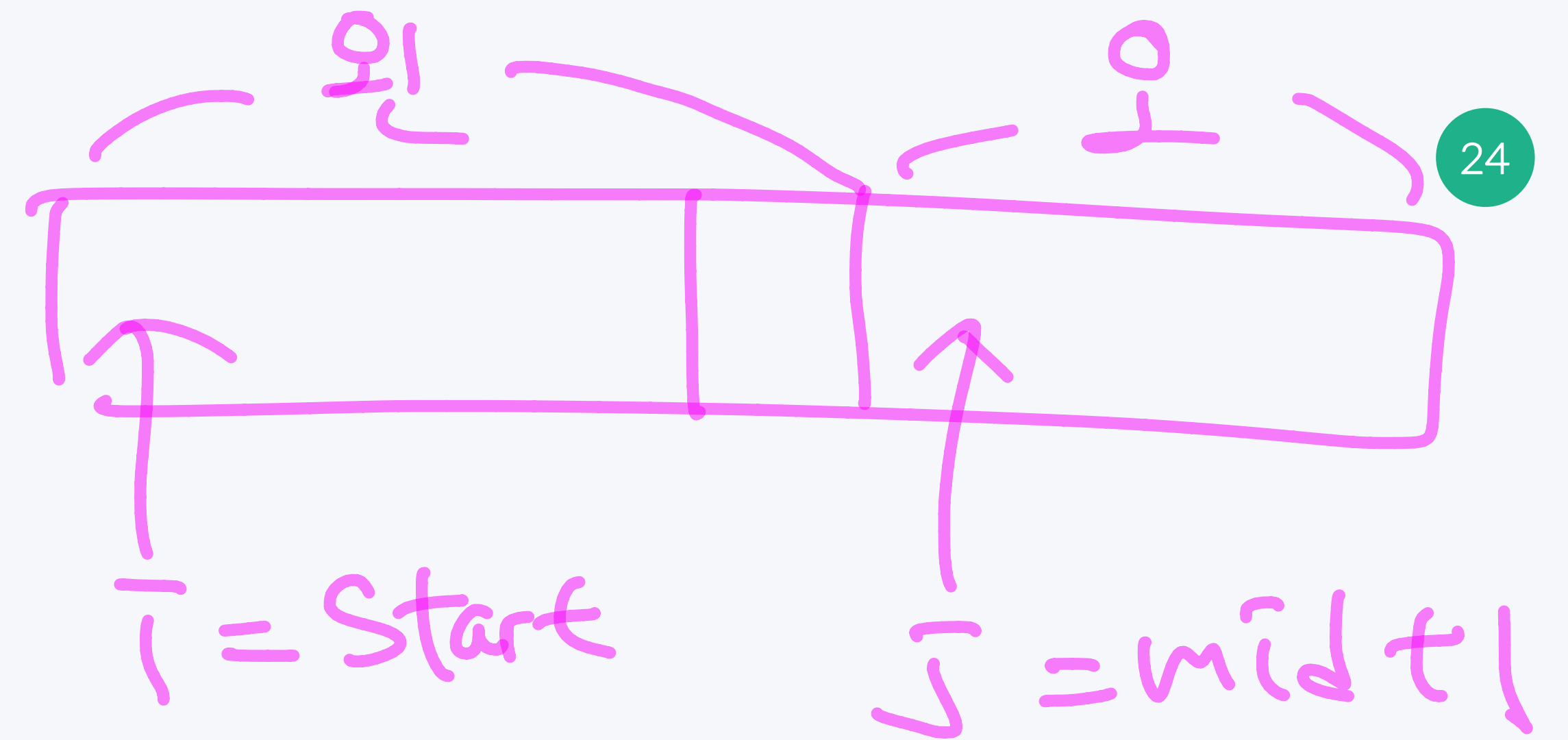
Merge Sort

```
void sort(int start, int end) {  
    if (start == end) {  
        return;  
    }  
    int mid = (start+end)/2;  
    sort(start, mid);  
    sort(mid+1, end);  
    merge(start, end);  
}
```

머지 소트

Merge Sort

```
void merge(int start, int end) {  
    int mid = (start+end)/2;  
    int i = start, j = mid+1, k = 0;  
    while (i <= mid && j <= end) {  
        if (a[i] <= a[j]) b[k++] = a[i++];  
        else b[k++] = a[j++];  
    }  
    while (i <= mid) b[k++] = a[i++];  
    while (j <= end) b[k++] = a[j++];  
    for (int i=start; i<=end; i++) {  
        a[i] = b[i-start];  
    }  
}
```



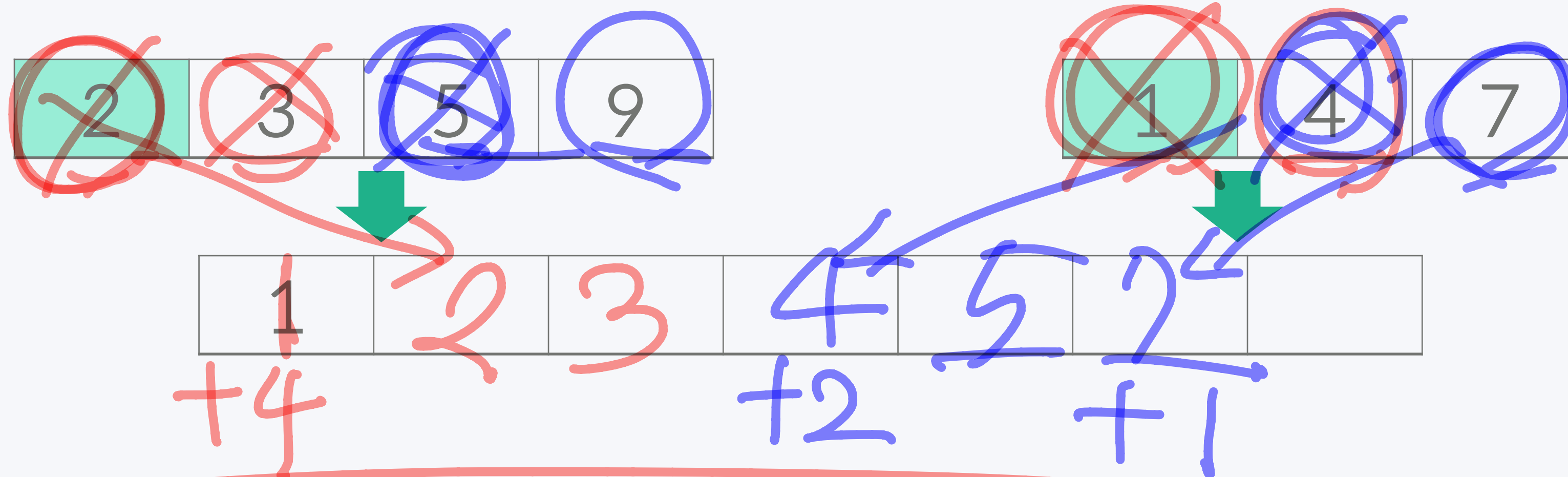
$O(N)$

배열 합치기

25

<https://www.acmicpc.net/problem/11728>

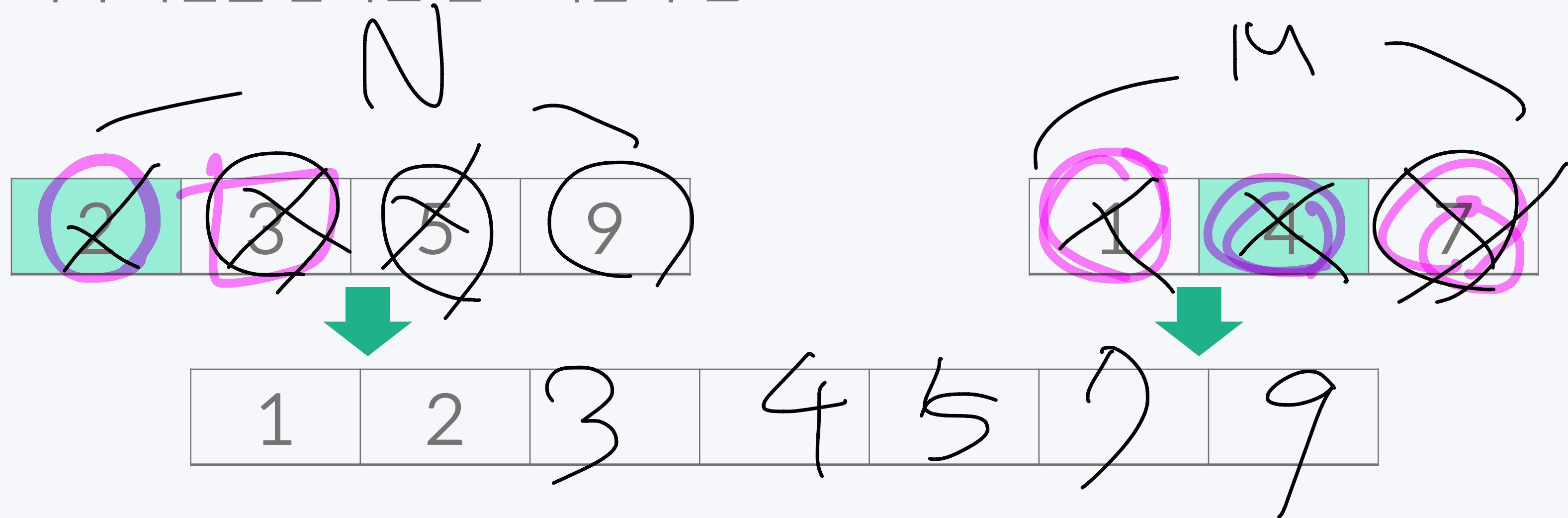
- 머지 소트에서 배열을 합치는 알고리즘 구현



배열 합치기

<https://www.acmicpc.net/problem/11728>

- 머지 소트에서 배열을 합치는 알고리즘 구현



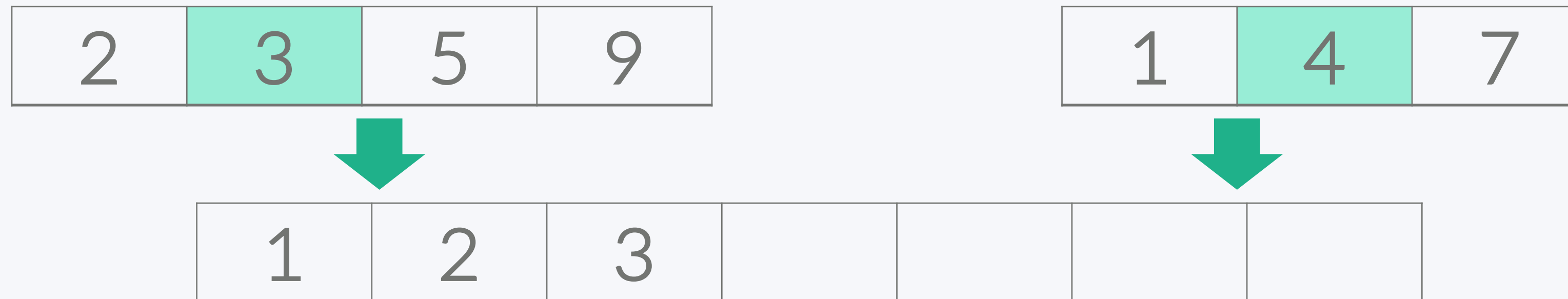
$$O(N+M)$$

배열 합치기

27

<https://www.acmicpc.net/problem/11728>

- 머지 소트에서 배열을 합치는 알고리즘 구현

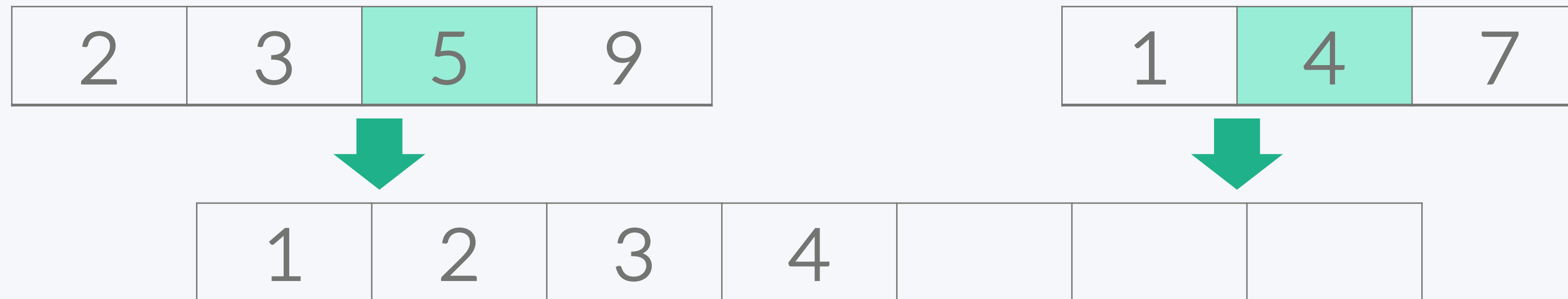


배열 합치기

28

<https://www.acmicpc.net/problem/11728>

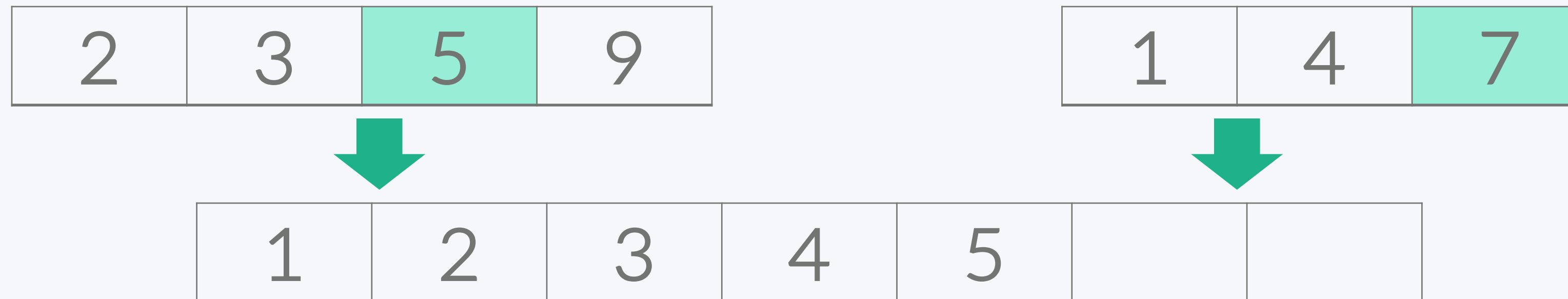
- 머지 소트에서 배열을 합치는 알고리즘 구현



배열 합치기

<https://www.acmicpc.net/problem/11728>

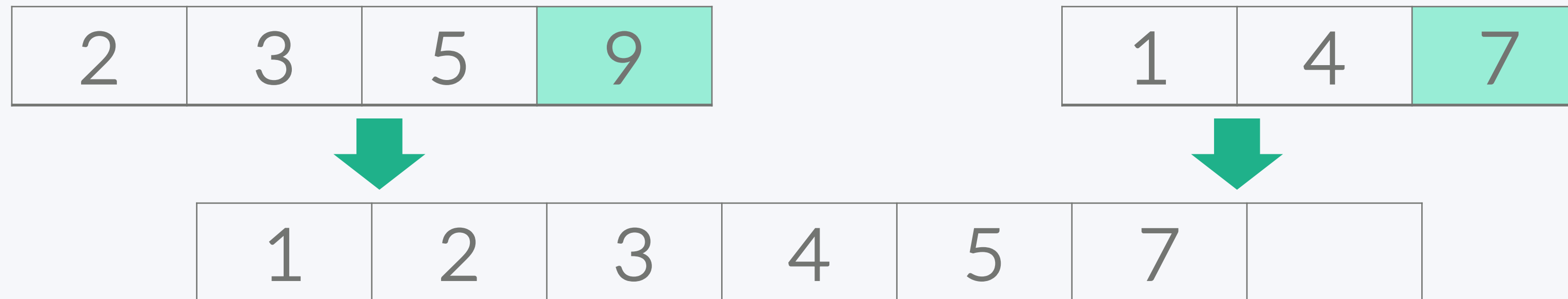
- 머지 소트에서 배열을 합치는 알고리즘 구현



배열 합치기

<https://www.acmicpc.net/problem/11728>

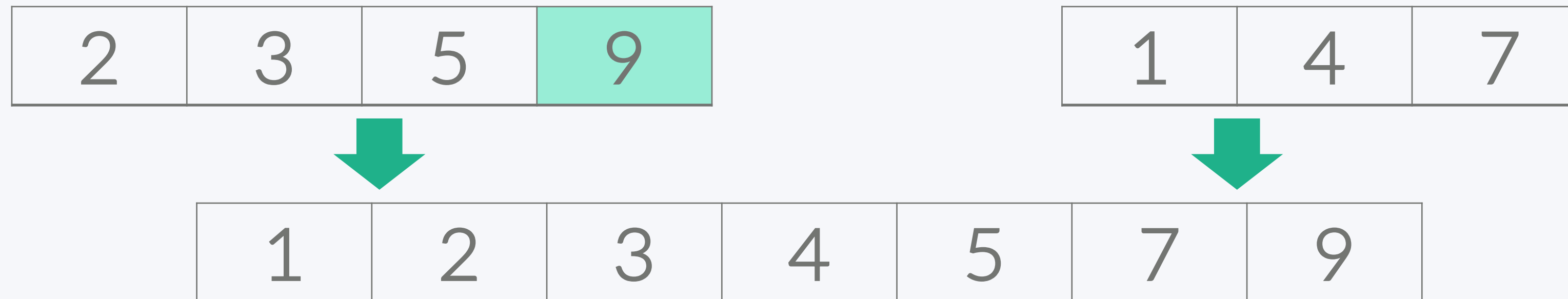
- 머지 소트에서 배열을 합치는 알고리즘 구현



배열 합치기

<https://www.acmicpc.net/problem/11728>

- 머지 소트에서 배열을 합치는 알고리즘 구현



배열 합치기

<https://www.acmicpc.net/problem/11728>

- 소스: <http://boj.kr/b1e44a9755fe4b2abcc437f54cc7edfe>

퀵 소트

퀵 소트

Quick Sort

- 평균적인 상황에서 최고의 성능을 자랑하는 알고리즘
- 피벗(pivot)을 하나 고른 다음, 그것보다 작은 것을 앞으로 큰 것을 뒤로 보낸다.
- 그 다음, 피벗의 앞과 뒤에서 퀵 정렬을 수행한다.
- 최악의 경우에는 $O(N^2)$ 이 걸린다.

퀵 소트

Quick Sort

- pivotValue = 5

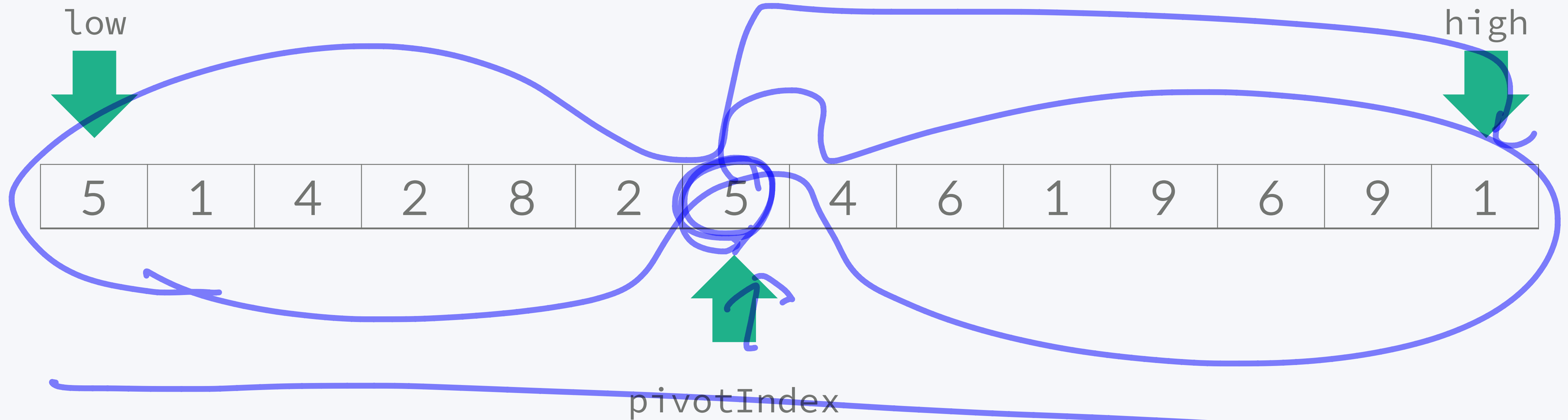
Handwritten notes showing the partitioning process:

Row 1: 1 2 3 4 5 6 7 8 9 (with 1 boxed)

Row 2: 2 3 4 5 6 7 8 9 (with 2 boxed)

Row 3: 3 4 5 6 7 8 9

35



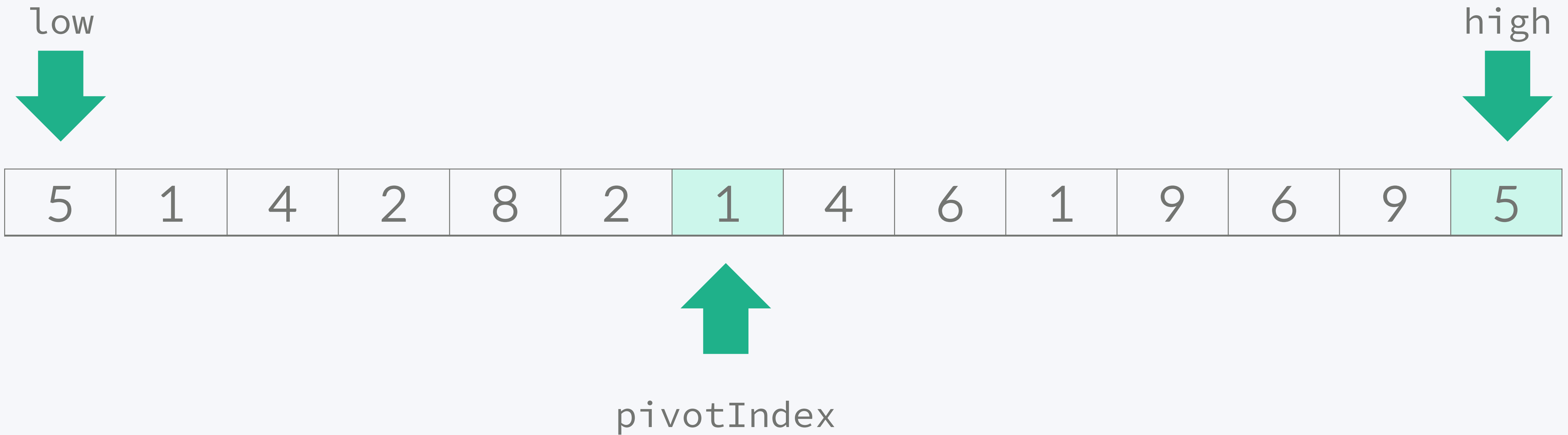
$O(N)$

퀵 소트

Quick Sort

36

- pivotValue = 5

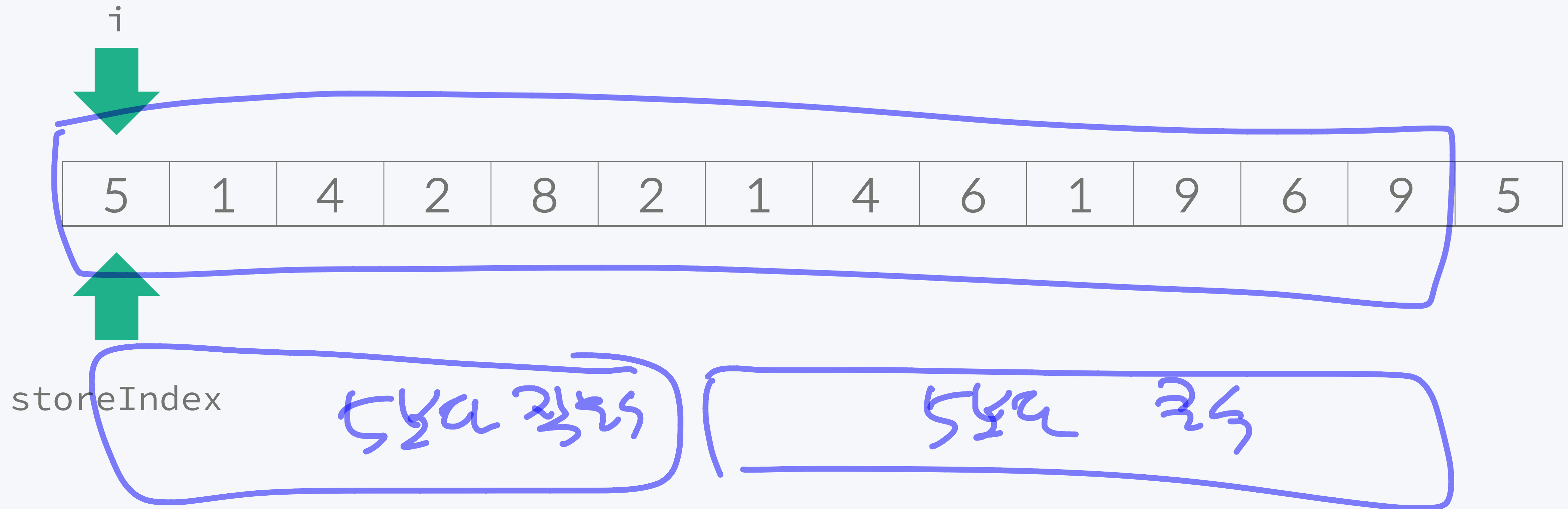


퀵 소트

37

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

38

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

i



5	1	4	2	8	2	1	4	6	1	9	6	9	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---



storeIndex

퀵 소트

39

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

i



1	5	4	2	8	2	1	4	6	1	9	6	9	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---



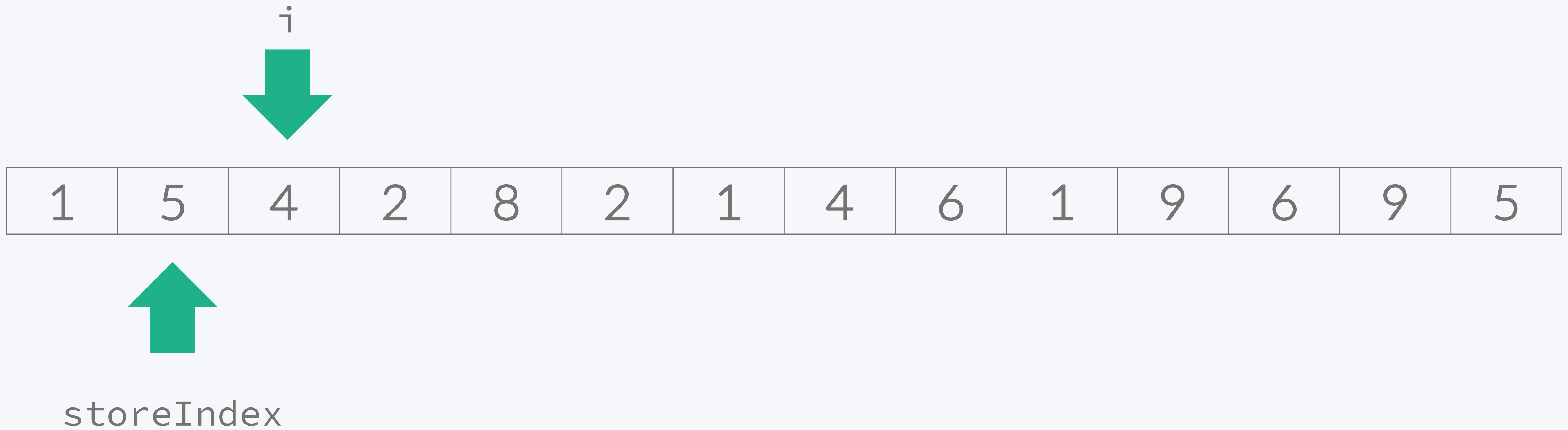
storeIndex

퀵 소트

40

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

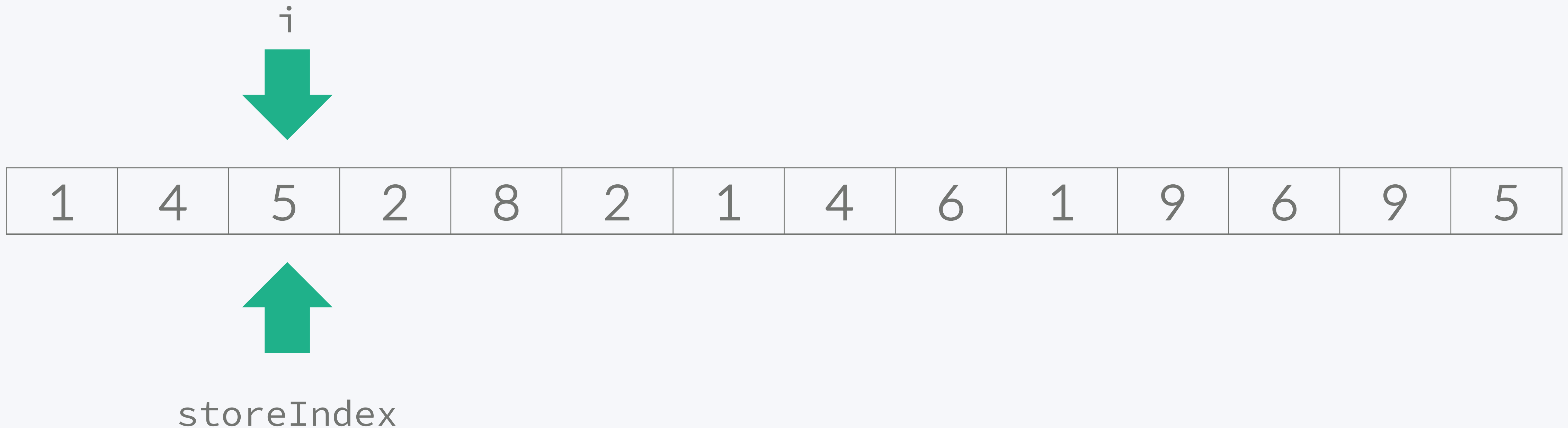


퀵 소트

41

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

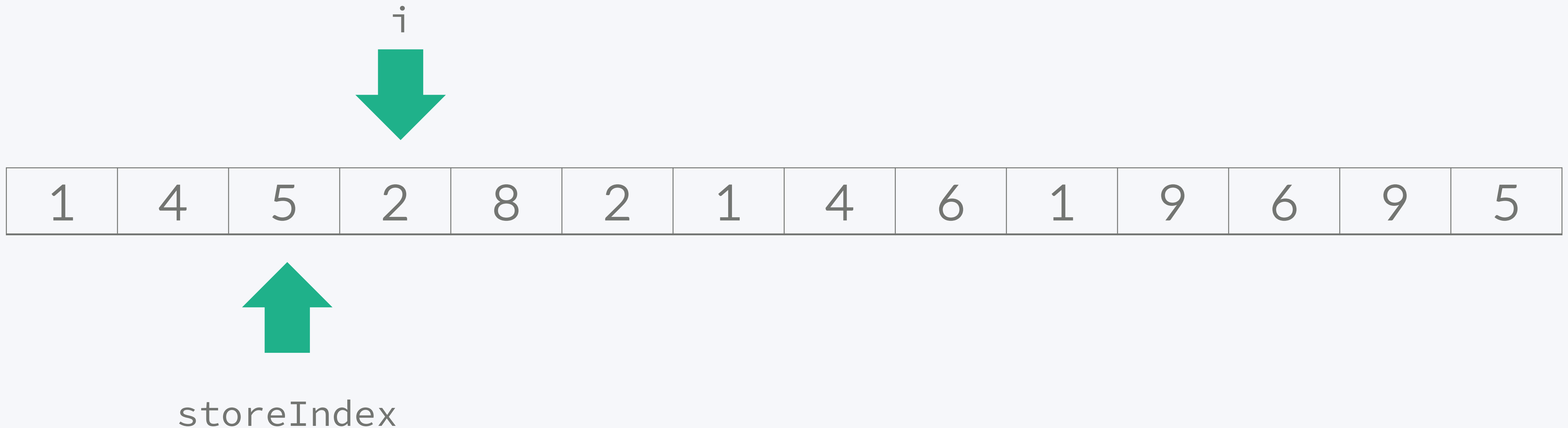


퀵 소트

42

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

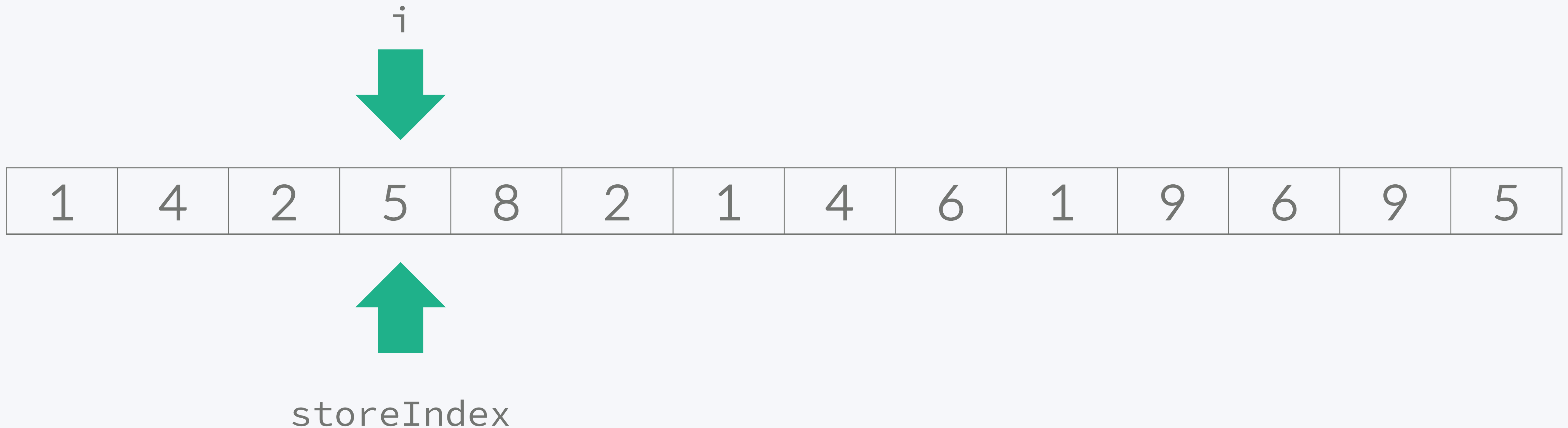


퀵 소트

43

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

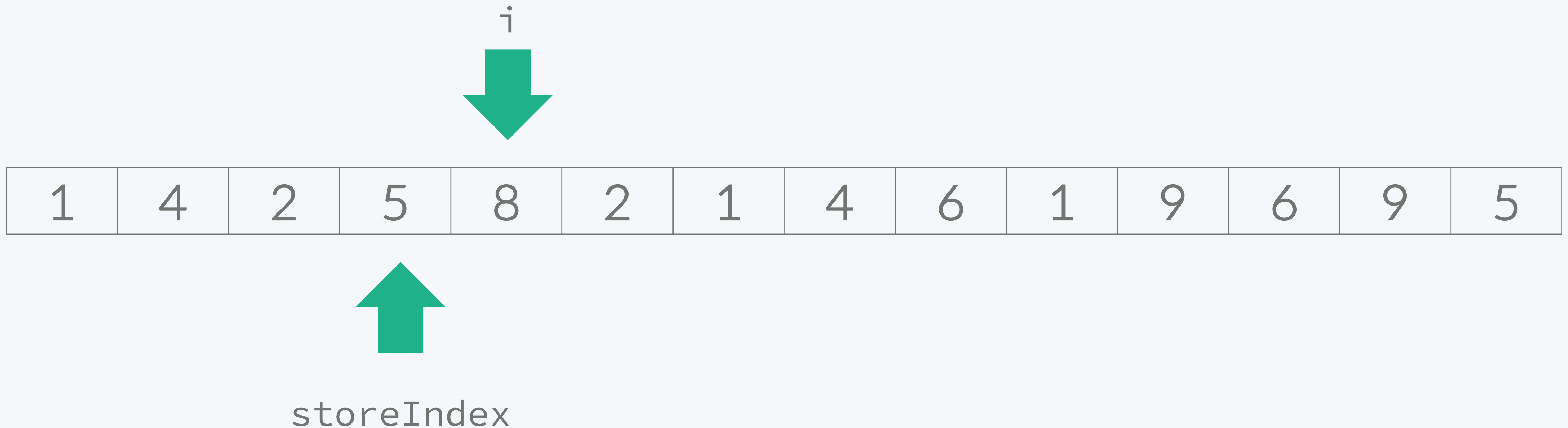


퀵 소트

44

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

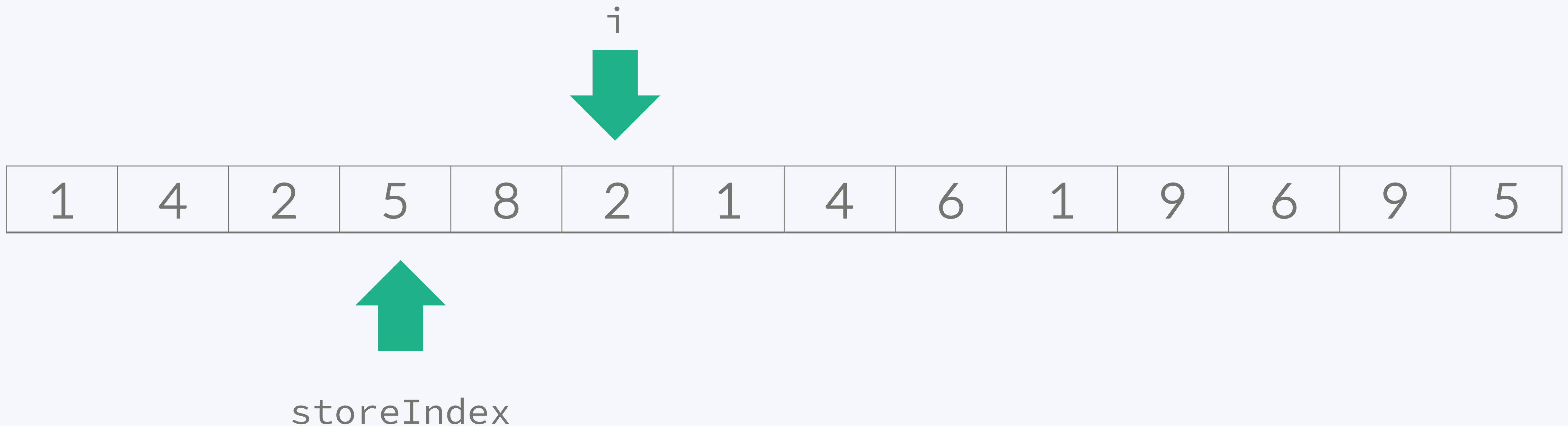


퀵 소트

45

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

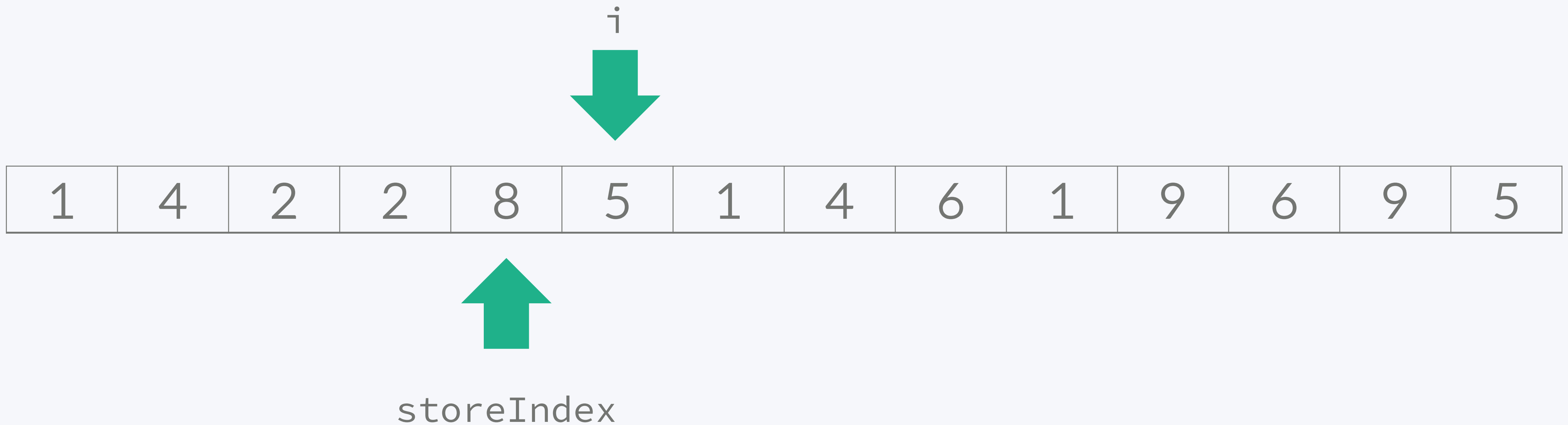


퀵 소트

46

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

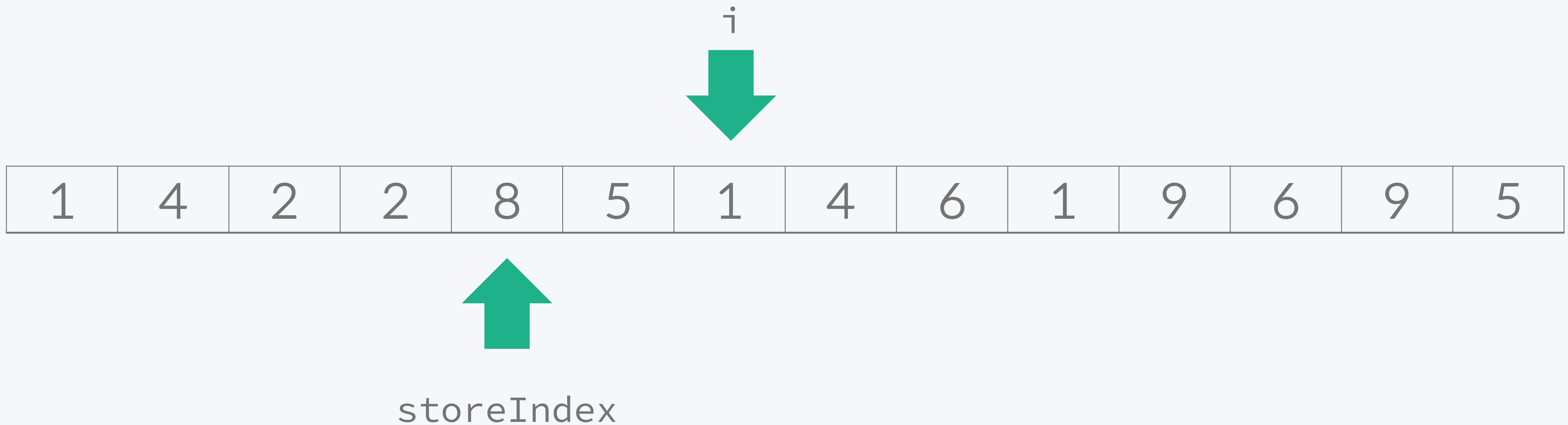


퀵 소트

47

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

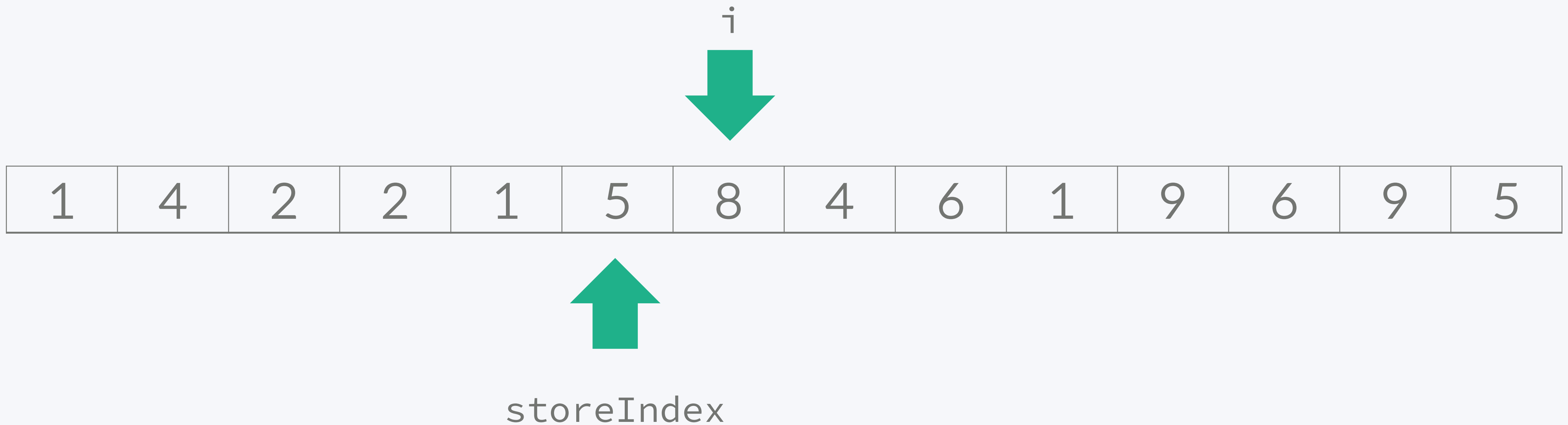


퀵 소트

48

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

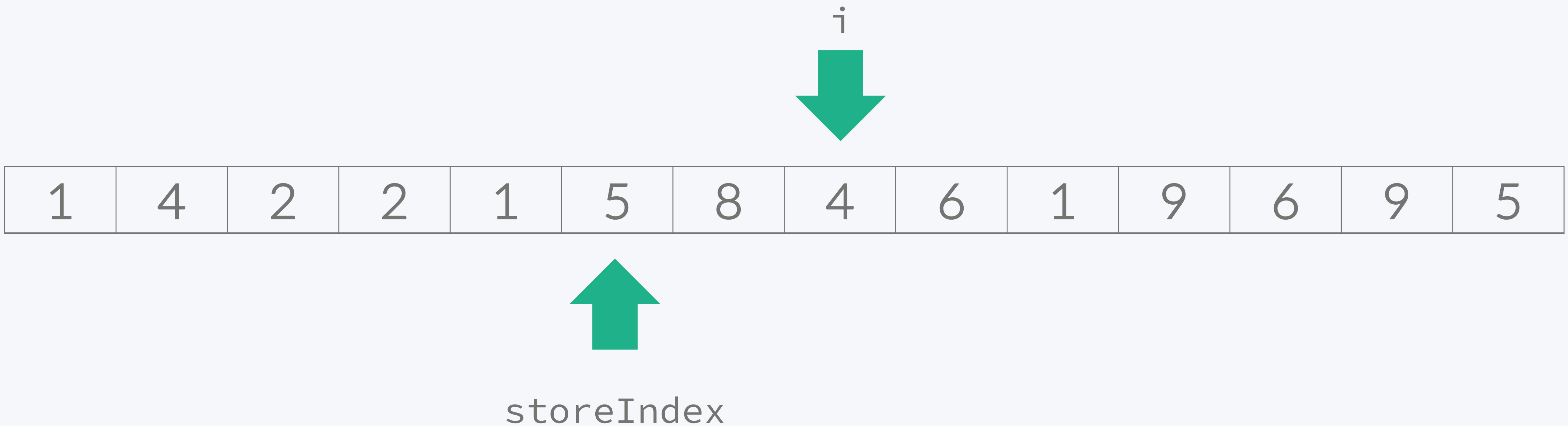


퀵 소트

49

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

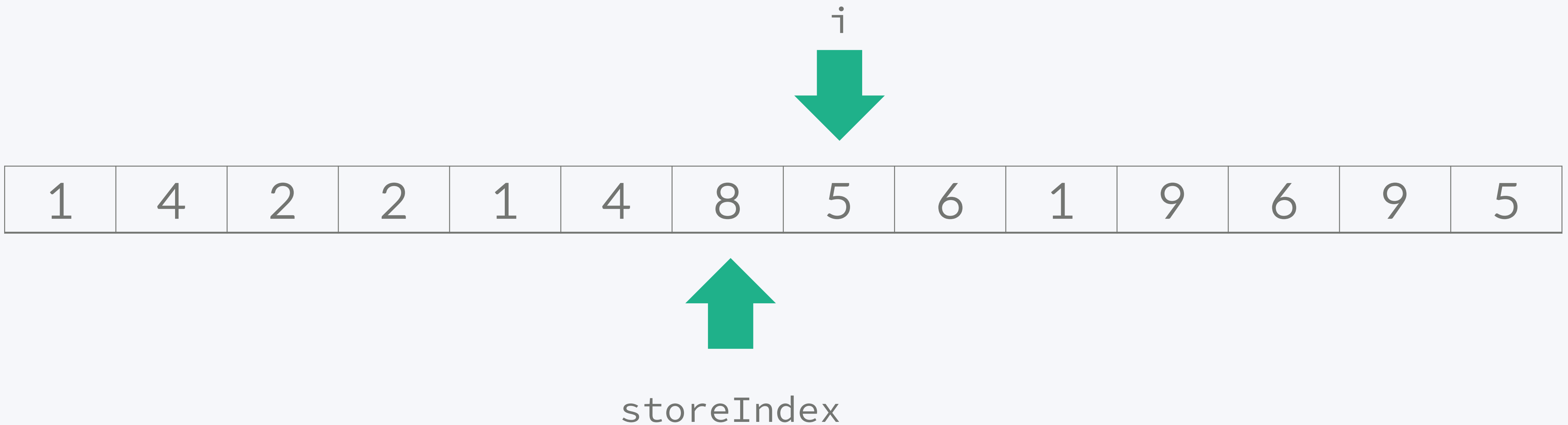


퀵 소트

50

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

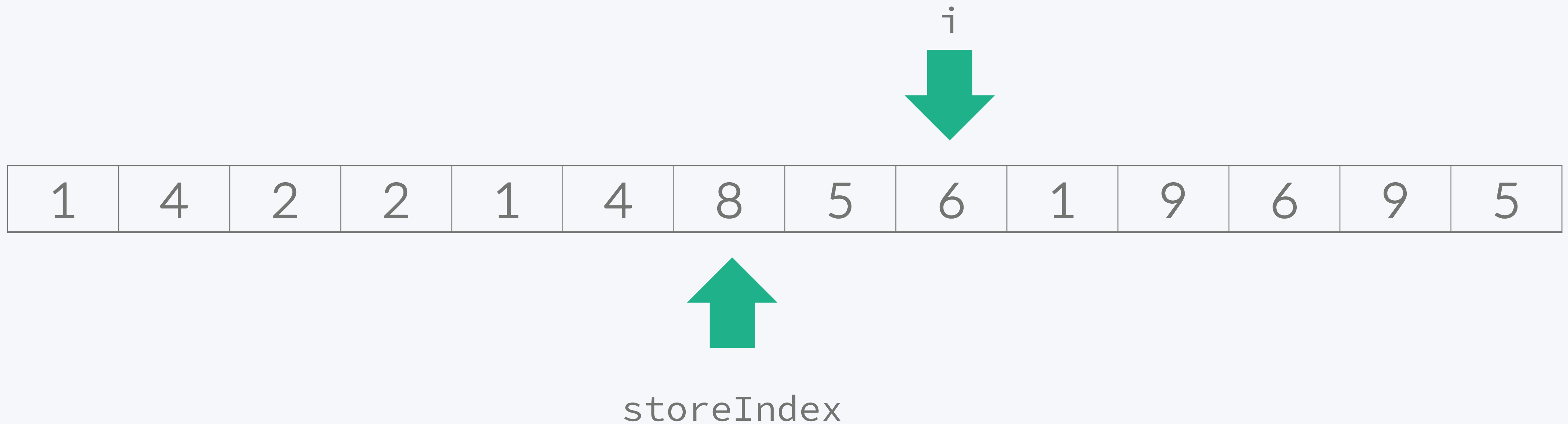


퀵 소트

51

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

52

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

53

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

54

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

55

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

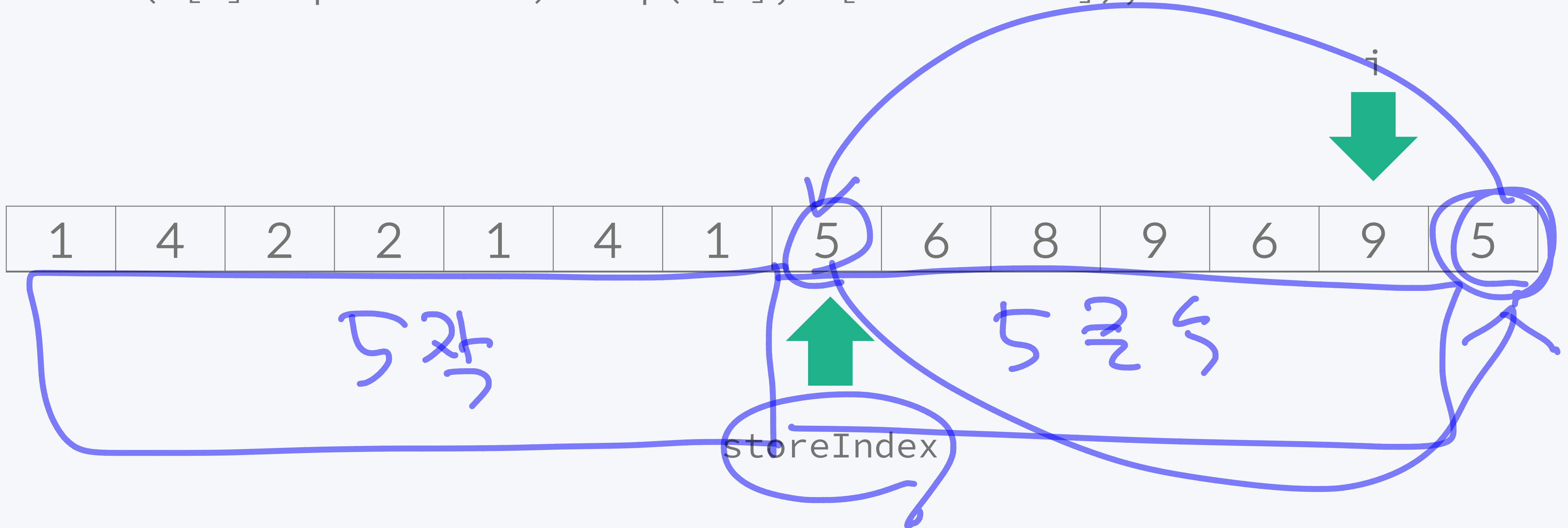


퀵 소트

56

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

57

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

58

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`

1	4	2	2	1	4	1	5	6	8	9	6	9	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---



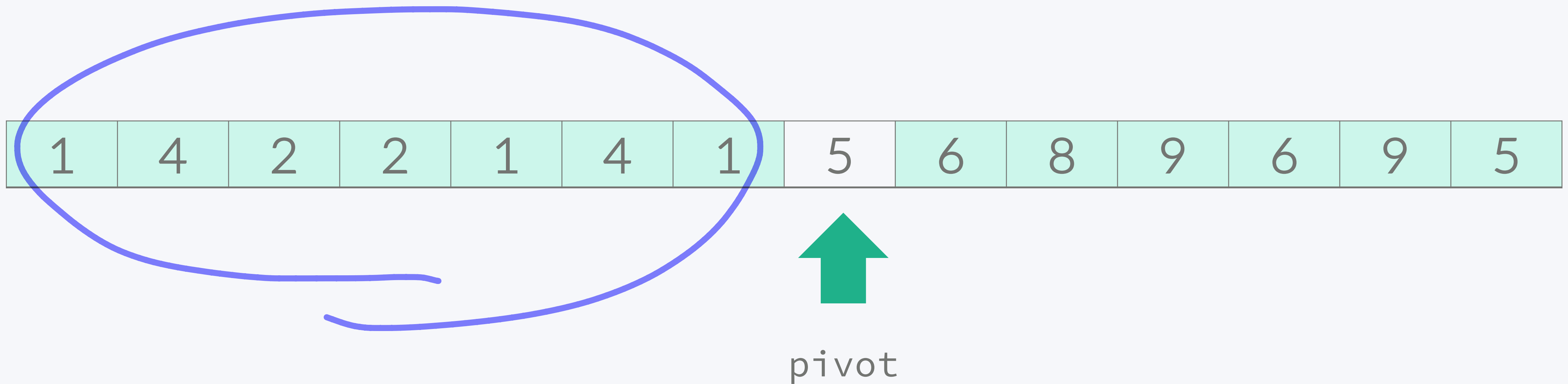
pivot

퀵 소트

59

Quick Sort

- `pivotValue = 5`
- `if (a[i] < pivotValue) swap(a[i], a[storeIndex]), storeIndex += 1`



퀵 소트

Quick Sort

$$\frac{lo+hi}{2}$$

```
int choosePivot(int low, int high) {  
    return low + (high-low)/2;  
}
```

퀵 소트

Quick Sort

```
int partition(int low, int high) {  
    int pivotIndex = choosePivot(low, high);  
    int pivotValue = a[pivotIndex];  
    swap(a[pivotIndex], a[high]);  
    int storeIndex = low;  
    for (int i=low; i<high; i++) {  
        if (a[i] < pivotValue) {  
            swap(a[i], a[storeIndex]);  
            storeIndex += 1;  
        }  
    }  
    swap(a[storeIndex], a[high]);  
    return storeIndex;  
}
```

퀵 소트

Quick Sort

```
void quicksort(int low, int high) {  
    if (low < high) {  
        int pivot = partition(low, high);  
        quicksort(low, pivot-1);  
        quicksort(pivot+1, high);  
    }  
}
```

Intro Sort

I ~~Heap Sort~~

문제 풀기

분할 정복

Divide & Conquer

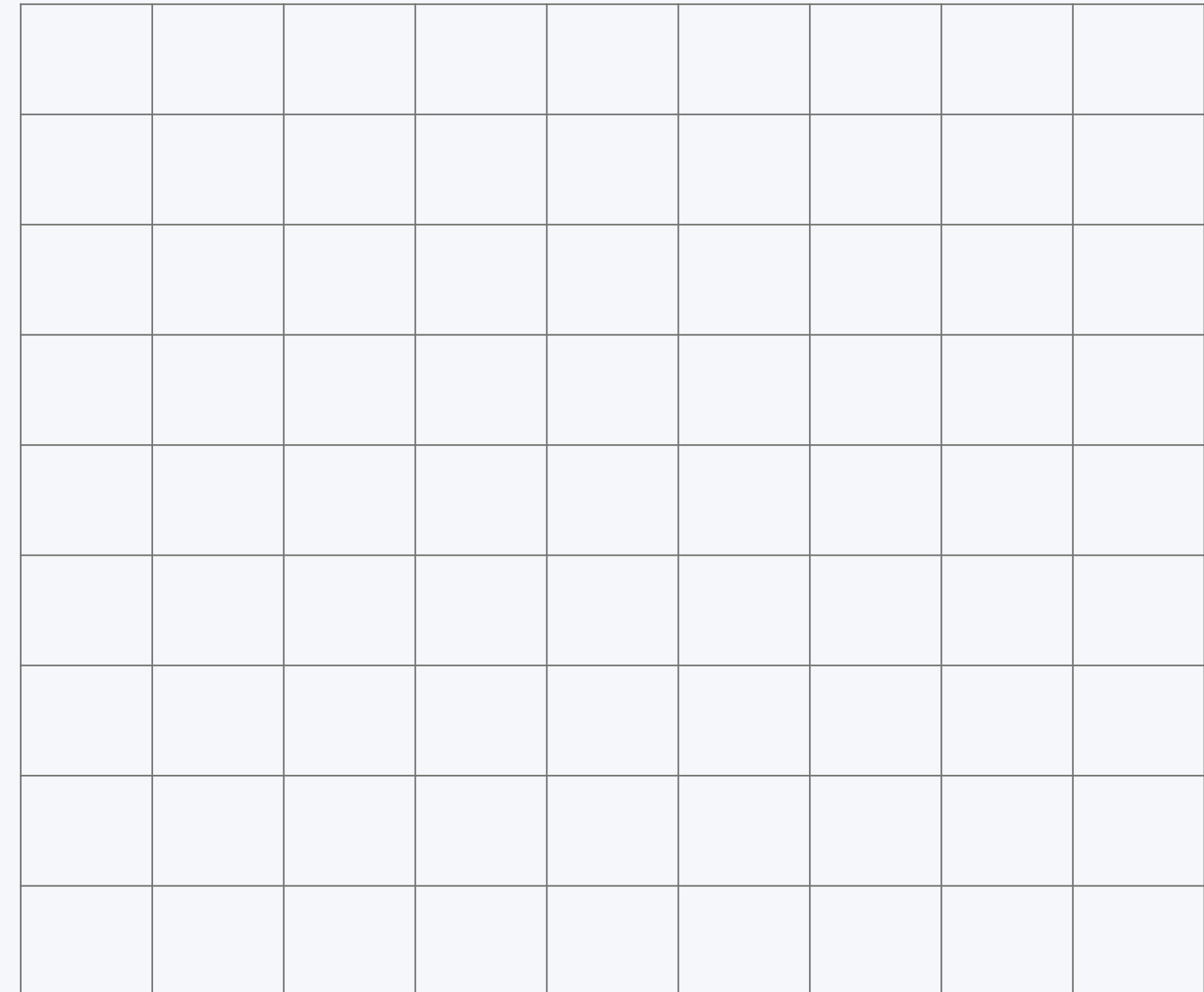
64

- 분할 정복 문제는 어떻게 함수를 만들어야 할지 결정해야 한다.
- 함수 -> 문제를 푸는 함수
- 그 함수 내에서는 작은 문제를 어떻게 호출해야 할지 결정
- 만약에 부분 문제의 정답을 합쳐야 하는 경우에는 합치는 것을 어떻게 해야 할지 결정

종이의 개수

<https://www.acmicpc.net/problem/1780>

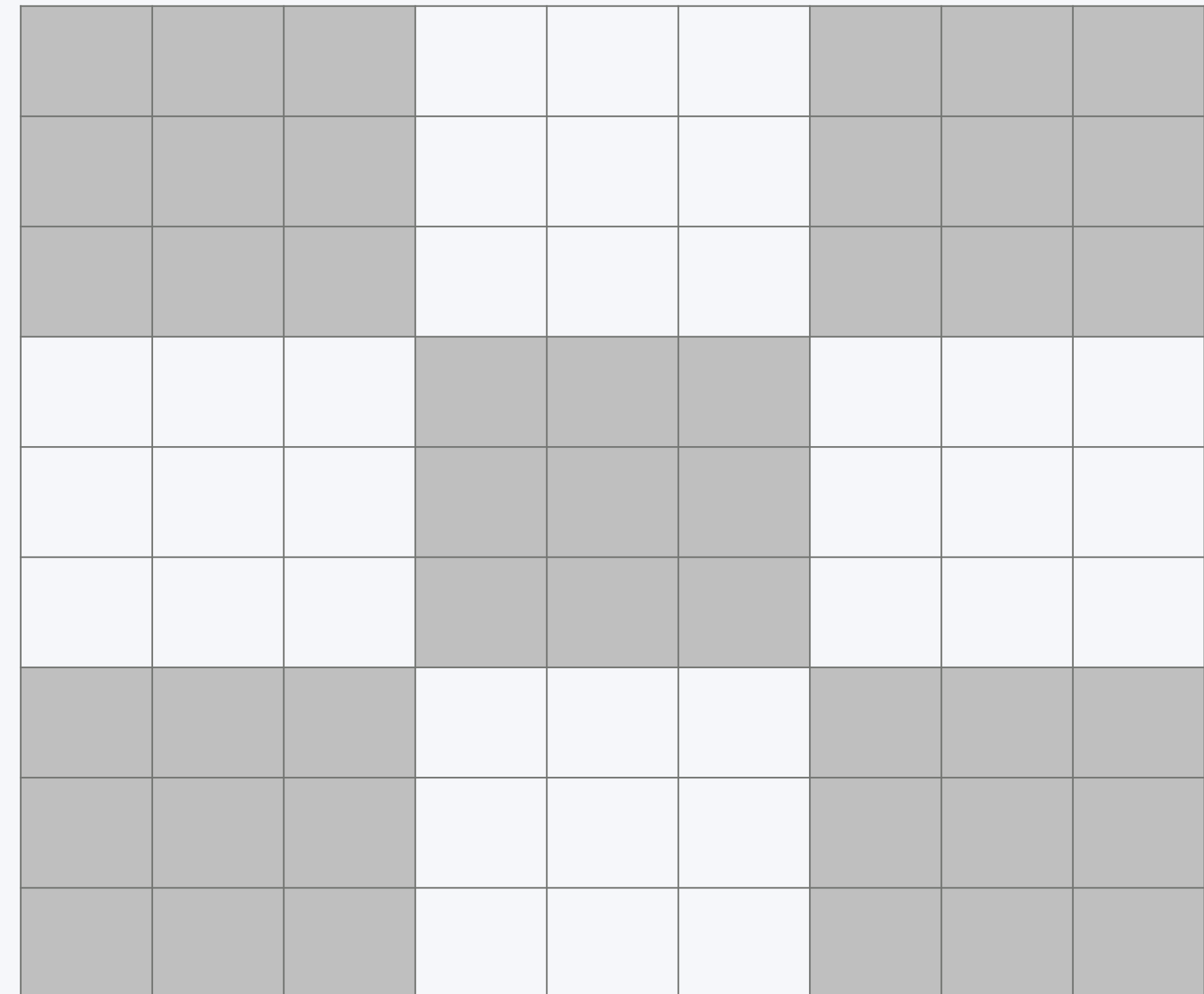
- `solve(0, 0, n)`
- $(0, 0)$ 부터 가로로 n 개, 세로로 n 개의 종이 개수를 확인하는 함수



종이의 개수

<https://www.acmicpc.net/problem/1780>

- $\text{solve}(x, y, n)$
- (x, y) 부터 가로로 n 개, 세로로 n 개의 종이 개수를 확인하는 함수
- 작은 부분 문제는 총 9개
- $m = n/3$ 이라고 했을 때
- 부분 문제를 나눠보면



종이의 개수

<https://www.acmicpc.net/problem/1780>

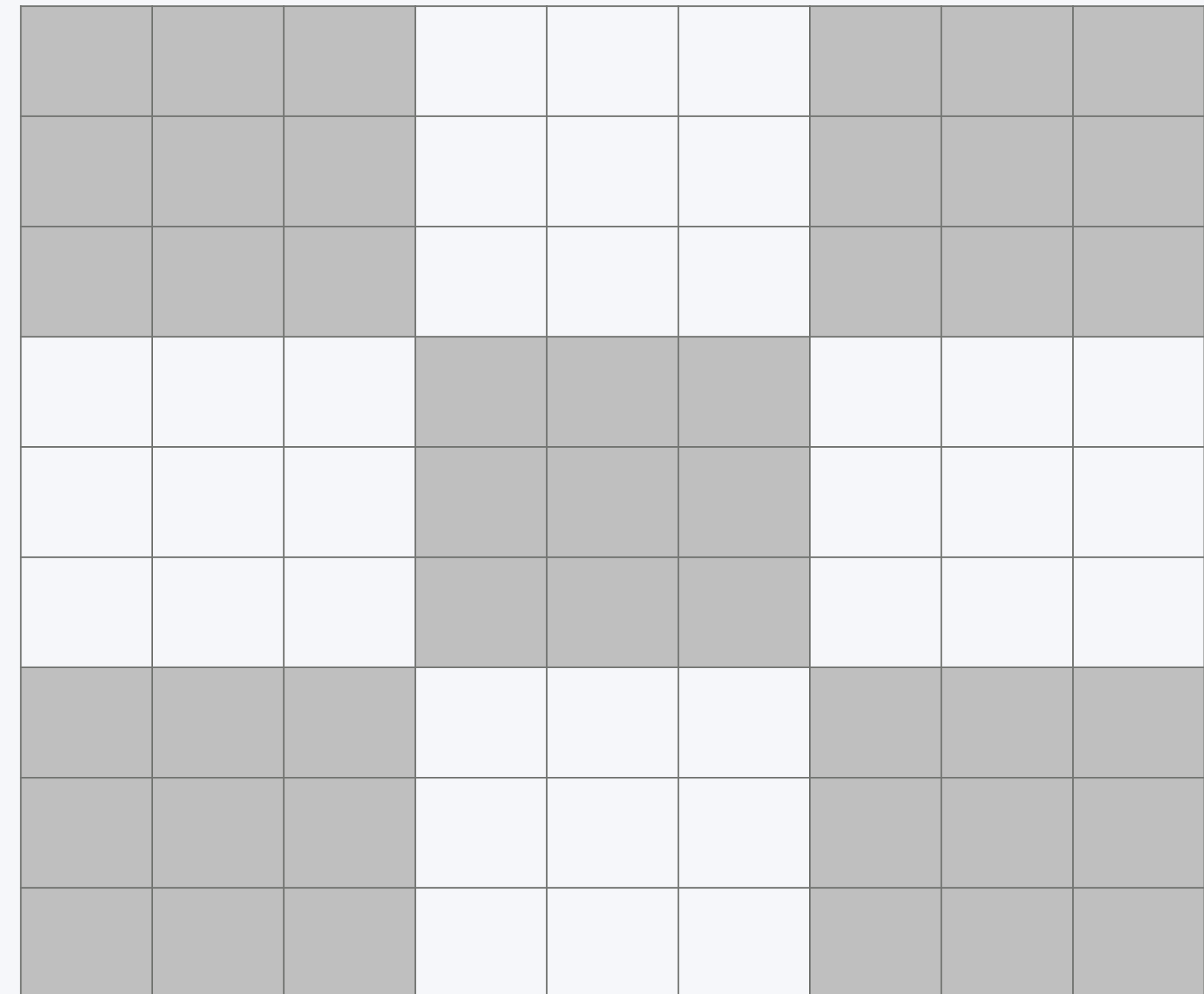
- $\text{solve}(x, y, n)$
 - 0: $\text{solve}(x, y, m)$
 - 1: $\text{solve}(x, y+m, m)$
 - 2: $\text{solve}(x, y+2*m, m)$
 - 3: $\text{solve}(x+m, y, m)$
 - 4: $\text{solve}(x+m, y+m, m)$
 - 5: $\text{solve}(x+m, y+2*m, m)$
 - 6: $\text{solve}(x+2*m, y, m)$
 - 7: $\text{solve}(x+2*m, y+m, m)$
 - 8: $\text{solve}(x+2*m, y+2*m, m)$

0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8

종이의 개수

<https://www.acmicpc.net/problem/1780>

- $\text{solve}(x, y, n)$
- (x, y) 부터 가로로 n 개, 세로로 n 개의 종이 개수를 확인하는 함수
- 부분 문제를 호출하기 전에
- 같은 수로 되어 있는지를 확인해야 한다
- 그게 아니면 부분 문제를 호출



종이의 개수

<https://www.acmicpc.net/problem/1780>

```
void solve(int x, int y, int n) {
    if (same(x, y, n)) {
        cnt[a[x][y]+1] += 1;
        return;
    }
    int m = n/3;
    for (int i=0; i<3; i++) {
        for (int j=0; j<3; j++) {
            solve(x+i*m, y+j*m, m);
        }
    }
}
```

종이의 개수

<https://www.acmicpc.net/problem/1780>

```
bool same(int x, int y, int n) {  
    for (int i=x; i<x+n; i++) {  
        for (int j=y; j<y+n; j++) {  
            if (a[x][y] != a[i][j]) {  
                return false;  
            }  
        }  
    }  
    return true;  
}
```

종이의 개수

71

<https://www.acmicpc.net/problem/1780>

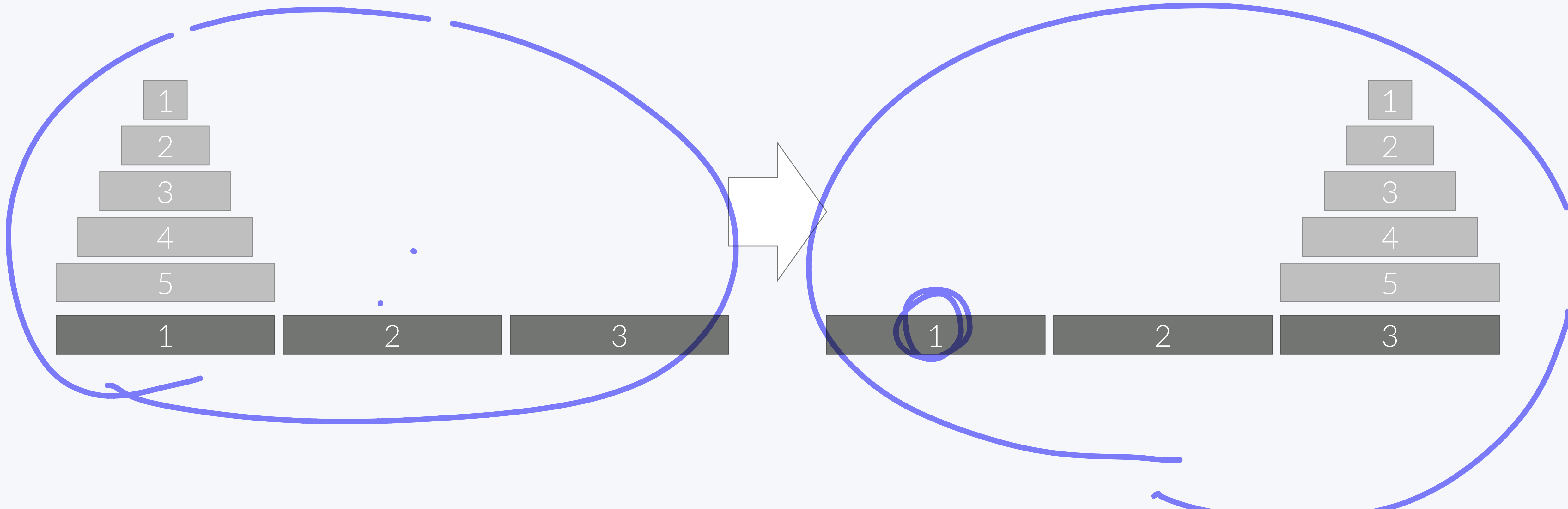
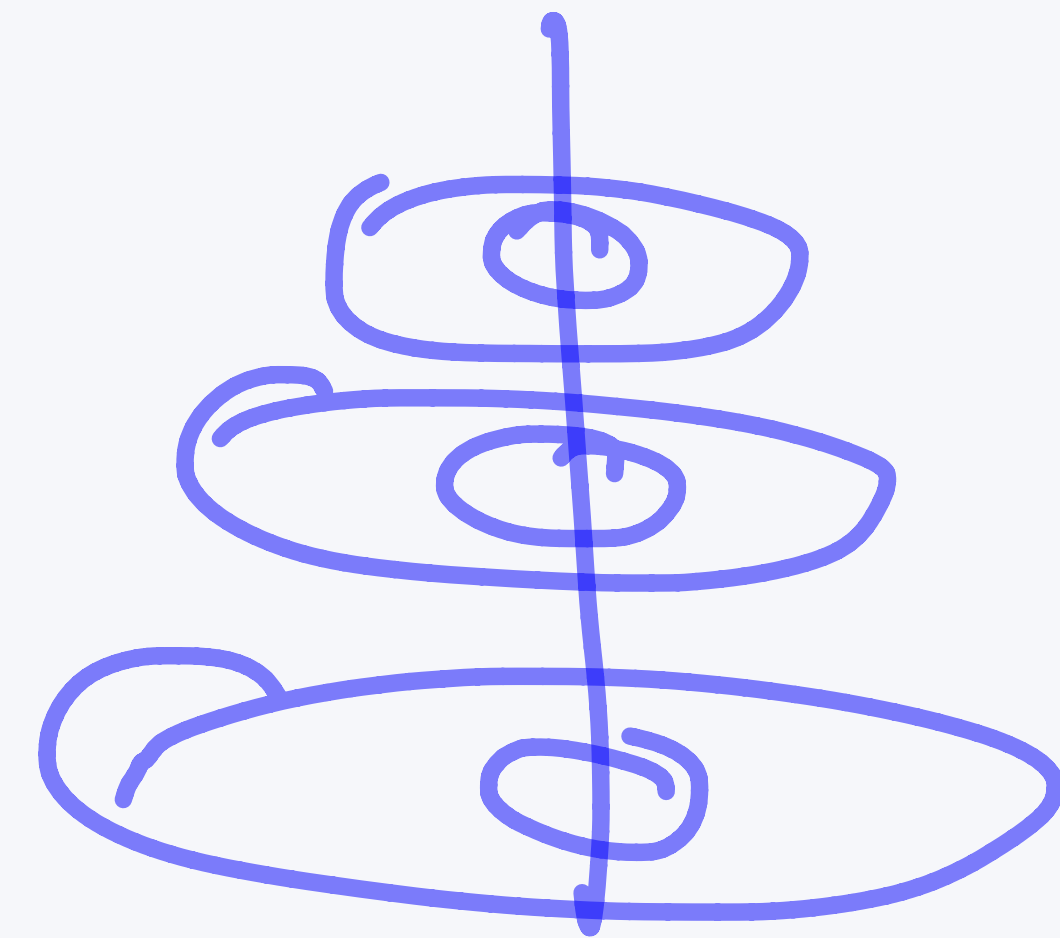
- 소스: <http://boj.kr/e7b36fc1604b4df2af42b8cd63f3cb59>

하노이 탑 이동 순서 $1 \sim n$

72

<https://www.acmicpc.net/problem/11729>

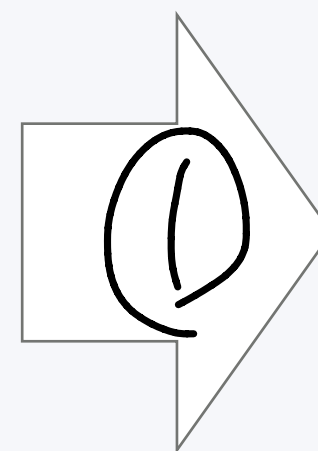
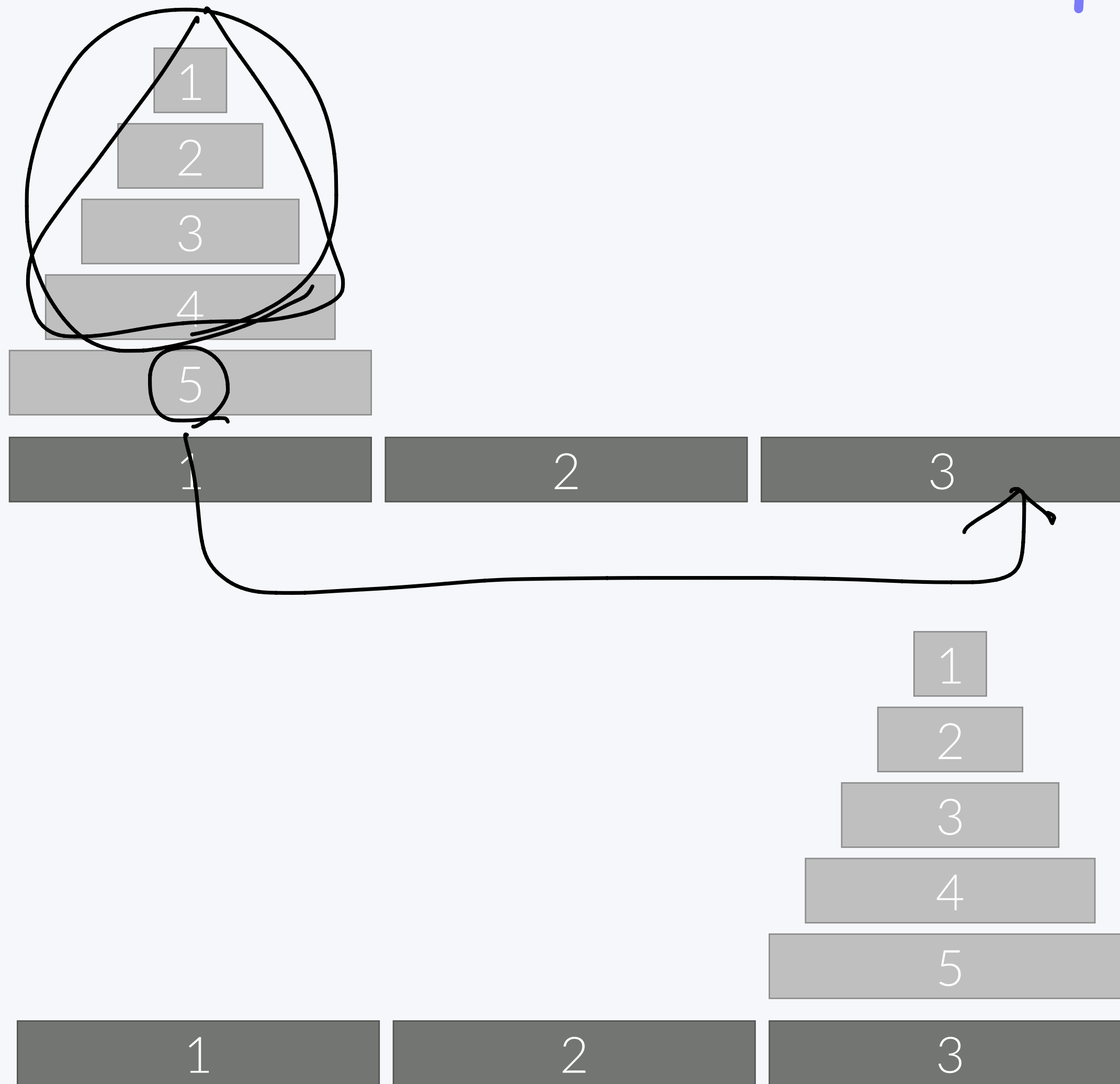
- 하노이 탑 문제
- 규칙 1: 한 번에 한 개의 원판만 다른 탑으로 옮길 수 있다
- 규칙 2: 쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다 (중간 과정 포함)



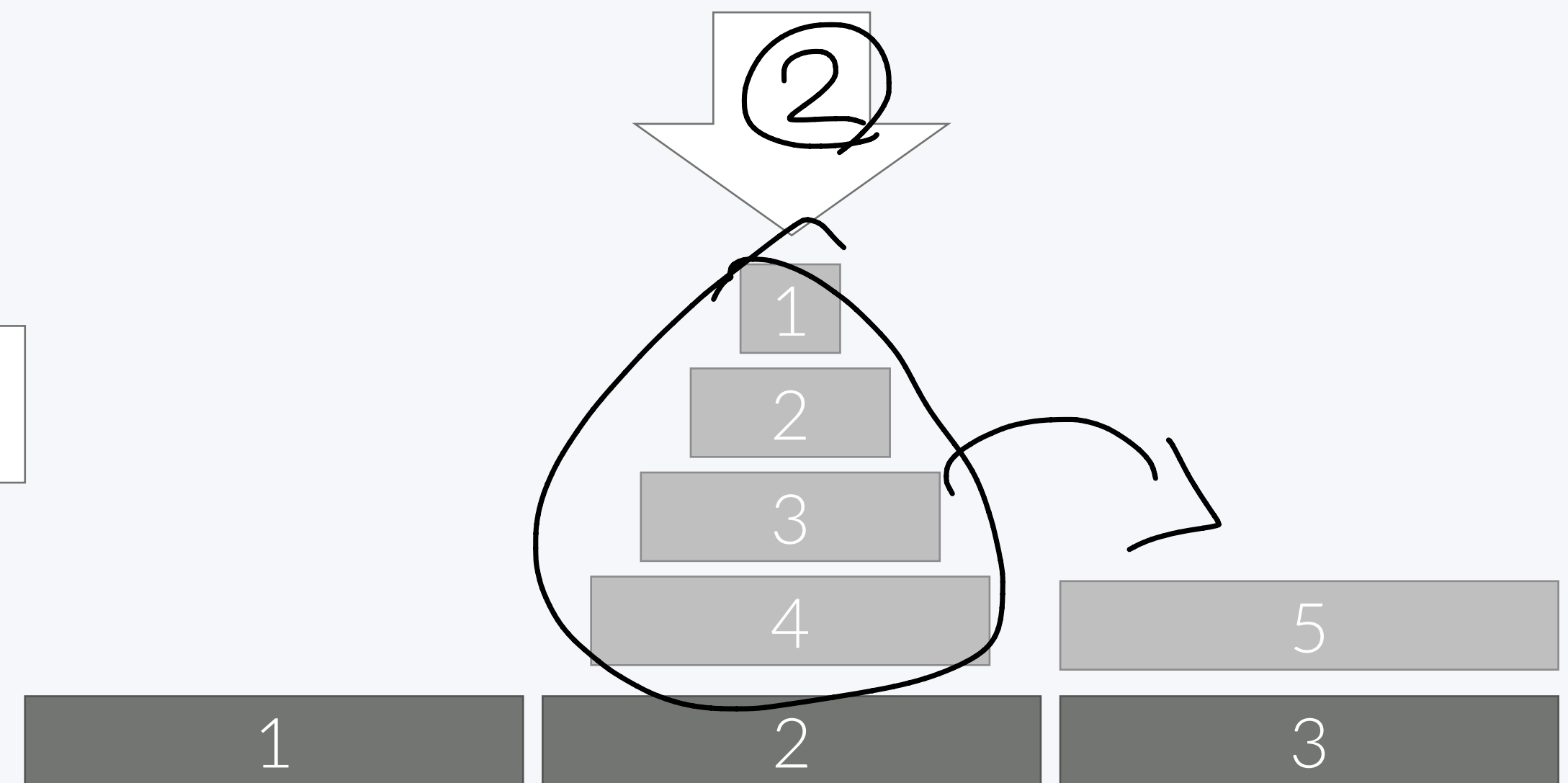
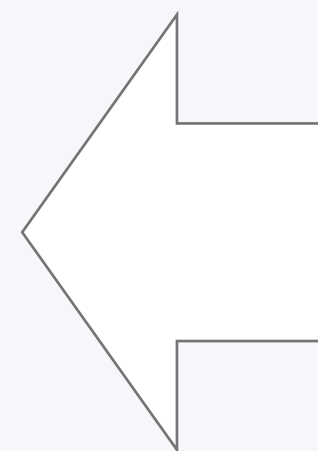
하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>

73



1~5번 원판 1번 => 3번
 ① 1~4번 원판 1번 => 2번
 ② 5번 원판 1번 => 3번
 ③ 1~4번 원판 2번 => 3번
 ④ 1~3번 원판 2번 => 1번
 ⑤ 4번 원판 2번 => 3번
 ⑥ 1~3번 원판 1번 => 3번



하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>

- solve(n, x, y)
- 1~n개의 원판을 x에서 y로 이동하는 함수

$\text{solve}(n-1, x, z)$

- 1~n-1개의 원판을 x에서 z로 이동한다. (z는 x와 y가 아닌 원판)
- n번 원판을 x에서 y로 이동한다. $\text{solve}(n-1, z, y)$
- 1~n-1개의 원판을 z에서 y로 이동한다.

하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>

- $\text{solve}(n, x, y)$
- 1~n개의 원판을 x에서 y로 이동하는 함수
- 1~n-1개의 원판을 x에서 z로 이동한다. (z는 x와 y가 아닌 원판)
 - $\text{solve}(n-1, x, z)$
- n번 원판을 x에서 y로 이동한다.
- 1~n-1개의 원판을 z에서 y로 이동한다.
 - $\text{solve}(n-1, z, y)$

하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>

```
void solve(int n, int x, int y) {  
    if (n == 0) return;  
    → solve(n-1, x, 6-x-y);  
    printf("%d %d\n", x, y);  
    → solve(n-1, 6-x-y, y);  
}
```

$DT[i] = (2^i - 1) \times 2$ 이동 횟수

$DT[1] = 1$

$$DT[i] = 2 \times DT[i-1] + 1$$
$$\rightarrow = 2^i - 1$$

$$\begin{aligned} \underline{DT[i] + 1} &= 2 DT[i-1] + 2 \\ &= \underline{2(DT[i-1] + 1)} \end{aligned}$$

하노이 탑 이동 순서

77

<https://www.acmicpc.net/problem/11729>

- 소스: <http://boj.kr/612a758702e9421c888efcb701c57040>

트리의 순회

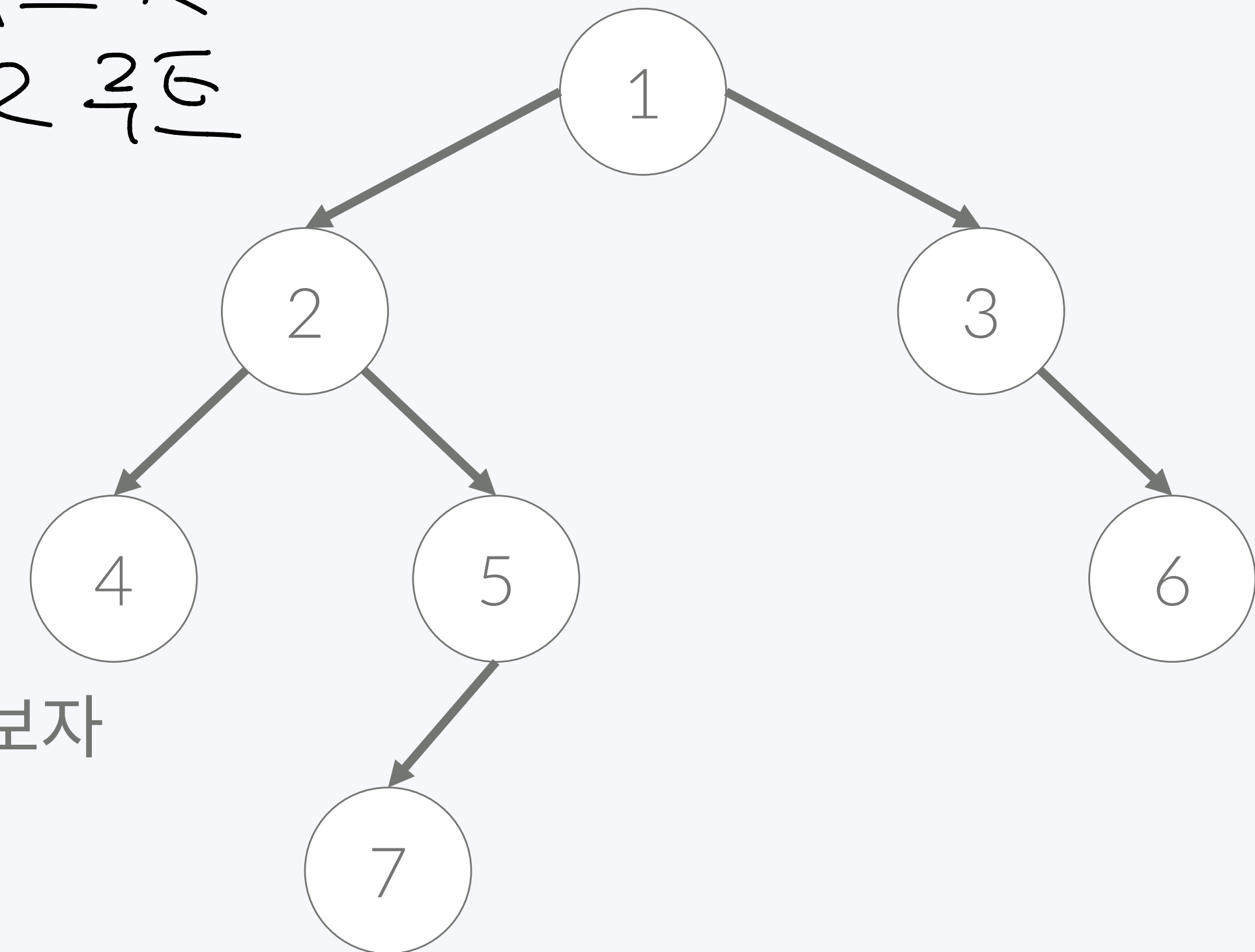
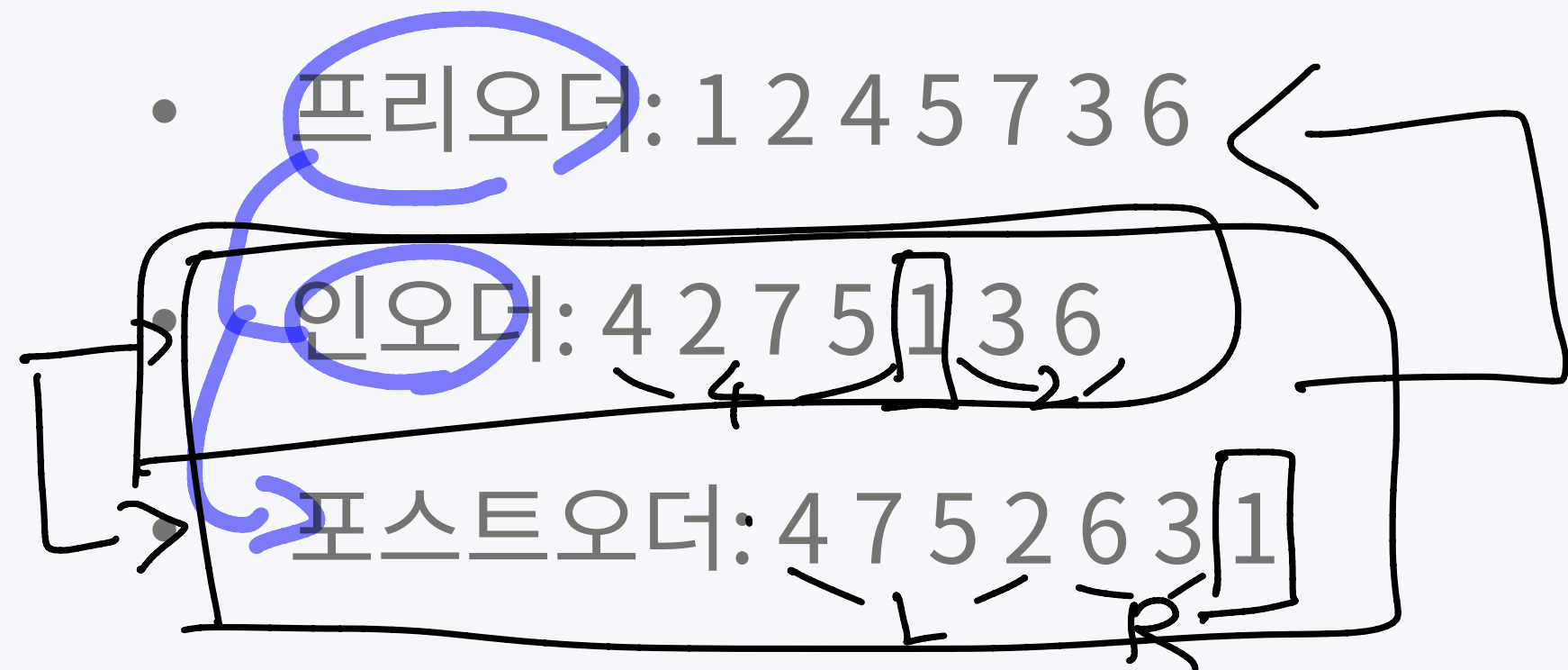
<https://www.acmicpc.net/problem/2263>

- N개의 정점을 갖는 이진 트리의 정점에 1부터 N까지 번호가 중복없이 매겨져 있다
- 이 이진 트리의 인오더와 포스트오더가 주어졌을 때 프리오더를 구하는 문제

트리의 순회

<https://www.acmicpc.net/problem/2263>

프리: 루트 LR
인: L 루트 R
포스트: LR 루트



- 인오더와 포스트오더를 가지고 프리오더를 만들어 보자

트리의 순회

80

<https://www.acmicpc.net/problem/2263>

- 포스트오더의 가장 마지막은 루트이다.

인오더

4	2	7	5	1	3	6
---	---	---	---	---	---	---

포스트오더

4	7	5	2	6	3	1
---	---	---	---	---	---	---

트리의 순회

<https://www.acmicpc.net/problem/2263>

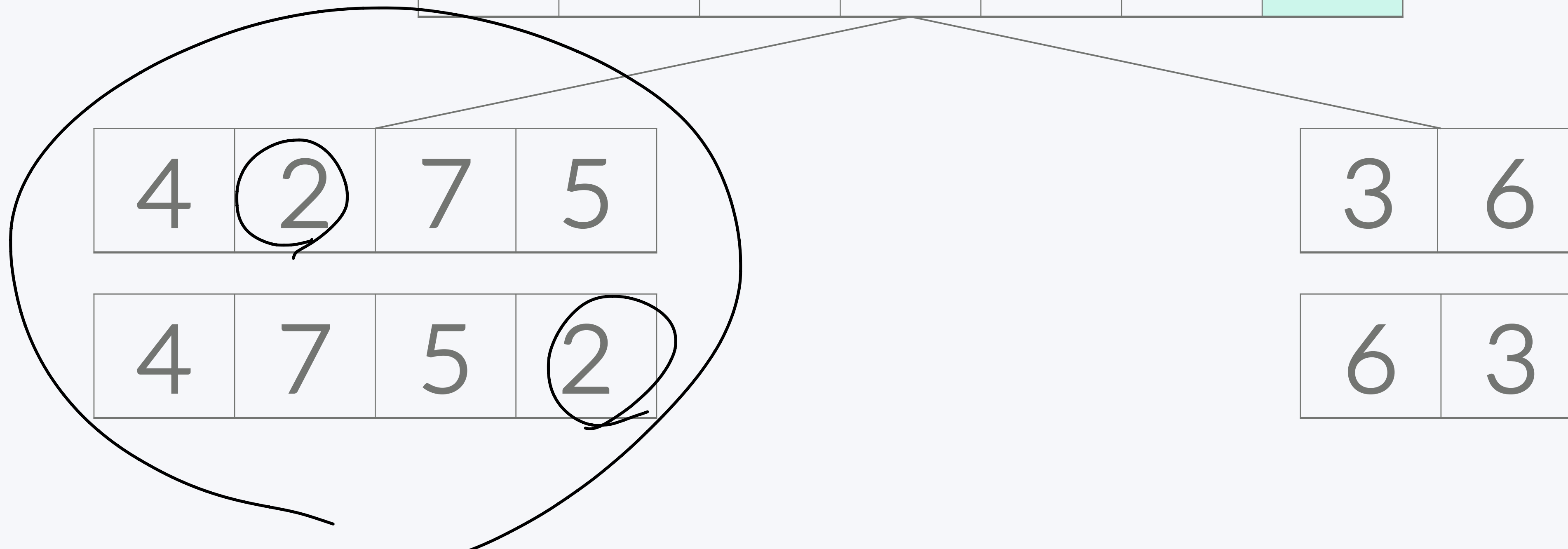
- 포스트오더의 가장 마지막은 루트이다.

	왼쪽	오른쪽
인오더	4275136	
포스트오더	4752631	
	왼쪽	오른쪽

트리의 순회

<https://www.acmicpc.net/problem/2263>

- 왼쪽과 오른쪽으로 나뉘서 풀 수 있다.



트리의 순회

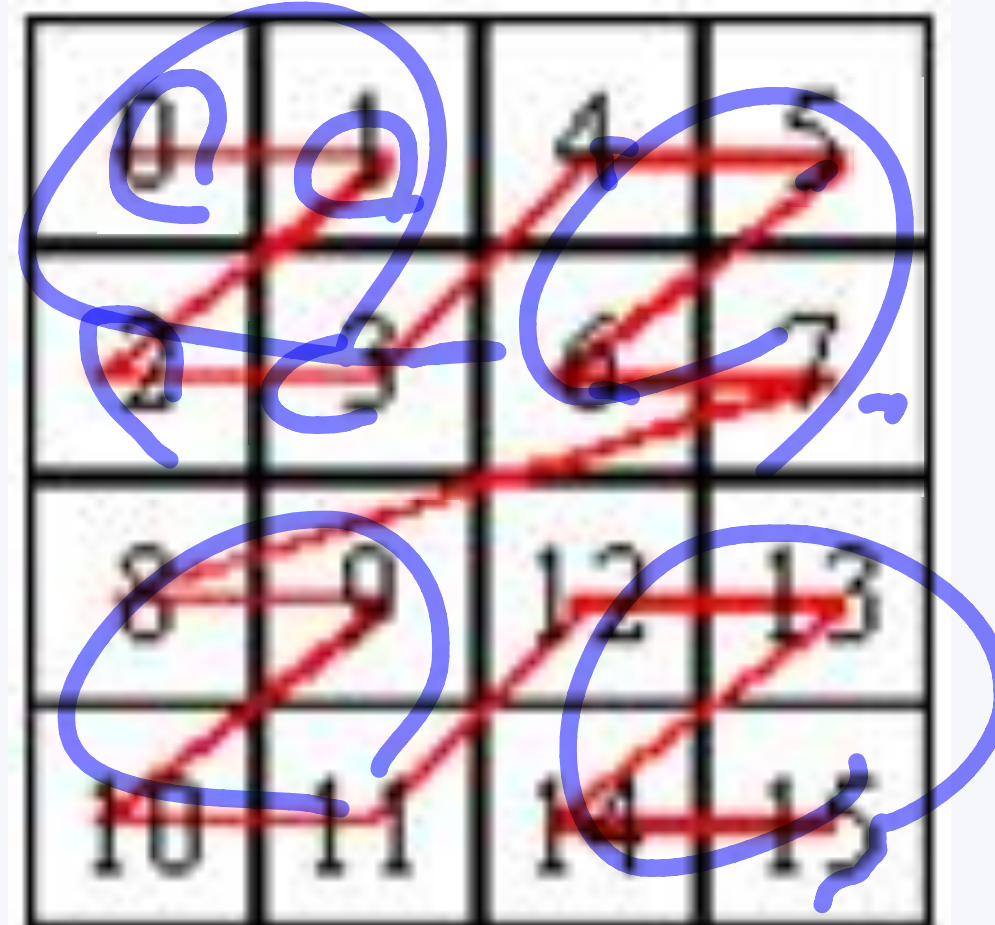
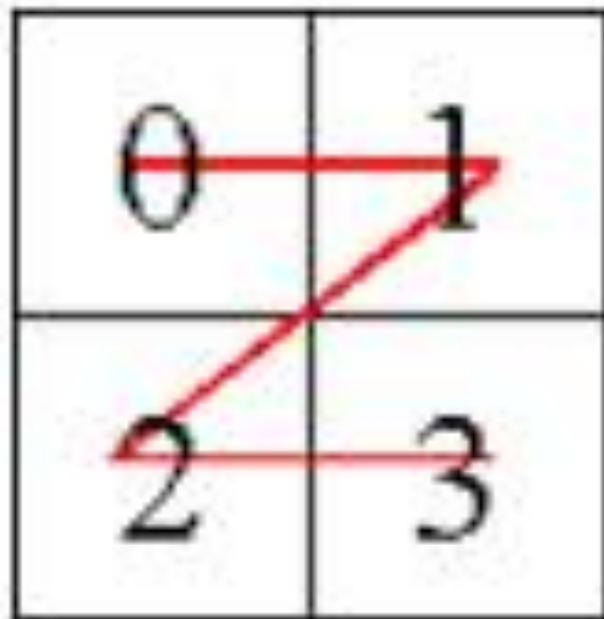
<https://www.acmicpc.net/problem/2263>

- 소스: <http://boj.kr/51b1d50cb8e3414390f11c8b3d0699df>

Z

<https://www.acmicpc.net/problem/1074>

- 만약, 2차원 배열의 크기가 $2^N * 2^N$ 라서 왼쪽 위에 있는 칸이 하나가 아니라면, 배열을 4등분한 후에 (크기가 같은 $2^{(N-1)}$ 로) 재귀적으로 순서대로 방문한다.



<https://www.acmicpc.net/problem/1074>

- 분할 정복을 이용해서 현재 칸이 4등분 했을 때, 어디에 속하는지 알아보고 분할 정복을 이용한다.

Z

<https://www.acmicpc.net/problem/1074>

- 소스: <http://boj.kr/8b26332358214825801b177d6d587dd7>

사분면

<https://www.acmicpc.net/problem/1891>

- 하나의 좌표평면은
- 네 개의 사분면으로 나뉜다



사분면

<https://www.acmicpc.net/problem/1891>

- 사분면은 다시 사분면으로 나뉜다

22	21	12	11
23	24	13	14
32	31	42	41
33	34	43	44

사분면

<https://www.acmicpc.net/problem/1891>

- 사분면 조각이 주어졌을 때
 - 입력으로 주어진 만큼 이동했을 때
 - 몇 번 조각으로 이동하는가?
-
- 341을 오른쪽으로 2칸, 위로 1칸 이동하면

222	221	212	211	122	121	112	111
223	224	213	214	123	124	113	114
232	231	242	241	132	131	142	141
233	234	243	244	133	134	143	144
322	321	312	311	422	421	412	411
323	324	313	314	423	424	413	414
332	331	342	341	432	431	442	441
333	334	343	344	433	434	443	444

사분면

<https://www.acmicpc.net/problem/1891>

- 사분면 조각이 주어졌을 때
 - 입력으로 주어진 만큼 이동했을 때
 - 몇 번 조각으로 이동하는가?
-
- 341을 오른쪽으로 2칸, 위로 1칸 이동하면
 - 424에 도착한다

222	221	212	211	122	121	112	111
223	224	213	214	123	124	113	114
232	231	242	241	132	131	142	141
233	234	243	244	133	134	143	144
322	321	312	311	422	421	412	411
323	324	313	314	423	424	413	414
332	331	342	341	432	431	442	441
333	334	343	344	433	434	443	444

사분면

<https://www.acmicpc.net/problem/1891>

- 두 개의 문제로 나누어져 있다고 볼 수 있다

사분면

<https://www.acmicpc.net/problem/1891>

- 두 개의 문제로 나누어져 있다고 볼 수 있다
1. 어떤 사분면 조각의 좌표는 몇인가?
 2. 어떤 좌표의 사분면 조각은 무엇인가?

사분면

<https://www.acmicpc.net/problem/1891>

- 좌표를 매겨보자

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

- 341을 찾아보자

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

- 341을 찾아보자

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

- 341을 찾아보자

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

- `go(index, r, c, size)`
- index번째 글자를 찾아야 함
- 가장 왼쪽 위 칸은 r행 c열
- 변의 길이는 size

사분면

<https://www.acmicpc.net/problem/1891>

- `go(index, r, c, size)`
- 1번 사분면
 - `go(index, r, c+size/2, size/2)`
- 2번 사분면
- 3번 사분면
- 4번 사분면

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

- `go(index, r, c, size)`
- 1번 사분면
 - `go(index, r, c+size/2, size/2)`
- 2번 사분면
 - `go(index, r, c, size/2)`
- 3번 사분면
- 4번 사분면

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

100

<https://www.acmicpc.net/problem/1891>

- `go(index, r, c, size)`
- 1번 사분면
 - `go(index, r, c+size/2, size/2)`
- 2번 사분면
 - `go(index, r, c, size/2)`
- 3번 사분면
 - `go(index, r+size/2, c, size/2)`
- 4번 사분면

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

- `go(index, r, c, size)`
- 1번 사분면
 - `go(index, r, c+size/2, size/2)`
- 2번 사분면
 - `go(index, r, c, size/2)`
- 3번 사분면
 - `go(index, r+size/2, c, size/2)`
- 4번 사분면
 - `go(index, r+size/2, c+size/2, size/2)`

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

사분면

<https://www.acmicpc.net/problem/1891>

```
pair<long long, long long> go(int index, long long r, long long c,
long long size) {
    if (size == 1) {
        return make_pair(r, c);
    } else {
        if (a[index] == '1') {
            return go(index+1, r, c+size/2, size/2);
        } else if (a[index] == '2') {
            return go(index+1, r, c, size/2);
        } else if (a[index] == '3') {
            return go(index+1, r+size/2, c, size/2);
        } else if (a[index] == '4') {
            return go(index+1, r+size/2, c+size/2, size/2);
        }
    }
}
return make_pair(0, 0);
}
```

사분면

<https://www.acmicpc.net/problem/1891>

```
string gogo(long long r, long long c, long long size, long long x,
long long y) {
    if (size == 1) return "";
    if (x < r+size/2 && y < c+size/2) {
        return "2" + gogo(r, c, size/2, x, y);
    } else if (x < r+size/2 && y >= c+size/2) {
        return "1" + gogo(r, c+size/2, size/2, x, y);
    } else if (x >= r+size/2 && y < c+size/2) {
        return "3" + gogo(r+size/2, c, size/2, x, y);
    } else {
        return "4" + gogo(r+size/2, c+size/2, size/2, x, y);
    }
}
```

사분면

104

<https://www.acmicpc.net/problem/1891>

- 소스: <http://boj.kr/d0378a47edb74c8cb41c585c72a38665>

버블 소트

<https://www.acmicpc.net/problem/1517>

- N개로 이루어진 수열 $A[1], A[2], \dots, A[N]$ 이 있을 때
- Swap이 총 몇 번 발생하는지 알아내는 문제
- $3\ 2\ 1 \rightarrow 2\ 3\ 1 \rightarrow 2\ 1\ 3 \rightarrow 1\ 2\ 3$ (3번)

10 1 5 2 3



버블 소트

<https://www.acmicpc.net/problem/1517>

- 이 문제는 수열에서 inversion의 개수를 세는 문제이다.
- inversion: $A[i] > A[j]$ ($i < j$)
- 머지 소트를 하면서 문제를 풀 수 있다.

T<J A<C<D<A<C<J

106



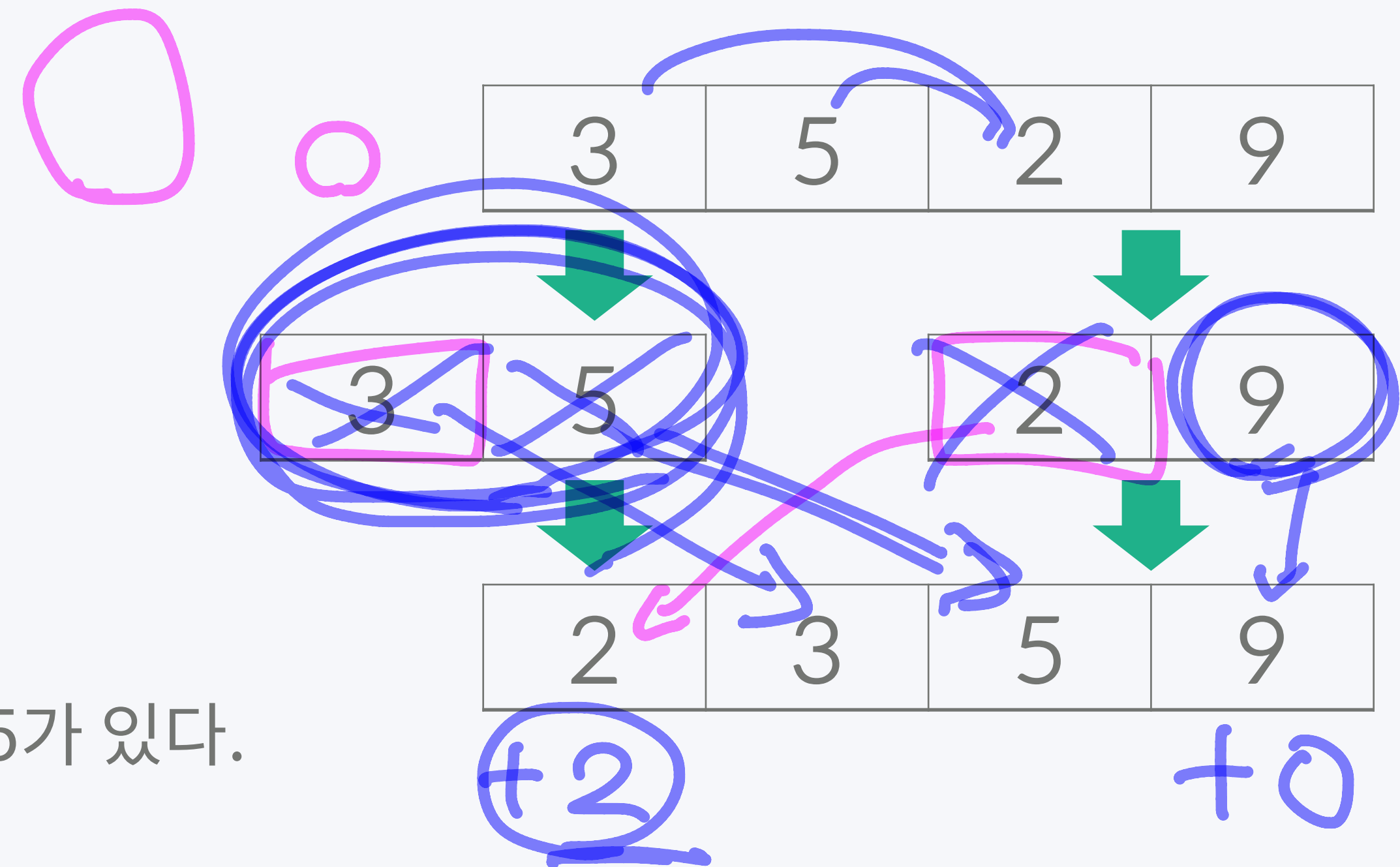
$O(N^2)$

버블 소트

107

<https://www.acmicpc.net/problem/1517>

- 3 5 2 9의 inversion의 개수는
- $3 > 2, 5 > 2$ 로 2개다.
- 오른쪽 절반이 이동할 때
- 왼쪽 절반에서 아직 이동하지 않은 것의 개수가
- 그 때의 inversion의 개수이다.
- 2가 먼저 이동하는데, 그 때 왼쪽 절반에는 3과 5가 있다.
- 이것은 $3 > 2, 5 > 2$ 를 의미한다.

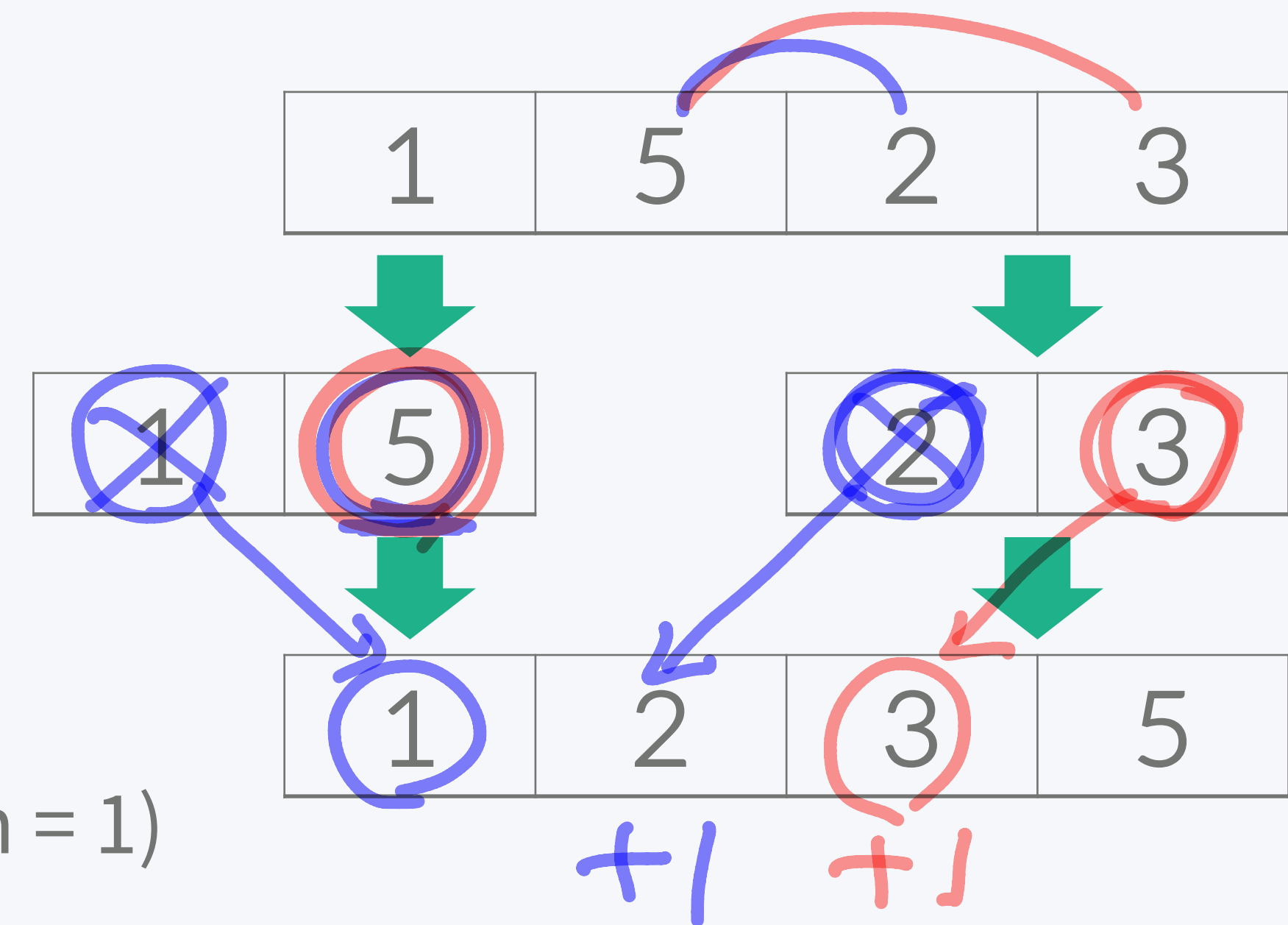


버블 소트

108

<https://www.acmicpc.net/problem/1517>

- 1 5 2 3의 inversion의 개수는
- $5 > 2, 5 > 3$ 로 2개다.
- 오른쪽 절반이 이동할 때
- 왼쪽 절반에서 아직 이동하지 않은 것의 개수가
- 그 때의 inversion의 개수이다.
- 2가 이동할 때, 왼쪽 절반에는 5가 남아있다. (inversion = 1)
- 3이 이동할 때도 왼쪽 절반에는 5가 남아있다. (inversion = 1)



버블 소트

109

<https://www.acmicpc.net/problem/1517>

```
long long solve(int start, int end) {  
    if (start == end) {  
        return 0;  
    }  
    int mid = (start+end)/2;  
    long long ans = solve(start, mid) + solve(mid+1, end);  
    int i = start;  
    int j = mid+1;  
    int k = 0;
```

버블 소트

110

<https://www.acmicpc.net/problem/1517>

```
while (i <= mid || j <= end) {
    if (i <= mid && (j > end || a[i] <= a[j])) {
        b[k++] = a[i++];
    } else {
        ans += (long long)(mid-i+1);
        b[k++] = a[j++];
    }
}
for (int i=start; i<=end; i++) {
    a[i] = b[i-start];
}
return ans;
}
```

버블 소트

111

<https://www.acmicpc.net/problem/1517>

- 소스: <http://boj.kr/ebc183787cee45fea2eacaefed7d55a9>

가장 가까운 두 점

112

<https://www.acmicpc.net/problem/2261>

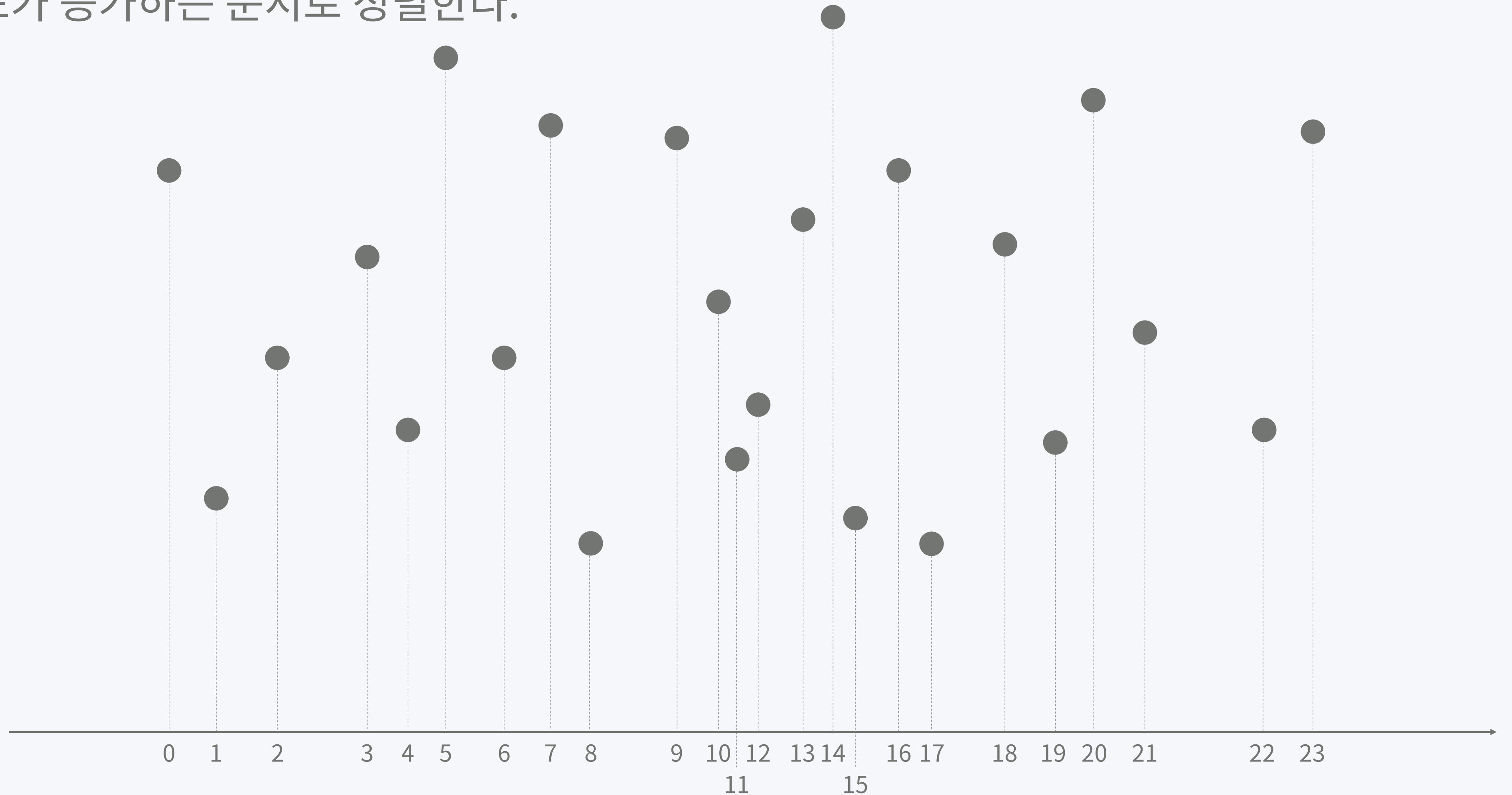
- 2차원 평면 위의 N개의 점 중에서 가장 가까운 두 점을 찾는 문제
- $2 \leq N \leq 100,000$

가장 가까운 두 점

113

<https://www.acmicpc.net/problem/2261>

- 먼저 점을 x좌표가 증가하는 순서로 정렬한다.

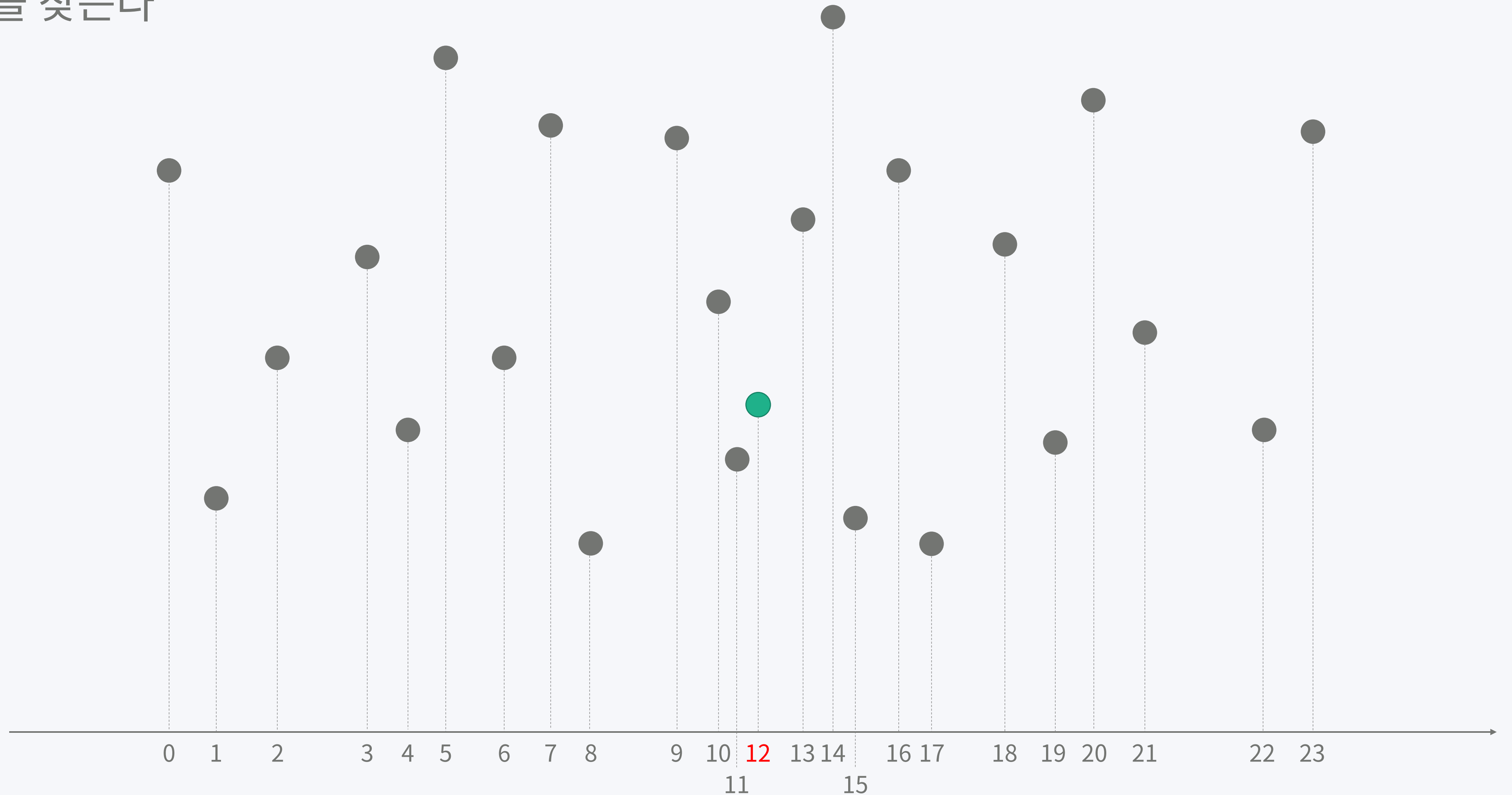


가장 가까운 두 점

114

<https://www.acmicpc.net/problem/2261>

- 중간에 있는 점을 찾는다

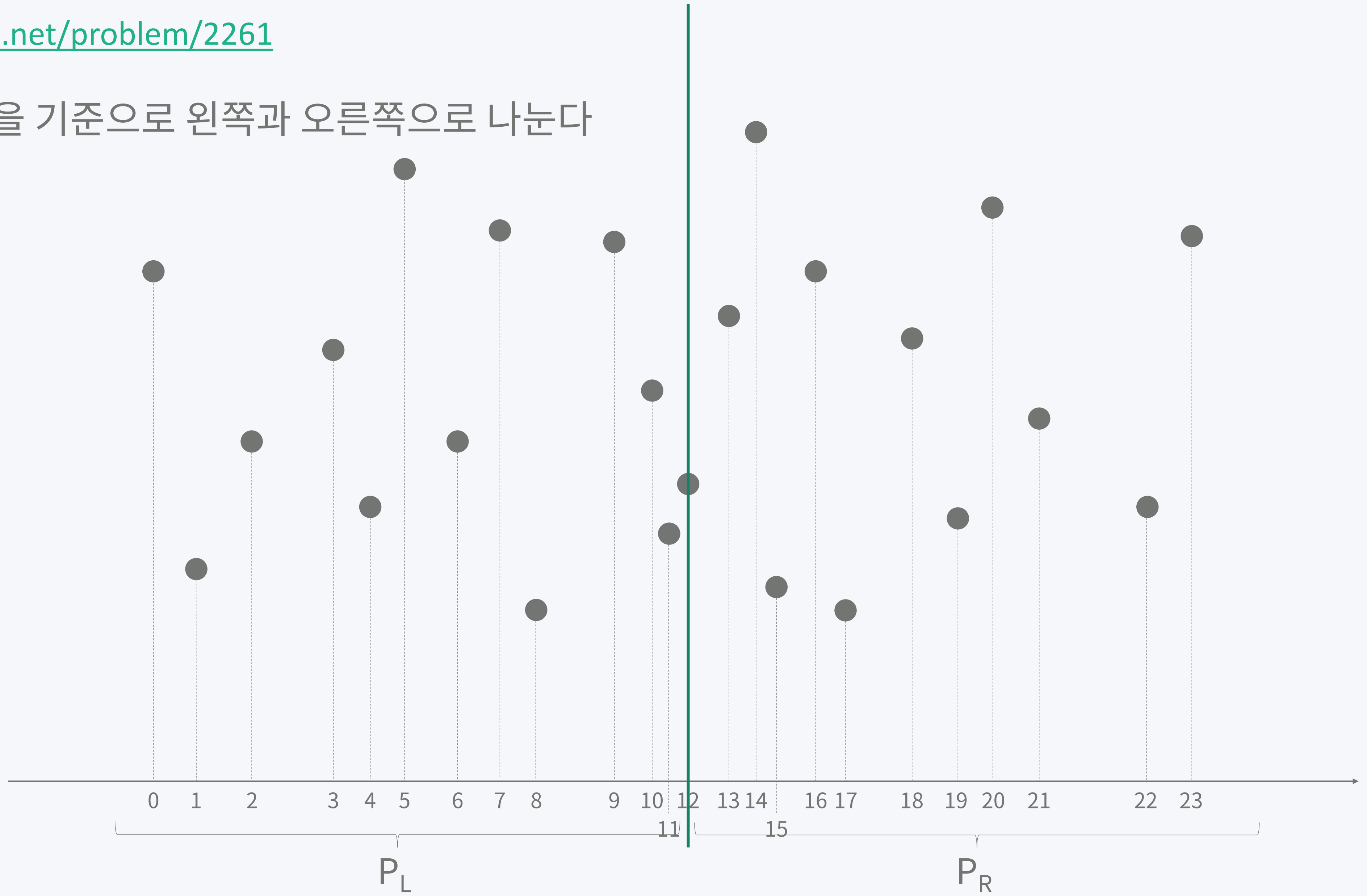


가장 가까운 두 점

115

<https://www.acmicpc.net/problem/2261>

- 중간에 있는 점을 기준으로 왼쪽과 오른쪽으로 나눈다
- 왼쪽: P_L
- 오른쪽: P_R



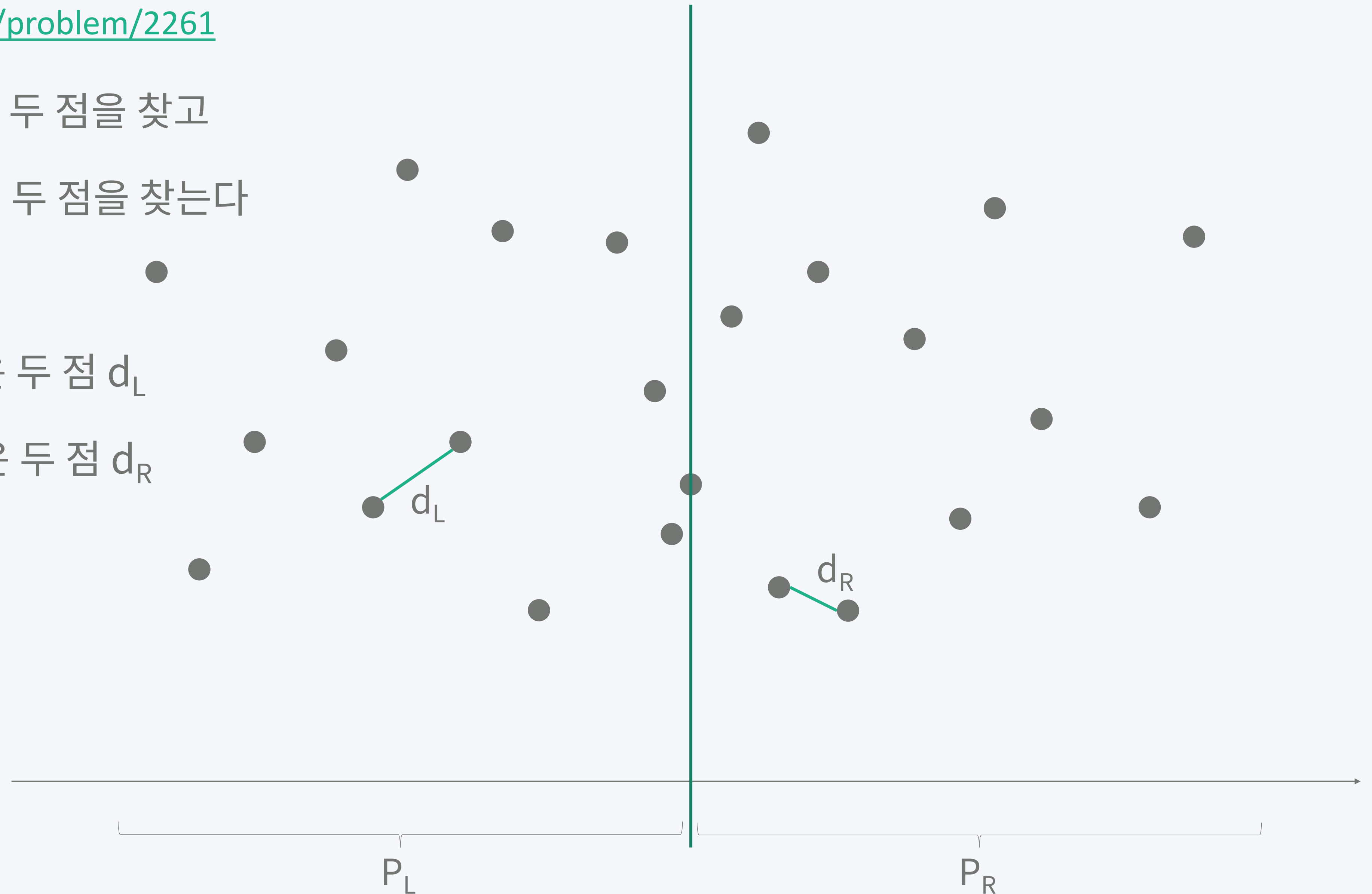
가장 가까운 두 점

116

<https://www.acmicpc.net/problem/2261>

- P_L 에서 가장 가까운 두 점을 찾고
- P_R 에서 가장 가까운 두 점을 찾는다

- P_L 에서 가장 가까운 두 점 d_L
- P_R 에서 가장 가까운 두 점 d_R

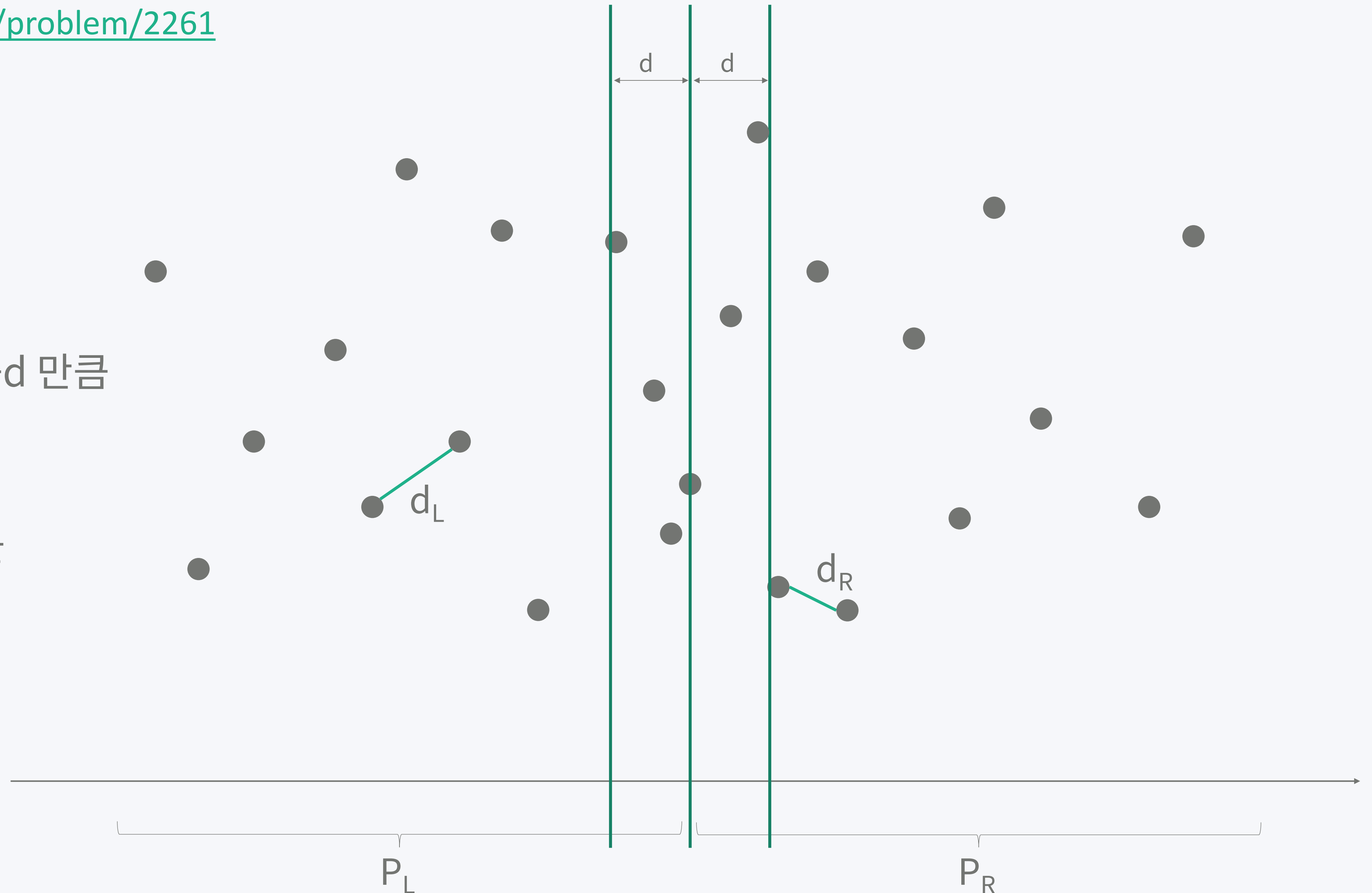


가장 가까운 두 점

117

<https://www.acmicpc.net/problem/2261>

- $d = \min(d_L, d_R)$
- 이라고 했을 때
- 가운데 점으로부터
- 가운데로부터 $-d, +d$ 만큼
- 떨어진 곳에서
- 가장 가까운 두 점을
- 찾아야 한다

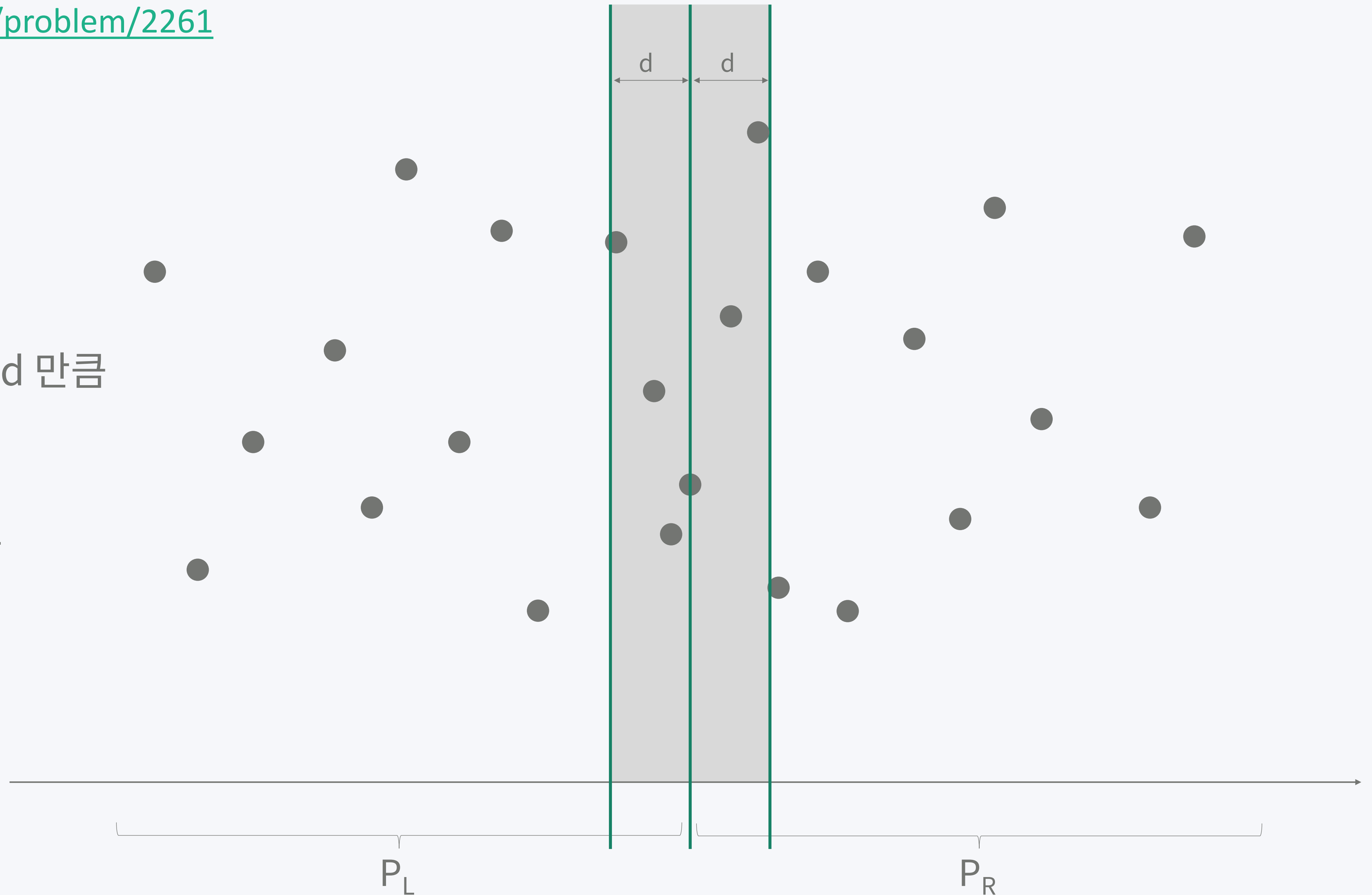


가장 가까운 두 점

118

<https://www.acmicpc.net/problem/2261>

- $d = \min(d_L, d_R)$
- 이라고 했을 때
- 가운데 점으로부터
- 가운데로부터 $-d, +d$ 만큼
- 떨어진 곳에서
- 가장 가까운 두 점을
- 찾아야 한다

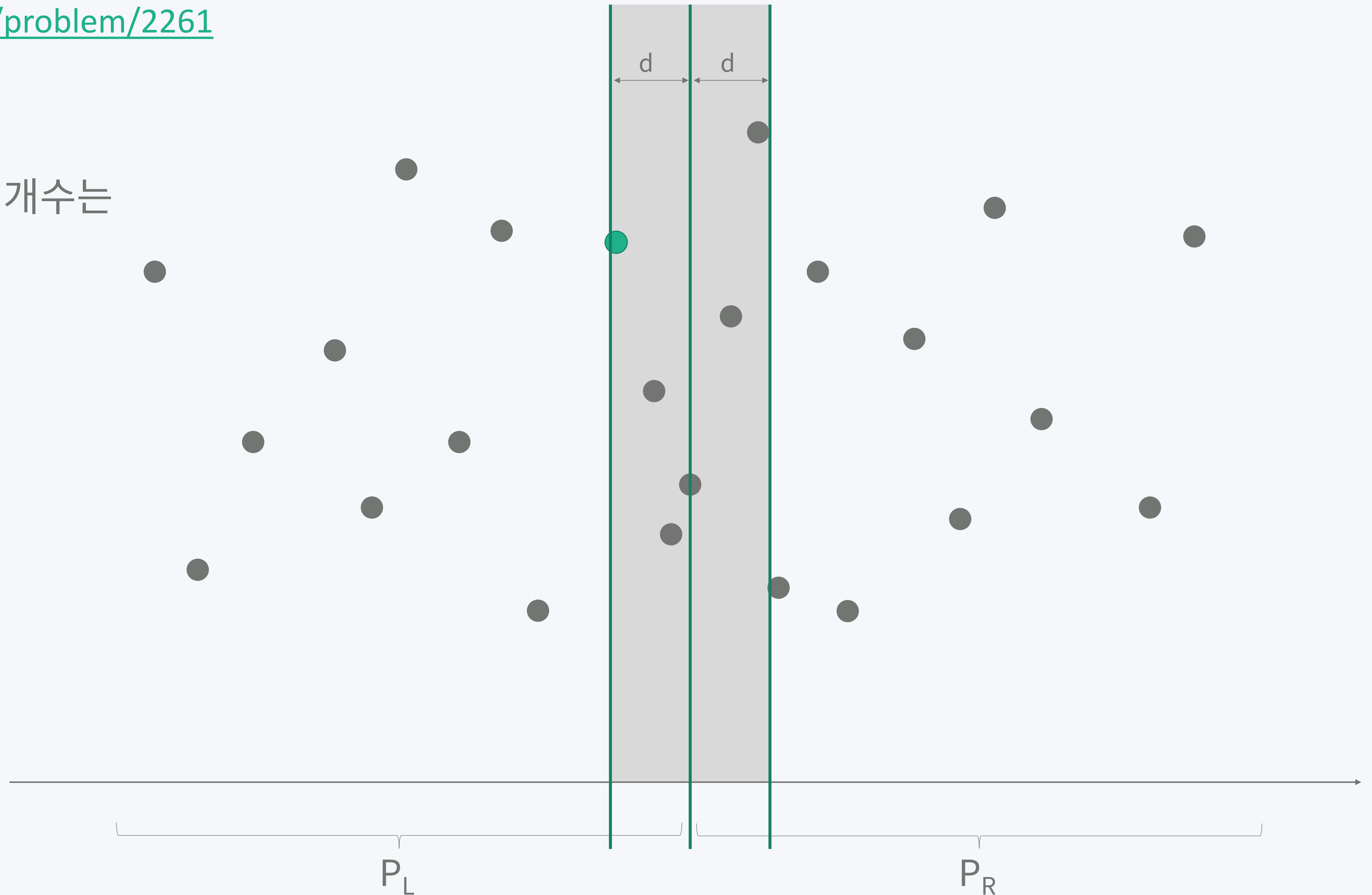


가장 가까운 두 점

119

<https://www.acmicpc.net/problem/2261>

- 한 점당
- 살펴봐야 하는 점의 개수는
- 6개이다.

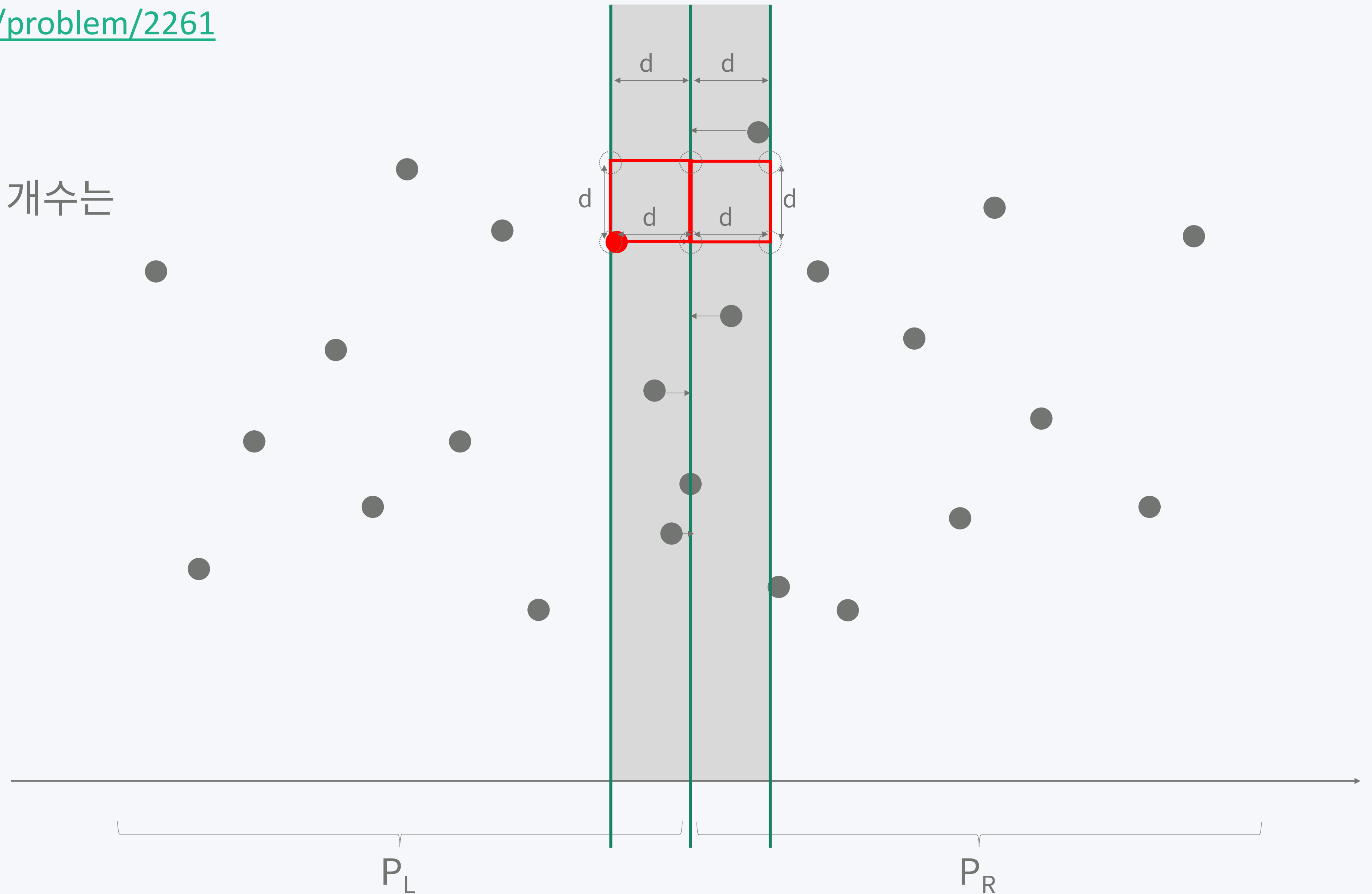


가장 가까운 두 점

120

<https://www.acmicpc.net/problem/2261>

- 한 점당
- 살펴봐야 하는 점의 개수는
- 6개이다.



가장 가까운 두 점

121

<https://www.acmicpc.net/problem/2261>

- 소스: <http://boj.kr/2e3849f2abf24668822eebf8be794a01>