

코딩 테스트 대비 핵심 알고리즘

핵심 유형 문제풀이

핵심 유형 문제풀이 | 다양한 문제를 접하며 코딩 테스트에 익숙해지기

강사 나동빈

코딩 테스트 대비 핵심 알고리즘

핵심 유형 문제풀이

코딩 테스트 대비
핵심 유형 문제풀이

혼자 힘으로 풀어보기

코테 대비
핵심 유형 문제풀이

문제 제목: 경사로

문제 난이도: ★★☆☆☆

문제 유형: 시뮬레이션

추천 풀이 시간: 60분

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 길: 한쪽 끝에서 다른 한쪽 끝까지 이어진 통로
- 본 문제는 “지날 수 있는 길(높이가 같은 길)”의 개수를 세는 문제다.

3	3	3	3	3	3
2	3	3	3	3	3
2	2	2	3	2	3
1	1	1	3	2	2
1	1	1	3	3	1
1	1	2	3	3	2

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 길: 한쪽 끝에서 다른 한쪽 끝까지 이어진 통로
- 본 문제는 “지날 수 있는 길(높이가 같은 길)”의 개수를 세는 문제다.

3	3	3	3	3	3	← 지날 수 있는 길 (높이가 모두 같음)
2	3	3	3	3	3	
2	2	2	3	2	3	
1	1	1	3	2	2	
1	1	1	3	3	1	
1	1	2	3	3	2	

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 길: 한쪽 끝에서 다른 한쪽 끝까지 이어진 통로
- 본 문제는 “지날 수 있는 길(높이가 같은 길)”의 개수를 세는 문제다.

3	3	3	3	3	3
2	3	3	3	3	3
2	2	2	3	2	3
1	1	1	3	2	2
1	1	1	3	3	1
1	1	2	3	3	2

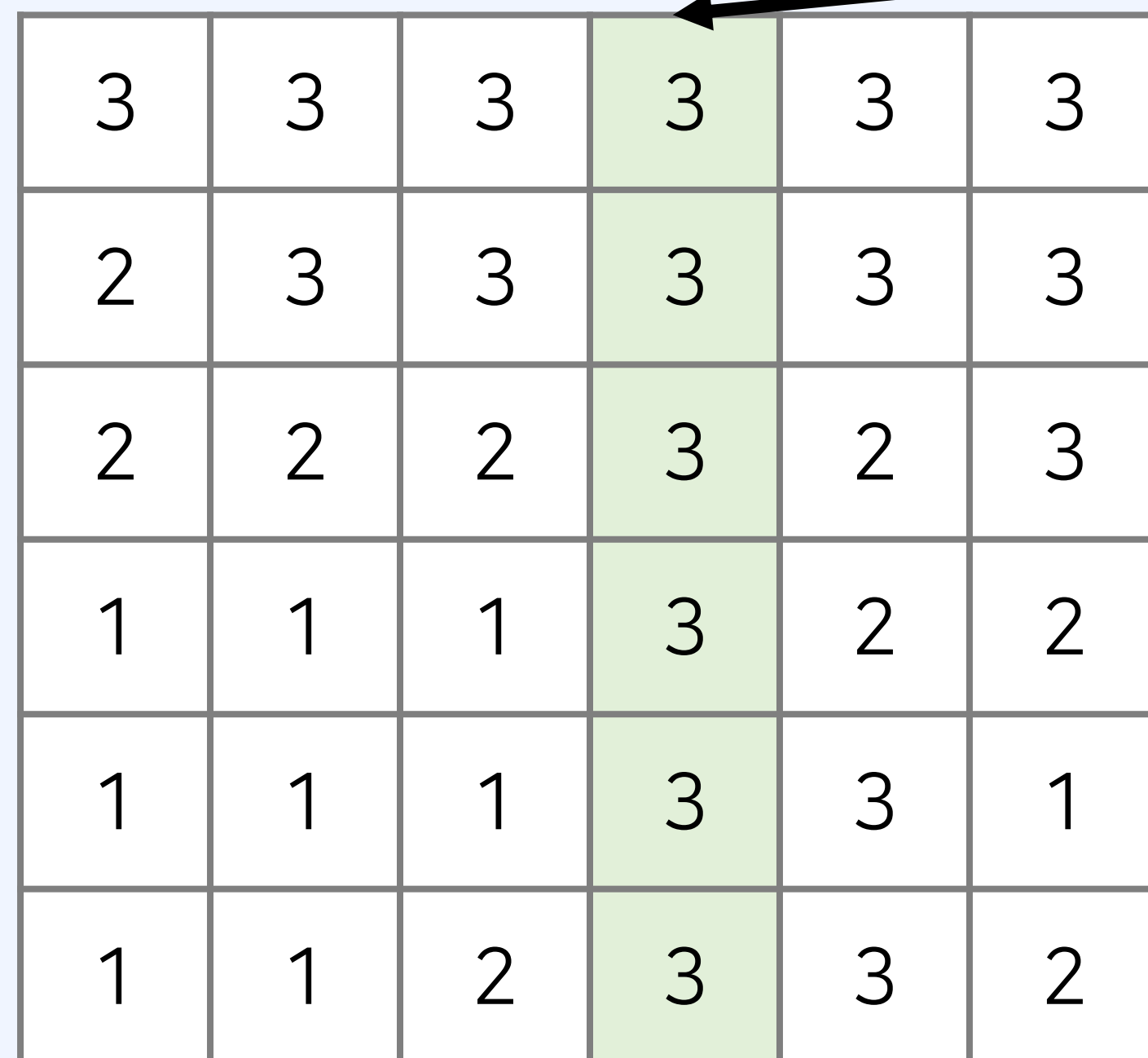
← 지날 수 없는 길

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 길: 한쪽 끝에서 다른 한쪽 끝까지 이어진 통로
- 본 문제는 “지날 수 있는 길(높이가 같은 길)”의 개수를 세는 문제다.



3	3	3	3	3	3
2	3	3	3	3	3
2	2	2	3	2	3
1	1	1	3	2	2
1	1	1	3	3	1
1	1	2	3	3	2

지날 수 있는 길 (높이가 모두 같음)

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 길: 한쪽 끝에서 다른 한쪽 끝까지 이어진 통로
- 본 문제는 “지날 수 있는 길(높이가 같은 길)”의 개수를 세는 문제다.

3	3	3	3	3	3
2	3	3	3	3	3
2	2	2	3	2	3
1	1	1	3	2	2
1	1	1	3	3	1
1	1	2	3	3	2

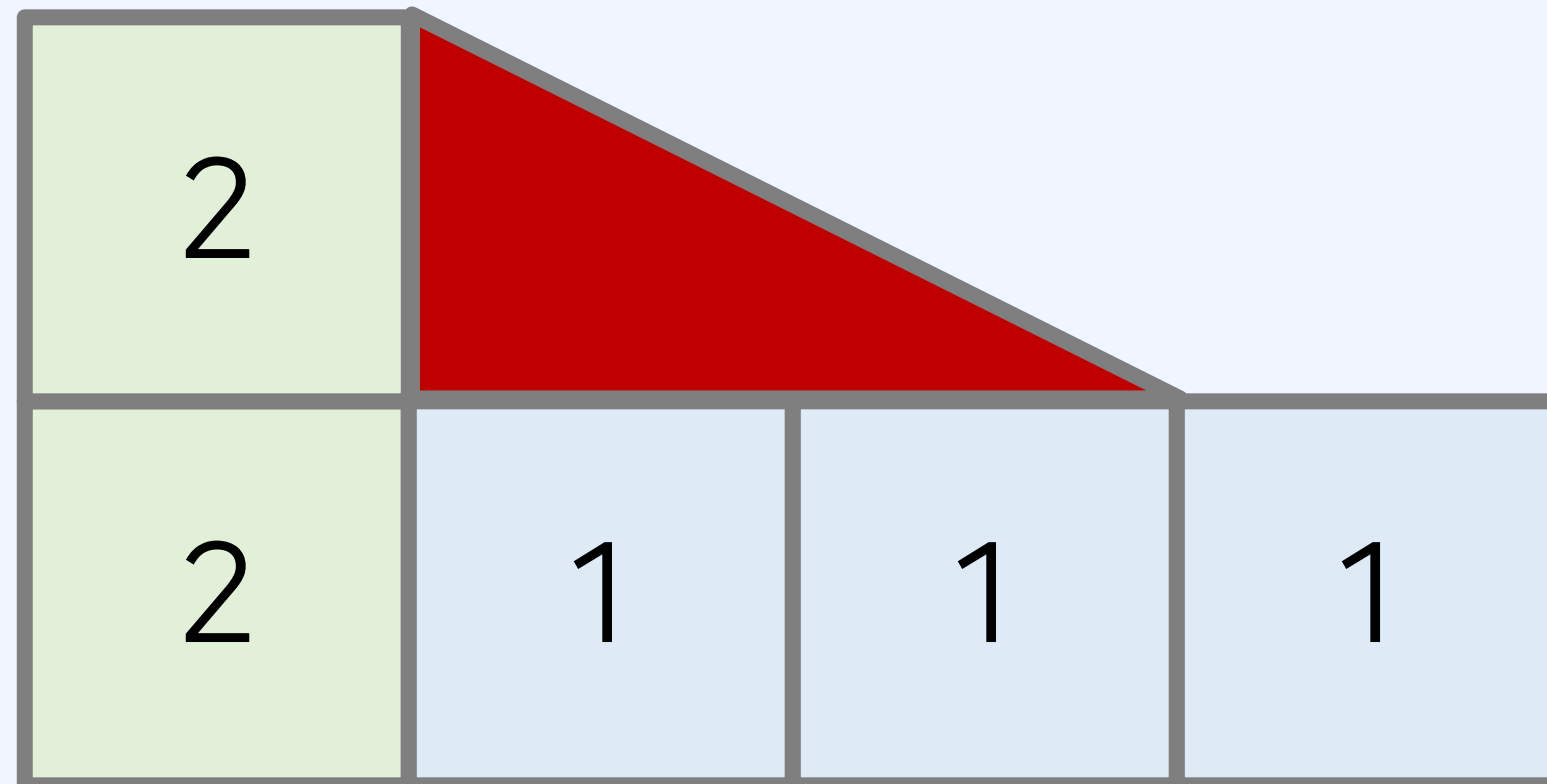
지날 수 없는 길

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 다만, 길을 지날 때 “경사로”를 설치하여 지날 수 있다.
- 경사로의 길이는 L 이고 높이는 항상 1이며, 경사로는 무한히 존재한다.
- 경사로를 놓을 낮은 칸의 높이는 모두 같아야 하고, L 개의 칸이 연속되어 있어야 한다.
- 경사로를 놓을 때 낮은 칸과 높은 칸의 높이 차이가 1이어야 한다.



- $L = 2$ 인 경우 예시

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 맵의 크기는 $N \times N$ 이므로, 최대 10,000개의 원소가 존재한다. (N 이 최대 100)
- [유의 사항] 미리 지도에 경사로를 설치하는 게 아니라, 길을 지나갈 때 경사로를 놓아서 지나가는 것이 목표다.
- 따라서 각 행과 각 열을 "개별적으로" 처리할 수 있다.
- **[해결 방법]** 하나의 1차원 배열이 있을 때, 지나갈 수 있는지 검사하는 함수를 작성하자.

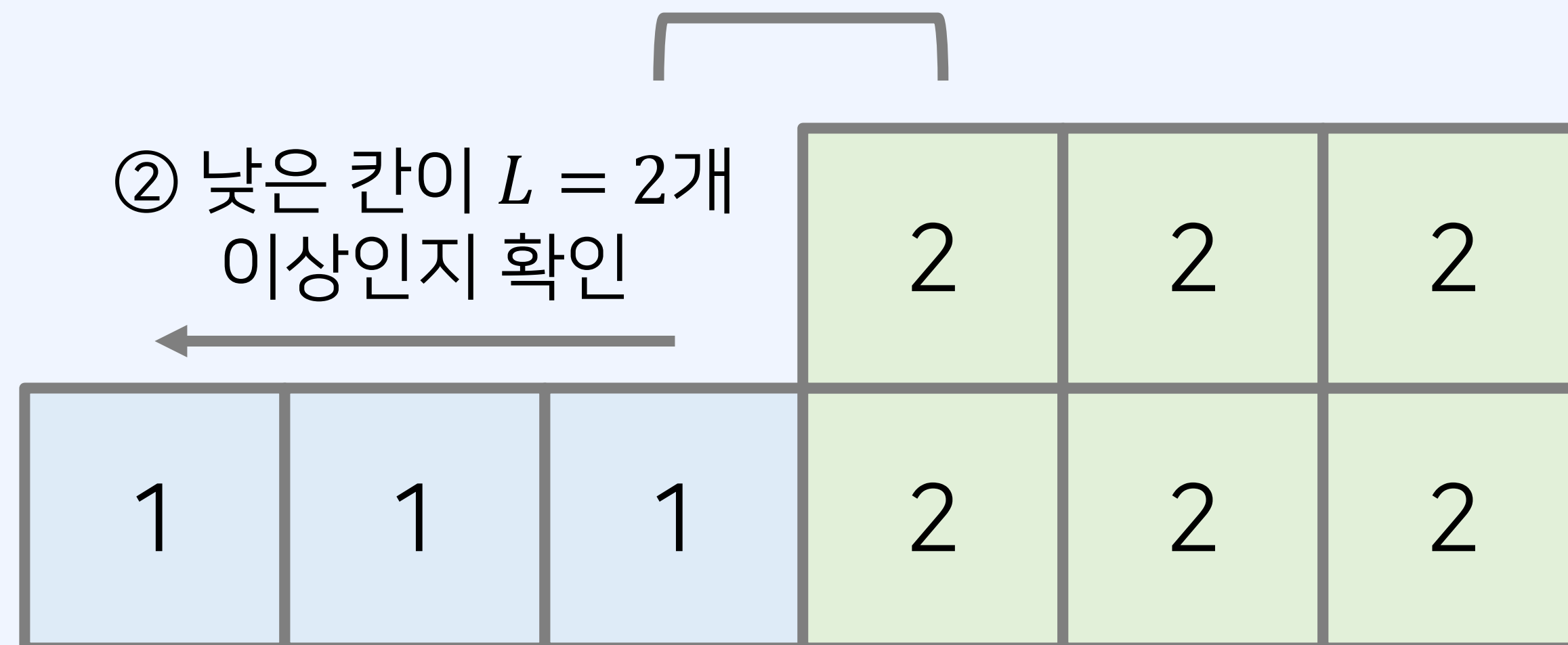
코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 1차원 배열이 들어왔을 때, 원소를 차례대로 확인한다.
- 인접한 위치의 높이 차이가 1인 경우, **낮은 위치가 L 개 이상 연속**되면 지나갈 수 있다.
- **[예시]** $L = 2$ 이고, **“올라가는”** 경우: 낮은 칸이 $L = 2$ 개 이상인지 확인한다.

① 높이 차이가 1일 때



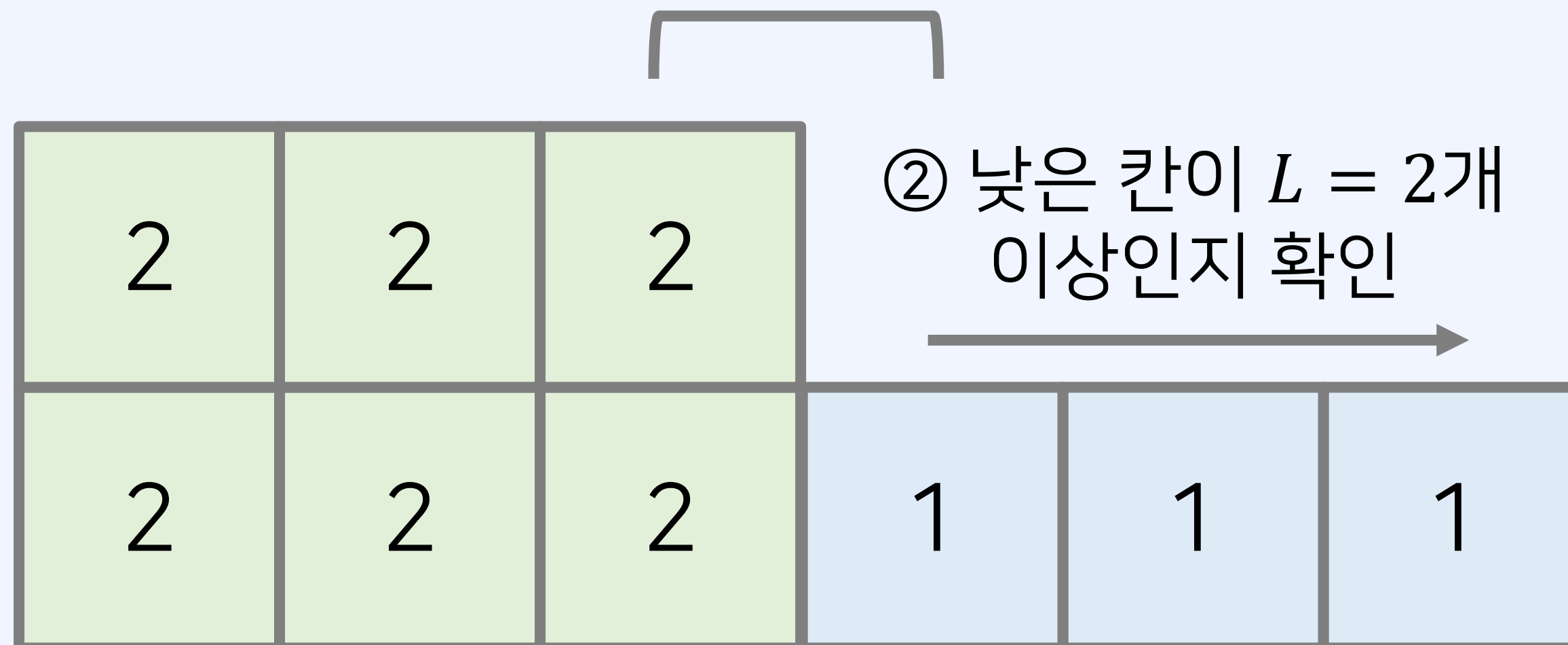
코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 1차원 배열이 들어왔을 때, 원소를 차례대로 확인한다.
- 인접한 위치의 높이 차이가 1인 경우, **낮은 위치가 L 개 이상 연속**되면 지나갈 수 있다.
- **[예시]** $L = 2$ 이고, “**내려가는**” 경우: 낮은 칸이 $L = 2$ 개 이상인지 확인한다.

① 높이 차이가 1일 때



코딩 테스트 대비 핵심 유형 문제풀이

소스 코드 1)

코테 대비.

핵심 유형 문제풀이

```
def check(row):
    visited = [False] * n # 각 위치에 대한 경사로 설치 여부
    for i in range(n - 1):
        # 오른쪽 위치와 높이 차이가 2 이상인 경우, 불가능
        if abs(row[i] - row[i + 1]) >= 2:
            return False
        # 높이 차이가 1인 경우
        if abs(row[i] - row[i + 1]) == 1:
            # L개 만큼 확인하며
            for j in range(1):
                # 한 칸 내려가는 경우, 오른쪽으로 이동하며
                if row[i] - 1 == row[i + 1]:
                    current = i + 1 + j
                    if current >= n: return False # 공간을 벗어나는 경우
                    if row[i + 1] != row[current]: return False # 높이가 다르다면
                # 한 칸 올라가는 경우, 왼쪽으로 이동하며
            elif row[i] + 1 == row[i + 1]:
                current = i - j
                if current < 0: return False # 공간을 벗어나는 경우
                if row[i] != row[current]: return False # 높이가 다르다면
            # 경사로가 이미 있다면, 지나가기 불가능
            if visited[current]: return False
            visited[current] = True # 경사로 설치하기

    return True
```

코딩 테스트 대비 핵심 유형 문제풀이

소스 코드 2)

코테 대비

핵심 유형 문제풀이

```
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline

n, l = map(int, input().split())
# 전체 N X N 맵 입력받기
arr = []
for i in range(n):
    row = list(map(int, input().split()))
    arr.append(row)

answer = 0
for row in arr: # 하나씩 행(row)을 확인하며
    if check(row):
        answer += 1
for i in range(n): # 하나씩 열(column)을 확인하며
    column = []
    for j in range(n):
        column.append(arr[j][i])
    if check(column):
        answer += 1
print(answer) # 지나갈 수 있는 길의 합
```

코딩 테스트 대비
핵심 유형 문제풀이

혼자 힘으로 풀어보기

코테 대비.
핵심 유형 문제풀이

문제 제목: 미세먼지 안녕!

문제 난이도: ★★☆☆☆

문제 유형: 시뮬레이션

추천 풀이 시간: 60분

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- R 과 C 가 최대 50이므로, 총 원소는 최대 2,500개다.
- 시간(초) 정보 T 가 최대 1,000이다.
- 각 초마다 $R \times C$ 크기의 확산 결과 배열을 생성할 수 있다.
- 미세먼지가 있는 모든 위치에서 확산을 진행한다.
- 연산할 양이 많은 문제이므로, 가능하면 **PyPy3 언어로 제출**한다.

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

[1. 미세먼지의 확산]

- (r, c) 에 있는 미세먼지는 인접한 네 방향으로 확산된다.
- 인접한 방향에 공기청정기가 있거나, 칸이 없으면 그 방향으로의 확산이 일어나지 않는다.
- 확산되는 양은 $A_{r,c}/5$ 이고 소수점은 버린다.
- (r, c) 에 남은 미세먼지의 양은 $A_{r,c} - (A_{r,c}/5) \times (\text{확산된 방향의 개수})$ 이다.

코딩 테스트 대비
핵심 유형 문제풀이

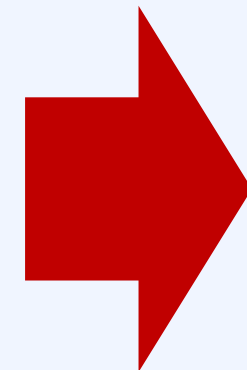
문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

[1. 미세먼지의 확산]

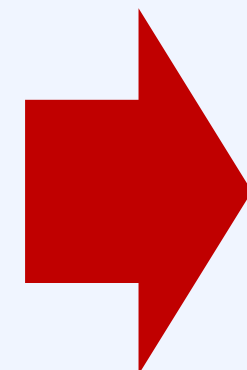
- 미세먼지가 확산되는 예시는 다음과 같다.

	34	
		7



	6	
6	10	7
	7	5

	5	
	20	



1	6	1
	9	4
	4	

 : 공기청정기가 있는 공간

코딩 테스트 대비 핵심 유형 문제풀이

소스 코드 1)

코테 대비.

핵심 유형 문제풀이

```
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline

# 방향(북, 동, 남, 서) 정의
dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]

# 맵의 크기(R X C)와 T초 정보 입력
r, c, t = map(int, input().split())
arr = [] # 전체 맵의 상태(미세먼지)
# 공기청정기의 위 칸(up_position), 아래 칸(down_position)
up_position = None
down_position = None
for i in range(r):
    arr.append(list(map(int, input().split())))
    for j in range(c):
        if arr[i][j] == -1: # 공기청정기인 경우
            up_position = (i - 1, j)
            down_position = (i, j)
```

코딩 테스트 대비 핵심 유형 문제풀이

소스 코드 2)

코테 대비

핵심 유형 문제풀이

```
# T초만큼 반복
for _ in range(t):
    result = [[0] * c for _ in range(r)] # 한 번 확산한 뒤의 결과 배열
    for x in range(r): # 각 위치에 대하여 미세먼지 확산
        for y in range(c):
            if arr[x][y] == -1: # 공기청정기가 있는 경우 무시
                result[x][y] = -1
                continue
            diffuse = arr[x][y] // 5 # 상, 하, 좌, 우 위치로 arr[x][y] // 5만큼씩 확산
            summary = 0
            for i in range(4):
                nx = x + dx[i]
                ny = y + dy[i]
                if nx < 0 or ny < 0 or nx >= r or ny >= c: # 범위를 벗어나는 경우 무시
                    continue
                if arr[nx][ny] == -1: # 공기청정기가 있는 경우 무시
                    continue
                summary += diffuse # 확산된 양
                result[nx][ny] += diffuse
            result[x][y] += arr[x][y] - summary # 확산되지 않고, 남아있는 미세먼지
    arr = result # 확산 완료
    cycle() # 공기청정기 가동

answer = 0
for row in arr: answer += sum(row)
print(answer + 2) # 공기 청정기로 2 감소한 것 되돌리기
```

코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비
핵심 유형 문제풀이

[2. 공기청정기 동작]

- 위쪽 공기청정기의 바람은 반시계방향으로 순환한다.
- 아래쪽 공기청정기의 바람은 시계방향으로 순환한다.
- 바람이 불면 미세먼지가 바람의 방향으로 모두 한 칸씩 이동한다.
- 공기청정기로 들어간 미세먼지는 모두 제거된다.

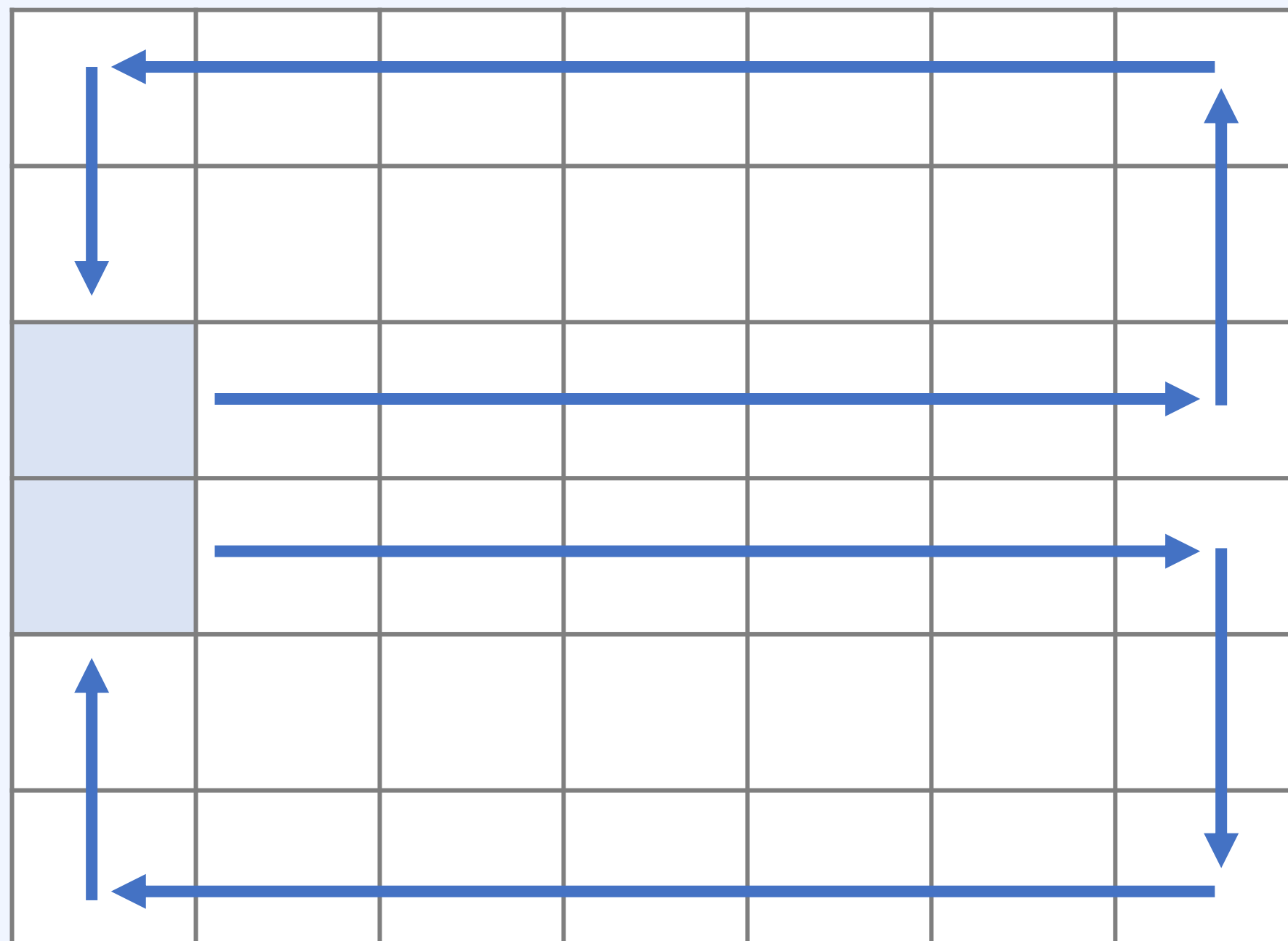
코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

[2. 공기청정기 동작]

- 위쪽 공기청정기의 바람은 반시계방향으로 순환한다.
- 아래쪽 공기청정기의 바람은 시계방향으로 순환한다.



□ : 공기청정기가 있는 공간

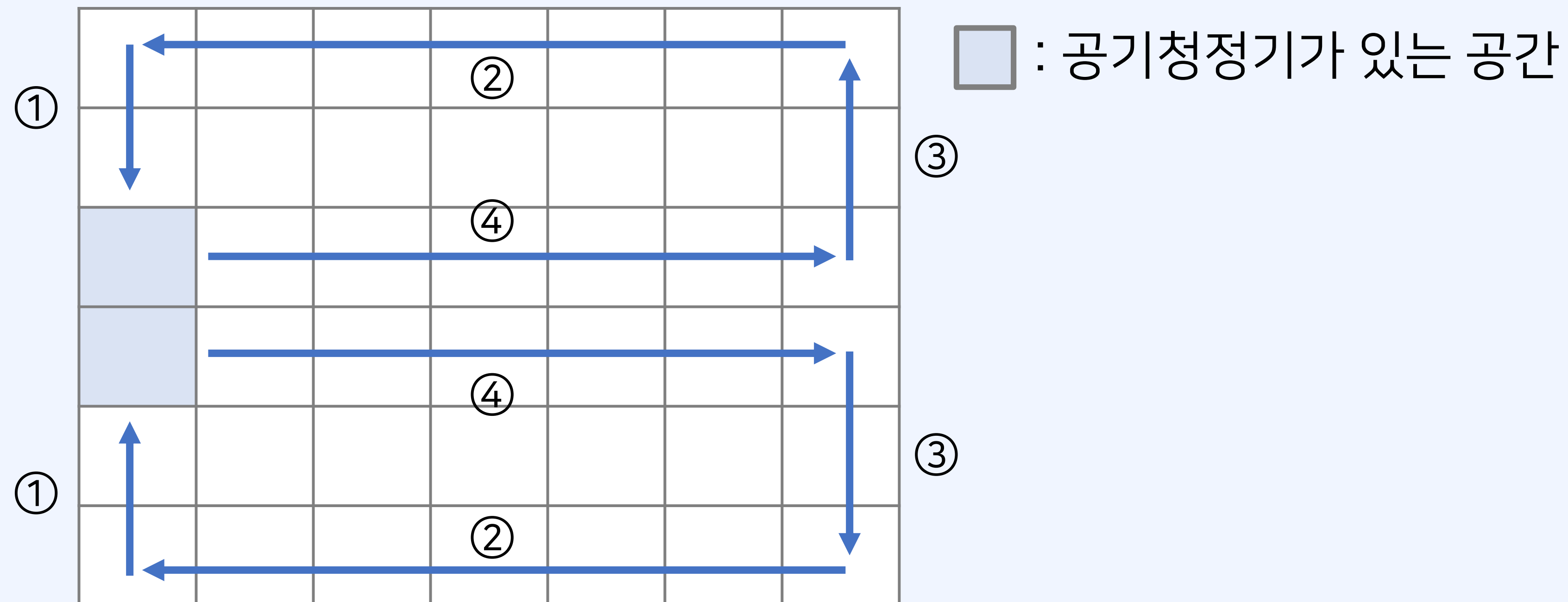
코딩 테스트 대비
핵심 유형 문제풀이

문제 해결 아이디어

코테 대비.
핵심 유형 문제풀이

[2. 공기청정기 동작]

- 구현상 **"빨아들여지는 위치"부터 처리**하는 것이 간단하다.



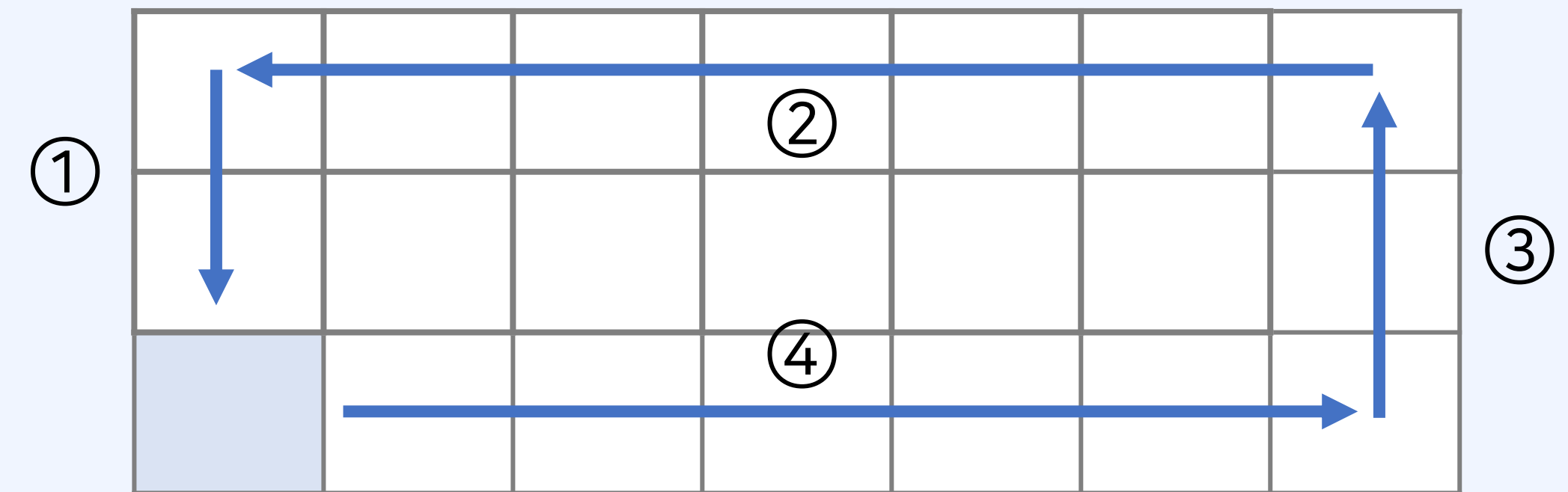
코딩 테스트 대비
핵심 유형 문제풀이

소스 코드 3)

코테 대비.
핵심 유형 문제풀이

공기청정기의 위 칸 처리하기(U → R → D → L 순서로 처리)

```
def up(arr, x, y, direction):
    if direction == 'U':
        nx, ny = x - 1, y # 위쪽 끝까지 도달한 경우
        if nx == -1:
            direction = 'R'
    if direction == 'R':
        nx, ny = x, y + 1 # 오른쪽 끝까지 도달한 경우
        if ny == c:
            direction = 'D'
    if direction == 'D':
        nx, ny = x + 1, y # 아래쪽 끝까지 도달한 경우
        if nx == up_position[0] + 1:
            direction = 'L'
    if direction == 'L':
        nx, ny = x, y - 1
    # 공기청정기로 돌아올 때까지
    if (nx, ny) != up_position:
        # (nx, ny) 위치에 있는 미세먼지를 (x, y) 위치로 이동
        arr[x][y] = arr[nx][ny]
        up(arr, nx, ny, direction)
```



코딩 테스트 대비
핵심 유형 문제풀이

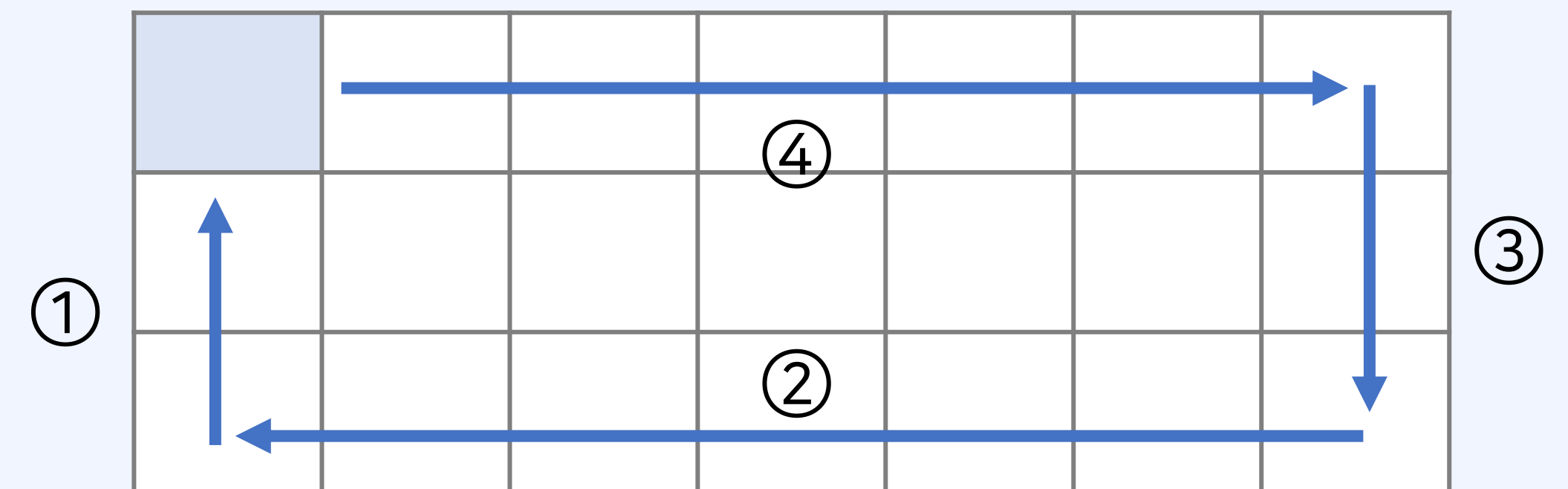
소스 코드 4)

코테 대비.

핵심 유형 문제풀이

공기청정기의 아래 칸 처리하기(D → R → U → L 순서로 처리)

```
def down(arr, x, y, direction):
    if direction == 'D':
        nx, ny = x + 1, y # 아래쪽 끝까지 도달한 경우
        if nx == r:
            direction = 'R'
    if direction == 'R':
        nx, ny = x, y + 1 # 오른쪽 끝까지 도달한 경우
        if ny == c:
            direction = 'U'
    if direction == 'U':
        nx, ny = x - 1, y # 위쪽 끝까지 도달한 경우
        if nx == down_position[0] - 1:
            direction = 'L'
    if direction == 'L':
        nx, ny = x, y - 1
    # 공기청정기로 돌아올 때까지
    if (nx, ny) != down_position:
        # (nx, ny) 위치에 있는 미세먼지를 (x, y) 위치로 이동
        arr[x][y] = arr[nx][ny]
        down(arr, nx, ny, direction)
```



코딩 테스트 대비 핵심 유형 문제풀이

소스 코드 5)

코테 대비.

핵심 유형 문제풀이

```
# 공기청정기 순환 함수
def cycle():
    global arr
    x, y = up_position
    up(arr, x - 1, y, 'U') # U → R → D → L 순서대로 처리
    arr[x][y + 1] = 0 # 바로 오른쪽 칸은 0
    x, y = down_position
    down(arr, x + 1, y, 'D') # D → R → U → L 순서대로 처리
    arr[x][y + 1] = 0 # 바로 오른쪽 칸은 0
```