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

혼자 힘으로 풀어보기

문제 제목: 이차원 배열과 연산

문제 난이도: ★★★☆☆

문제 유형: 시뮬레이션

추천 풀이 시간: 50분

문제 해결 아이디어

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 배열의 크기는 $R \times C$이며, 최대 100 X 100이다.
- **최대 100초(K)**까지만 연산이 수행되므로, 단순 구현으로 문제를 해결할 수 있다.
- 본 문제는 아래 두 개의 함수를 구현하여 해결할 수 있다.

1. 각 행(row)을 기준에 따라 정렬하는 함수
2. 특정한 행렬에 대하여 전치 행렬(transposed matrix)을 구하는 함수

문제 해결 아이디어

[정렬 수행]

- 원하는 기준에 따라서 정렬을 수행해야 한다.
- **정렬 기준**: ① 수의 등장 횟수가 커지는 순, ② 수가 커지는 순

1. [3, 1, 1]은 3이 1번, 1이 2번 등장하여 결과는 [3, 1, 1, 2]다.
2. [3, 1, 1, 2]는 2가 1번, 3이 1번, 1이 2번 등장하여 결과는 [2, 1, 3, 1, 1, 2]다.

- 각 행을 기준으로 정렬을 수행하면, 가장 큰 행을 기준으로 모든 행의 크기가 변경된다.
- 빈 곳에는 0을 채우고, 수를 정렬할 때는 0은 무시한다.

문제 해결 아이디어

- **정렬 기준**: ① 수의 등장 횟수가 커지는 순, ② 수가 커지는 순

```
# 하나의 행(row)에 대하여 정렬을 수행하는 함수
def sort_row(row):
    counter = dict()  # 원소의 개수를 세는 카운터(counter)
    for x in row:
        if x == 0: continue  # 0은 무시
        if x not in counter: counter[x] = 1
        else: counter[x] += 1
    # 정렬 기준: ① 수의 등장 횟수가 커지는 순, ② 수가 커지는 순
    sorted_counter = sorted(counter.items(), key=lambda x: (x[1], x[0]))
    # 정렬된 결과를 배열에 넣을 때는 (수, 등장 횟수)를 넣기
    result = []
    for val, cnt in sorted_counter:
        result += [val, cnt]
    return result
```

문제 해결 아이디어

[전치 행렬 이용하기]

- 필요에 따라서 열(column)을 기준으로 정렬을 수행해야 한다.
- 이는 행렬(matrix) A에 대하여 ① **전치 행렬을 계산**한 뒤에, ② 행(row)을 기준으로 정렬을 수행하고, ③ **다시 전치 행렬을 계산**한 것과 같다.

문제 해결 아이디어

[전치 행렬 이용하기]

- C 연산 = ① 전치 행렬로 변환 → ② R 연산 → ③ 전치 행렬로 변환

```
# 전치 행렬(transposed matrix)을 구하는 함수
def transpose(matrix):
    row_length = len(matrix)  # 행(row)의 길이
    column_length = len(matrix[0])  # 열(column)의 길이
    # 전치 행렬 만들기
    result = [[0] * row_length for i in range(column_length)]
    # 전치 행렬에 값 채워넣기
    for i in range(column_length):
        for j in range(row_length):
            result[i][j] = matrix[j][i]
    return result
```

이차원 배열과 연산: https://www.acmicpc.net/problem/17140

코딩 테스트 대비
핵심 유형 문제풀이

소스 코드

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


# 매초마다 배열에 대한 연산을 처리하는 함수
def process(matrix, operator):
    if operator == 'C':  # C 연산인 경우
        matrix = transpose(matrix)
    max_length = 0
    # 각 행을 기준으로 정렬 수행
    for i in range(len(matrix)):
        matrix[i] = sort_row(matrix[i])
        # 가장 긴 행 계산하기
        max_length = max(max_length, len(matrix[i]))
    # 가장 긴 행보다 짧은 행들은 0을 채우기
    for i in range(len(matrix)):
        gap = max_length - len(matrix[i])
        matrix[i] += [0] * gap
        # 길이가 100을 넘어가지 않도록 자르기
        matrix[i] = matrix[i][:100]
    if operator == 'C':  # C 연산인 경우
        matrix = transpose(matrix)
    return matrix
```

```python
# 3 X 3 배열(matrix) 및 r, c, k 입력받기
r, c, k = map(int, input().split())
matrix = []
for i in range(3):
    matrix.append(list(map(int, input().split())))

t = 0
while True:
    row_length = len(matrix)
    column_length = len(matrix[0])
    # 범위를 벗어나지 않으며
    if row_length >= r and column_length >= c:
        # matrix[r][c]의 값이 k일 때 종료
        if matrix[r - 1][c - 1] == k:
            print(t)
            break
    # 100초가 경과한 경우 종료
    if t == 100:
        print(-1)
        break
    # R 연산 혹은 C 연산 수행
    if row_length >= column_length:
        matrix = process(matrix, 'R')
    else:
        matrix = process(matrix, 'C')
    t += 1  # 1초 증가
```

톱니바퀴: https://www.acmicpc.net/problem/14891

혼자 힘으로 풀어보기

문제 제목: 톱니바퀴

문제 난이도: ★★★☆☆

문제 유형: 시뮬레이션

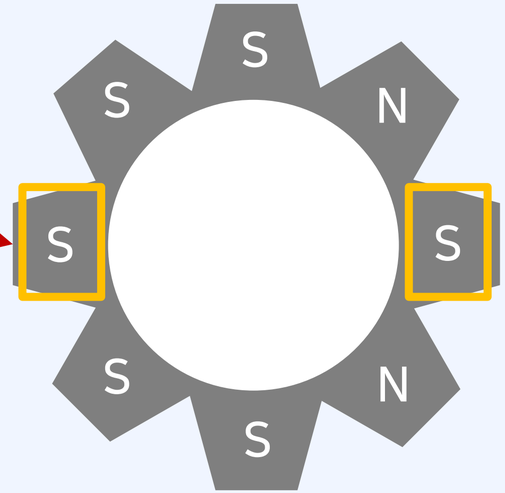
추천 풀이 시간: 50분

톱니바퀴: https://www.acmicpc.net/problem/14891

문제 해결 아이디어

- 문제의 요구사항 그대로 구현하면 되는 **시뮬레이션** 유형의 문제다.
- 톱니바퀴(gear)는 항상 4개이고, 각 톱니바퀴는 8개의 원소로 구성된다.
- 문제에서의 예시를 리스트로 표현하면 다음과 같다.

1. gears[0] = [1, 0, 1, 0, 1, 1, 1, 1]
2. gears[1] = [0, 1, 1, 1, 1, 1, 0, 1]
3. gears[2] = [1, 1, 0, 0, 1, 1, 1, 0]
4. gears[3] = [0, 0, 0, 0, 0, 0, 1, 0]



- 12시 방향부터 시계방향으로, (N극: 0, S극: 1)에 해당한다.
- 왼쪽 극은 인덱스(index) 6, 오른쪽 극은 인덱스(index) 2이다.

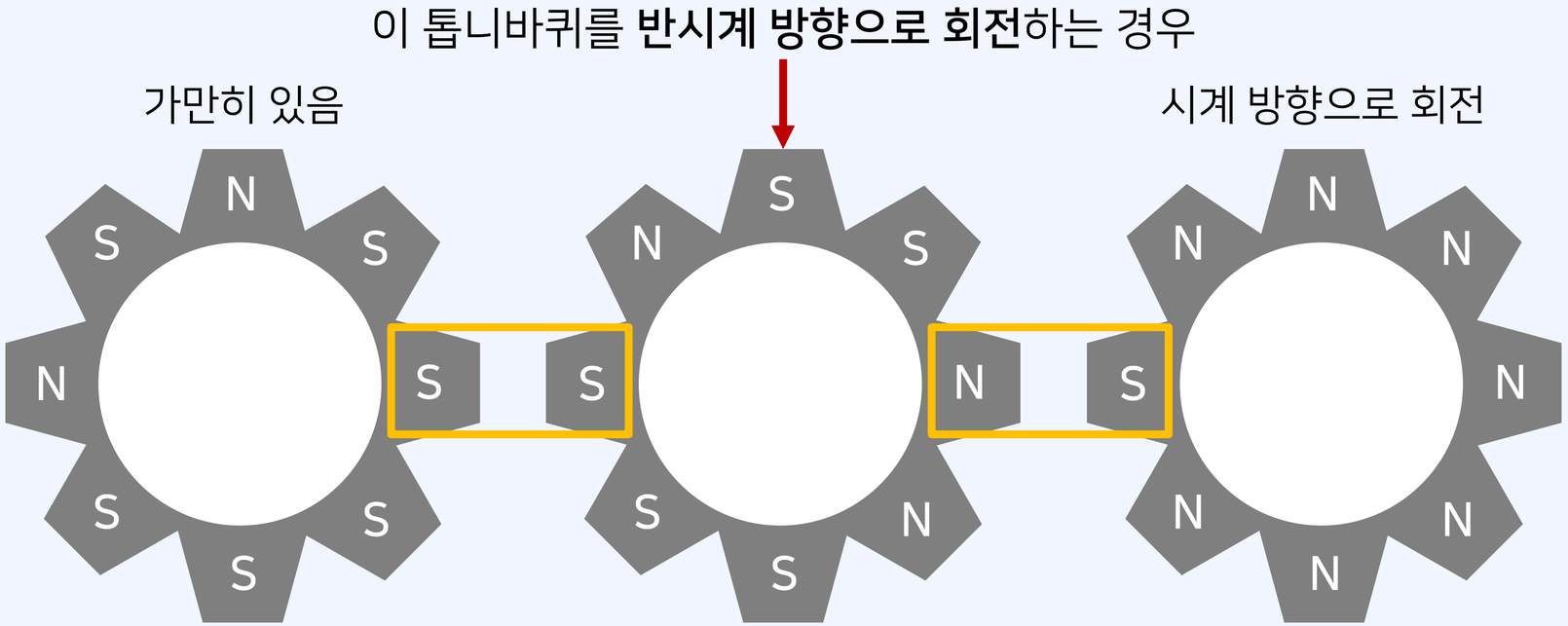
톱니바퀴: https://www.acmicpc.net/problem/14891

문제 해결 아이디어

- 특정한 **톱니바퀴를 회전**시킬 수 있다.

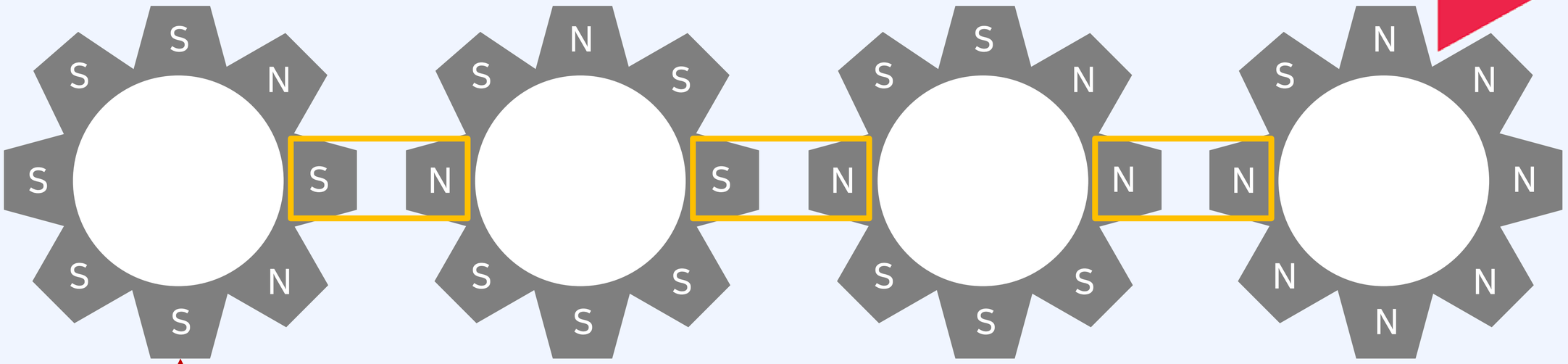
1) 맞닿은 극이 같으면, 인접한 톱니바퀴는 가만히 있는다.

2) 맞닿은 극이 서로 다르면, 인접한 톱니바퀴는 반대 방향으로 회전한다.



톱니바퀴: https://www.acmicpc.net/problem/14891

문제 해결 아이디어



이 톱니바퀴를
시계 방향으로 회전하는 경우

- 특정한 **톱니바퀴를 회전**시킴에 따라서 연쇄적으로 회전이 이루어집니다.

톱니바퀴: https://www.acmicpc.net/problem/14891

문제 해결 아이디어

- 특정한 **톱니바퀴를 회전**시키는 함수를 구현해야 합니다.
- 나머지 연산자(%)를 사용하여, 회전 기능을 구현할 수 있습니다.

```
# 특정한 톱니바퀴(index)를 시계 방향(R) 및 반시계 방향(L)으로 회전
def rotate(index, rotation):
    result = [None] * 8
    # 톱니바퀴를 시계 방향으로 회전시키는 경우
    if rotation == 'R':
        for i in range(8):
            result[(i + 1) % 8] = gears[index][i]
    # 톱니바퀴를 반시계 방향으로 회전시키는 경우
    if rotation == 'L':
        for i in range(8):
            result[(i - 1) % 8] = gears[index][i]
    gears[index] = result
```

톱니바퀴: https://www.acmicpc.net/problem/14891

소스 코드 1)


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

# 특정한 인덱스(index)의 톱니바퀴를 회전시키는 함수
# 연쇄 방향(direction): 오른쪽(R) 혹은 왼쪽(L)으로 인접 톱니바퀴를 회전
def dfs(index, direction, rotation, move=True):
    # 오른쪽에 있는 톱니바퀴를 회전시키기
    if index <= 2 and direction == 'R':
        # 현재 기어의 오른쪽 극과 오른쪽 기어의 왼쪽 극이 다르다면
        if gears[index][2] != gears[index + 1][6]:
            next_rotation = 'R'
            if rotation == 'R': next_rotation = 'L'
            dfs(index + 1, 'R', next_rotation)
    # 왼쪽에 있는 톱니바퀴를 회전시키기
    elif index >= 1 and direction == 'L':
        # 현재 기어의 왼쪽 극과 왼쪽 기어의 오른쪽 극이 다르다면
        if gears[index][6] != gears[index - 1][2]:
            next_rotation = 'R'
            if rotation == 'R': next_rotation = 'L'
            dfs(index - 1, 'L', next_rotation)
    # 최종적으로 내 톱니바퀴를 회전시키기
    if move: rotate(index, rotation)
```

```python
# 4개의 톱니바퀴(gear) 정보 입력
gears = [[] for _ in range(4)]
for i in range(4):
    gear = input().strip()
    for j in range(len(gear)):
        gears[i].append(int(gear[j]))


# 한 톱니바퀴(index)를 시계 방향(R) 혹은 반시계 방향(L)으로 회전
def rotate(index, rotation):
    result = [None] * 8
    # 톱니바퀴를 시계 방향으로 회전시키는 경우
    if rotation == 'R':
        for i in range(8):
            result[(i + 1) % 8] = gears[index][i]
    # 톱니바퀴를 반시계 방향으로 회전시키는 경우
    if rotation == 'L':
        for i in range(8):
            result[(i - 1) % 8] = gears[index][i]
    gears[index] = result
```

톱니바퀴: https://www.acmicpc.net/problem/14891

소스 코드 2)


```
k = int(input())  # 회전할 톱니바퀴의 수
for i in range(k):
    index, rotation = map(int, input().split())
    index -= 1
    # 시계 방향: R(1), 반시계 방향: L(-1)
    if rotation == 1: rotation = 'R'
    else: rotation = 'L'
    # 시작 위치에서부터 오른쪽(R), 왼쪽(L)으로 인접한 톱니바퀴 회전
    dfs(index, 'R', rotation, move=False)
    dfs(index, 'L', rotation)  # 시작 톱니바퀴는 1번만 회전

# 최종 점수(score) 계산
result = 0
if gears[0][0] == 1: result += 1
if gears[1][0] == 1: result += 2
if gears[2][0] == 1: result += 4
if gears[3][0] == 1:
    result += 8
print(result)
```