3.2.1

문제1, 문제2

23.10.23 송이

문제1 이해하기

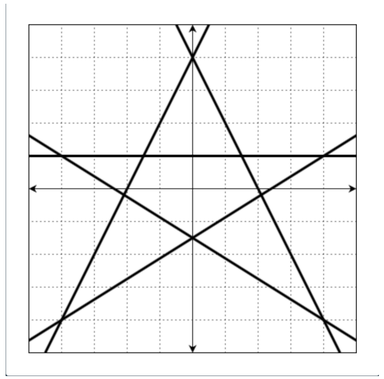
교점에 별 만들기

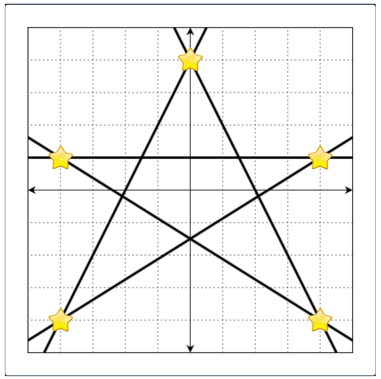
`Ax + By + C = 0` 으로 표현할 수 있는 `n` 개의 직선이 주어질 때, 이 직선의 교점 중 정수 좌표에 별을 그리려 합니다.

예를 들어, 다음과 같은 직선 5개를

- `2x - y + 4 = 0`
- `-2x - y + 4 = 0`
- `-y + 1 = 0`
- `5x - 8y - 12 = 0`
- `5x + 8y + 12 = 0`

좌표 평면 위에 그리면 아래 그림과 같습니다.





```
".....*....."
"....*.*...."
"...*...*..."
"..*.....*.."
".*.......*."
"..*.....*.."
"...*...*..."
"....*.*...."
".....*....."
```

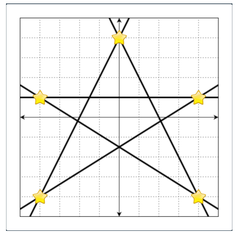
```
".....*....."
"....*.*...."
"...*...*..."
"..*.....*.."
"*.......*."
"..*.....*.."
"...*...*..."
"....*.*...."
".....*....."
```

제한사항

- line의 세로(행) 길이는 2 이상 1,000 이하인 자연수입니다.
  - line의 가로(열) 길이는 3입니다.
  - line의 각 원소는 [A, B, C] 형태입니다.
  - A, B, C는 -100,000 이상 100,000 이하인 정수입니다.
  - 무수히 많은 교점이 생기는 직선 쌍은 주어지지 않습니다.
  - A = 0이면서 B = 0인 경우는 주어지지 않습니다.
- 정답은 1,000 * 1,000 크기 이내에서 표현됩니다.
- 별이 한 개 이상 그려지는 입력만 주어집니다.

단계 나누기

1. 직선에서 교점 구하기 (정수 교점만)



pos라고 하는 배열에 [x,y] 형태로 저장하기

2. 교점을 모두 표현할 수 있는 최소한의 사각형 크기 구하고, 기본틀 만들기

3. 기본틀에서 교점에만 * 로 바꾸기 ★★

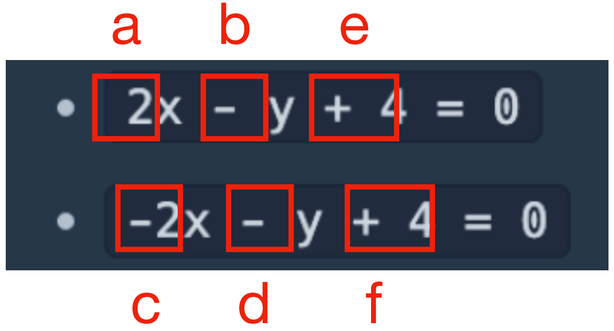
STEP1
정수 교점만 구하기

전체코드

```
def solution(line):
    pos,answer = [],[]
    n = len(line)

    x_min = y_min = int(1e15)
    x_max = y_max = -int(1e15)

    for i in range(n):
        a,b,e = line[i]
        for j in range(i+1,n):
            c,d,f = line[j]
            if a*d == b*c:  #두 직선이 평행인 경우
                continue
            x = (b*f-e*d) / (a*d-b*c)
            y = (e*c-a*f) / (a*d-b*c)

            if x==int(x) and y==int(y): #교점이 정수인 경우
                x = int(x)
                y = int(y)
                pos.append([x,y])
                if x_min > x: x_min = x
                if y_min > y: y_min = y
                if x_max < x: x_max = x
                if y_max < y: y_max = y
```

```
pos,answer = [],[]      pos는 각 교점[x,y]를 저장하는 공간
n = len(line)           answer는 최종 정답
```

```
for i in range(n):
    a,b,e = line[i]
    for j in range(i+1,n):
        c,d,f = line[j]
        if a*d == b*c:  #두 직선이 평행인 경우
            continue
        x = (b*f-e*d) / (a*d-b*c)
        y = (e*c-a*f) / (a*d-b*c)
```




```
if x==int(x) and y==int(y): #교점이 정수인 경우
            x = int(x)
            y = int(y)
            pos.append([x,y])
```

참고 사항

$Ax + By + E = 0$

$Cx + Dy + F = 0$

두 직선의 교점이 유일하게 존재할 경우, 그 교점은 다음과 같습니다.

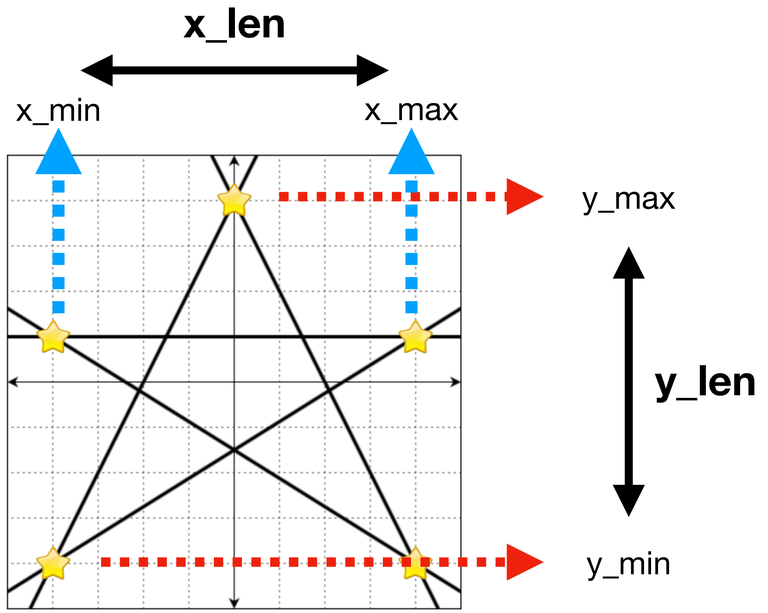
$$x = \frac{BF - ED}{AD - BC} \qquad y = \frac{EC - AF}{AD - BC}$$

또, AD - BC = 0인 경우 두 직선은 평행 또는 일치합니다.

STEP2

교점을 모두 표현할 수 있는 최소한의 사각형 구하고 기본틀 만들기

```
if x==int(x) and y==int(y): #교점이 정수인 경우
    x = int(x)
    y = int(y)
    pos.append([x,y])
    if x_min > x: x_min = x
    if y_min > y: y_min = y
    if x_max < x: x_max = x
    if y_max < y: y_max = y
```



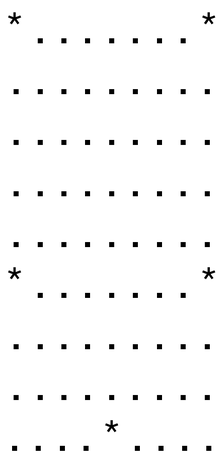

```
x_len = x_max - x_min+1
y_len = y_max - y_min +1
coord = [['.'] * x_len for _ in range(y_len)]
```

STEP3

교점에만 *로 바꿔주기

책에서 처음에 설명한 방법

```
for start_x,start_y in pos:
    nx = start_x + abs(x_min) if x_min<0 else start_x - x_min
    ny = start_y + abs(y_min) if y_min<0 else start_y - y_min
    coord[ny][nx] = '*'


for result in coord:
    answer.append(''.join(result))
return answer[::-1]
```



책에서 두번째로 설명한 방법

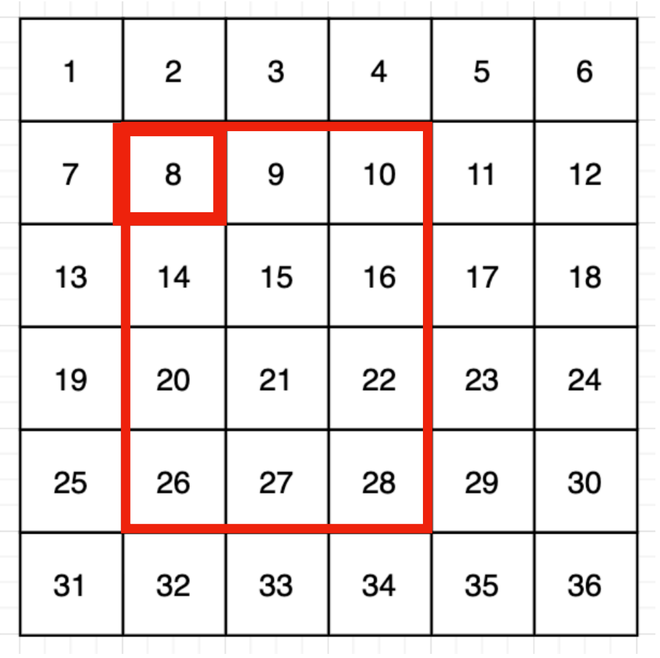
```
pos = sorted(pos,key=lambda i : -i[1])
for x,y in pos:
ny = y_max - y
nx = x - x_min
coord[ny][nx] = '*'
```

가장 작은 좌표부터
시작해서 뒤집을 필요가 없음.

```
answer = [['.'] * (x_max - x_min + 1) for _ in range(y_max - y_min + 1)]
    for x, y in points:
        answer[y_max - y][x - x_min] = '*'  # 교점에 별 만들기
    return list(map(''.join, answer))
```

문제2 이해하기

행렬 테두리 회전하기

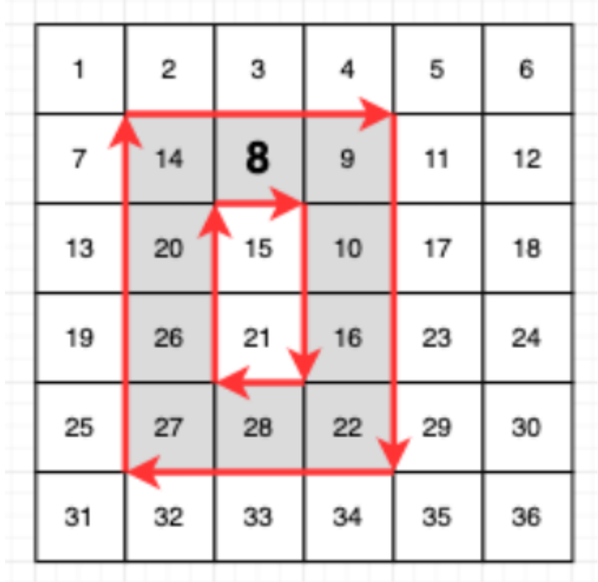


rows x columns 크기인 행렬이 있습니다. 행렬에는 1부터 rows x columns까지의 숫자가 한 줄씩 순서대로 적혀있습니다. 이 행렬에서 직사각형 모양의 범위를 여러 번 선택해, 테두리 부분에 있는 숫자들을 시계방향으로 회전시키려 합니다. 각 회전은 (x1, y1, x2, y2)인 정수 4개로 표현하며, 그 의미는 다음과 같습니다.

- x1 행 y1 열부터 x2 행 y2 열까지의 영역에 해당하는 직사각형에서 테두리에 있는 숫자들을 한 칸씩 시계방향으로 회전합니다.

rows	columns	queries	result
6	6	[[2,2,5,4],[3,3,6,6],[5,1,6,3]]	[8, 10, 25]
3	3	[[1,1,2,2],[1,2,2,3],[2,1,3,2],[2,2,3,3]]	[1, 1, 5, 3]
100	97	[[1,1,100,97]]	[1]

2행2열 -> 5행 4열 사이의 값들을 시계방향으로 회전시키기



단계 나누기

1. 1씩 증가하는 행렬 생성하기
2. 회전해야 할 위치들의 값을 받아오기 (queries에서 하나씩)
3. 행렬을 시계 방향으로 회전시키기 ★★★
4. 3번 과정에서의 최소값을 answer 배열에 추가하기

STEP1 & STEP2

1씩 증가하는 행렬 생성하기 , queries에서 값을 하나씩 가져오기

```
matrix = [[(i) * columns + (j+1) for j in range(columns)] for i in range(rows)]
```

```
[1, 2, 3, 4, 5, 6]
[7, 8, 9, 10, 11, 12]
[13, 14, 15, 16, 17, 18]
[19, 20, 21, 22, 23, 24]
[25, 26, 27, 28, 29, 30]
[31, 32, 33, 34, 35, 36]
```

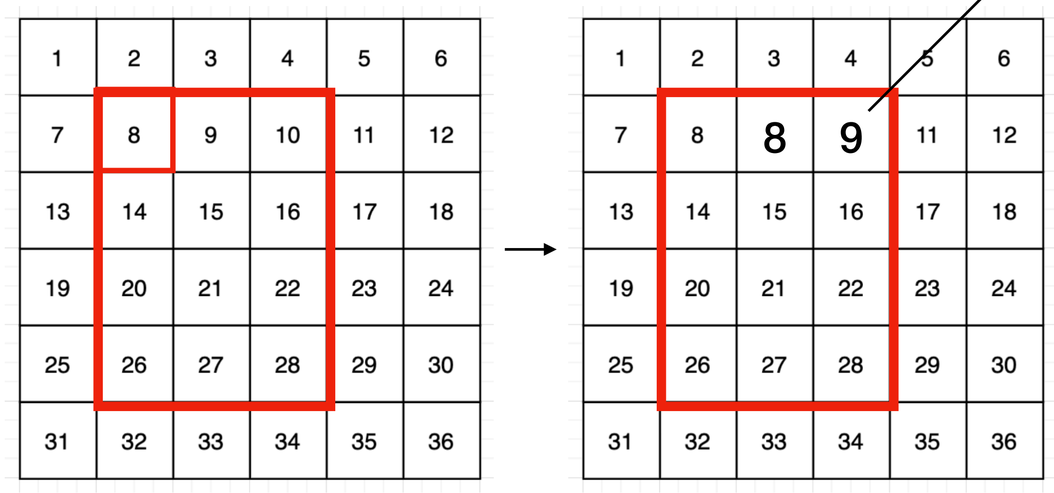
```
for x1,y1,x2,y2 in queries:
    result.append(rotate(x1-1,y1-1,x2-1,y2-1,matrix))
```

값 자체는 1부터 시작하나
인덱스는 0부터 시작하니 모든 위치값에서 -1씩 해주어야 한다.

STEP3

행렬을 시계 방향으로 회전시키기

시계 방향으로 돌릴때의 문제점

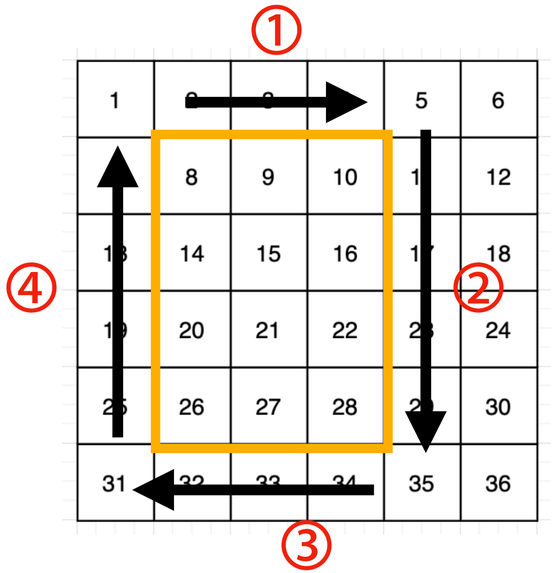


값이 하나씩 밀리면서 원래 있던 값 10이 사라짐.

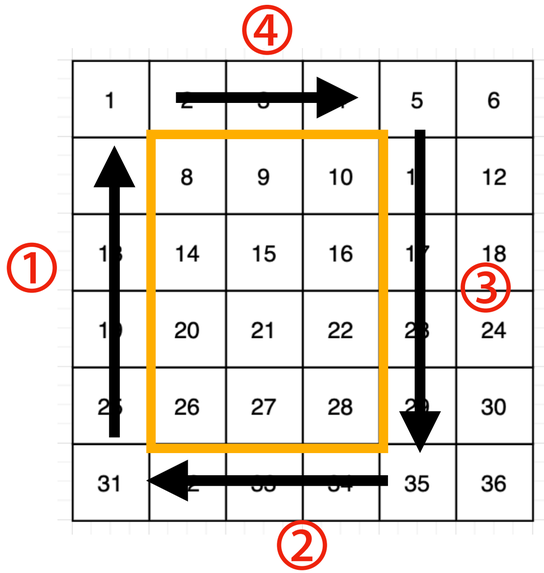
STEP3

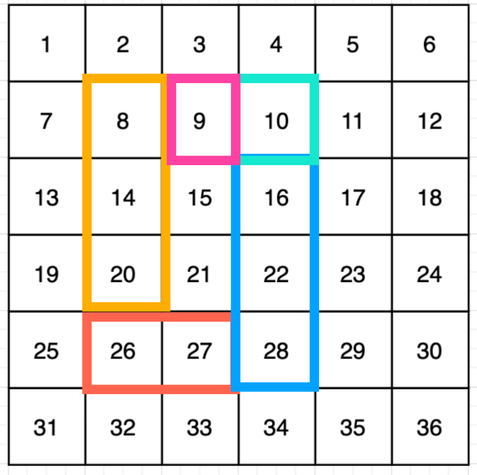
행렬을 시계 방향으로 회전시키기

원래 방식



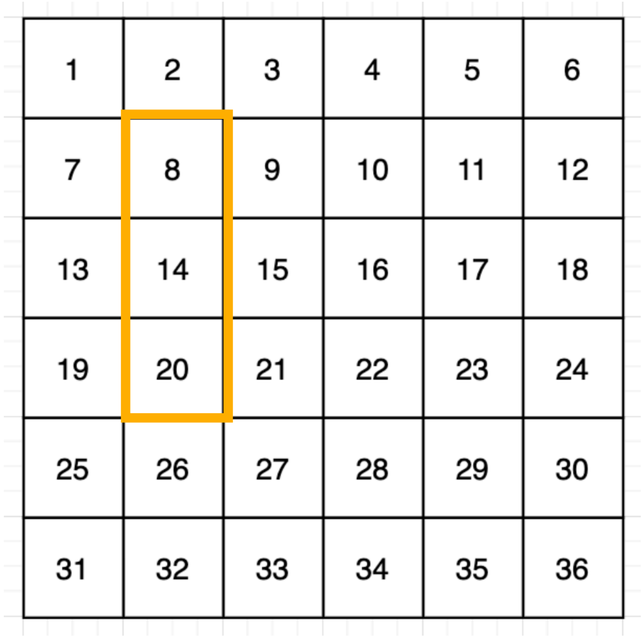
다른 방식





STEP3

행렬을 시계 방향으로 회전시키기



1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

```
for k in range(x1,x2):
    matrix[k][y1] = matrix[k+1][y1]
    min_value = min(min_value,matrix[k+1][y1])
```

x1 = 2, x2=5 k = 2,3,4

k=2일때 matrix[2][2] = matrix[3][2] 8-> 14

k=3일때 matrix[3][2] = matrix[4][2] 14 -> 20

k=4일때 matrix[4][2] = matrix[5][2] 20 -> 26

STEP4

3번 과정에서 최소값 찾기

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

first = matrix[x1][y1]

min_value = first

for k in range(x1,x2):

    matrix[k][y1] = matrix[k+1][y1]

    min_value = min(min_value,matrix[k+1][y1])

matrix[x1][y1+1] = first

return min_value -> 8

최종답 : [8,10,25]