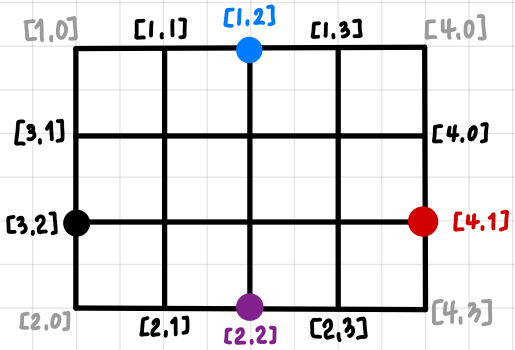
예제 입력

```
4 3   → 가로 세로
3     → 상점 개수 (n_store)
1 2
2 1   → 0 ~ (n_store-1)번째 상점 위치
3 2
4 1   → 동근 위치
```



i) 경계에 있는 각 점을 시계 방향 순서로 2차원 배열에 저장

arr[14][2] 생성

⇒ [[1,0], [1,1], [1,2], [1,3],
[4,0], [4,1], [4,2], [4,3],
[2,3], [2,2], [2,1], [2,0],
[3,2], [3,1]]

ii) 각 상점의 시계 방향 & 반시계 방향 기준 위치를 배열에 저장

clockwise = [2, 10, 12]

anticlockwise = [12, 4, 2]

⇒ 14-2, 14-10, 14-12

iii) 동근이의 시계 방향 & 반시계 방향 기준 위치를 변수에 저장

dong_cwIndex = 5

dong_acwIndex = 9

iv) 동근 - 상점 최단거리 구하기

if) [1,0] 기준 시계 방향으로 돌았을 때, 동근이가 상점보다 멀리있는 경우

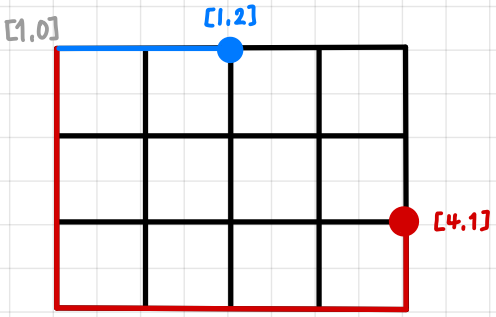
⇒ dong_cwIndex > clockwise[i]

ex. 0번 상점 [1, 2]

1) 시계 방향으로 돌 때 (cw_dong에 저장)

= 0번 상점 시계방향 위치 + 동근이 반시계 방향위치

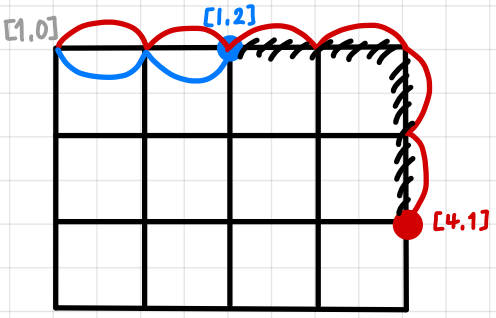
= clockwise[0] + dong_acwIndex



2) 반시계 방향으로 돌 때 (acw_dong에 저장)

= 동근이 시계 방향위치 - 0번 상점 시계방향 위치

= dong_cwIndex - clockwise[0]



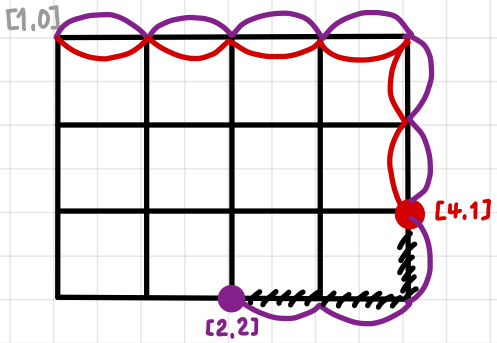
⇒ result += Math.min(cw_dong, acw_dong)

else) ex. 1번상점 [2,2]

1) 시계 방향으로 돌 때 (cw_dong에 저장)

= 1번 상점 시계 방향위치 - 동근이 시계 방향위치

= clockwise[1] - dong_cwIndex



2) 반시계 방향으로 돌 때 (acw_dong에 저장)

= 동근이 시계 방향 위치 + 1번상점 반시계 방향위치

= dong_cwIndex + anticlockwise[1]