공격 상황 시뮬레이션

CCW(CounterClockWise) 알고리즘

두 벡터가 주어졌을 때, 두 벡터사이의 위치 관계를 파악할 때 사용

> 두 벡터라는 말은 곧 세 점이라는 것과 같습니다.
>
> $$p_1, p_2, p_3$$
>
> 을 두 벡터
>
> $$\vec{A} = (p_2 - p_1), \vec{B} = (p_3 - p_1)$$

> 로 변환 시켜서 외적을 계산하면
>
> $$(x_1y_2 + x_2y_3 + x_3y_1) - (y_1x_2 + y_2x_3 + y_3x_1)$$
>
> 이 됩니다.

구현

> 그렇다면 어떻게 저 알고리즘을 사용해서 문제를 해결할 수 있을까요?
>
> > 간단하게 생각하면 N개의 점에 대해서 다른 N개의 점과의 위치 관계를 조사하면 O(N^2)으로 손쉽게 풀 수 있습니다.

> > 근데 N<300000 으로 O(N^2)으로는 100점을 맞을 수 없습니다. 결국 계산을 줄이기 위해서 이전에 사용했던 정보를 재사용하는 법을 찾아봅시다.

> > 문제 조건을 삼각형 내부에 점이 있는지로 보지말고, 벡터간의 위치관계를 묻는 질문으로 바꿔 생각해 봅시다.

> > A점에서 각 공격수 까지의 벡터들은 골대 사이와의 각도를 기준으로 정렬 할 수 있습니다. O(nlogn) 그렇다면 가장 각도가 작은 친구부터 시작하면 B점 조건이 없다고 생각할 때, n - 1개가 정답이 되겠죠.

> > 하지만 B점에서 시작하는 벡터들도 고려해 주어야 합니다. 같은 방식으로 정렬을 하고, 자신의 등수(rank)를 기억해 봅시다.

> > 그 후 다시 처음부터 시작해서 A점에서 정렬했던 값들을 기준으로 모든 점들을 검사하면 됩니다. 각 점들의 입장에서 이전에 나왔던 점들을 A기준으로 자신이 포함하고 있으므로, 이제 B기준에서 봤을 때, 자신의 각도가 이전까지 나왔던 모든 점보다 더 작았다면 이점이 아무것도 포함하지 않는 점이 될 수 있습니다. (유효한 공격수)

> > 쉽게 생각해서 자신의 B기준 랭크가 이전까지의 B로 제한되는 벡터를 탈출 할 수 있는지만 보면 됩니다. 어차피 A기준으로는 정렬된 상태라 이미 전부 탈출 할 수 없어서 B만 보면 됩니다. 결국 관리해야 되는건 각각 점들의 등수 뿐입니다.
>
> > 구현은 간단합니다. (생략)