문제정의.

처음에 오름차순으로 카드를 선택하고, 이후 내림차순으로 카드를 선택.
오름차순 일때는 현재 제시된 카드가 마지막으로 선택한 카드보다 크고, 전에 선택하지 않은 카드라면 take 가능.
내림차순 일때는 현재 제시된 카드가 마지막으로 선택한 카드보다 작고, 전에 선택하지 않았던 카드라면 take 가능.

따라서 오름차순 / 내림차순 두 개의 phase로 나누어, 기존에 선택한 card set과 중복되지 않되 각 phase 조건에 맞게 카드를 take.

아이디어.

첫 번째 카드부터 마지막 카드까지, increasing에서 decreasing으로 바뀌는 기준이 되는 카드를 i번째 카드 (1 <= i <= N)이라 하자.
모든 i에 대해 첫 번째 카드부터 i번째 카드까지 increasing하며 take하는 카드의 수와, i+1번째 카드부터 N번째 카드까지 decreasing하며 take하는 카드의 수를 구한다. 그렇게 구한 increasing[i]와 decreasing[i]를 더해 'i번째에서 switch했을 때의 선택할 수 있는 카드 수'를 나타내는 score[i]를 구한다. score[i] 중에서 가장 큰 값이 곧 maximum score이다.

해결방법.

1. N개의 카드 sequence를 받아 각 카드를 순서대로 컨테이너 cards에 저장한다.
2. 모든 i(1 <= i <= N)에 대해, 선택한 카드들을 넣을 수 있는 2차원 컨테이너 picekd를 선언한다.
    a. 첫 번째 카드부터 i번째 카드까지 cards[i-1] < cards[i]이고, cards[i]가 picked[i]에 없다면 cards[i]를 picked[i]에 넣는다.
    b. i번째카드부터 N번째 카드까지, cards[i] > cards[i+1]이고, cards[i]가 picked[i]에 없다면 cards[i]를 picked[i]에 넣는다.
3. picked[i] (1 <= i <= N) 중 크기가 가장 큰 picked[i] -> maximum score.

시간복잡도.

주어진 카드의 개수 N에 대해, 2번 과정에서 각 카드에 대한 경우를 확인 -> O(N^2),cards[i]가 picked에 있는지 확인 -> O(n)
따라서, O(N^3).