# C6. Card Game

21400707 정희석, 21700477 윤다은

## 1. Problem Analysis

- 0~10^9의 숫자가 적힌 n개의 카드를 순차적으로 선택할 수 있다.
- 뽑은 카드를 기준으로 그 이전에 뽑은 카드는 오름차순 정렬이 돼야하고, 이후에 뽑은 카드는 내림차순 정렬이 되도록 카드를 뽑아야 한다. 이 때, 가장 많이 뽑을 수 있는 카드의 수를 구하라.

## 2. Solution

### n개의 카드(cards) 중 어떤 카드를 중심으로 삼을지 고르는 문제

**전체 흐름**

1) 중심카드의 인덱스를 i라고 할 때, 0에서 n-1까지 i를 늘리면서 i를 기준으로 [0, i]까지의 최장 증가 수열(asc_v)을 만들고, 그 길이를 구하여 벡터(asc_count_v)의 i번째 원소에 저장한다.
2) 이후 다시 0에서 n까지 i를 늘리면서 [n-1, n-1-i]까지 최장 감소 수열(des_v)을 만들고, 그 길이를 구하여 벡터(des_count_v)의 i번째 원소에 저장한다.
3) 수열을 찾는 과정이 종료되면, 두 벡터의 합 중 가장 큰 원소 -1을 출력한다(cards[i]가 두번 선택되었으므로).

**최장 증가 수열 만들기**

조건1) asc_v가 비었을 때 (i = 0)
→ asc_v에 cards[i]를 넣는다.
조건2) asc_v의 맨 마지막 원소가 cards[i]보다 작은 경우
→ asc_v의 끝에 cards[i]를 추가한다.
조건3) asc_v의 맨 마지막 원소가 cards[i]보다 큰 경우
→ j를 size(asc_v)-2에서 0까지 1씩 감소시킨다. 만약 asc_v[j] 원소보다 cards[i]가 더 크면, asc_v[j+1]의 값을 cards[i]로 갱신하고 반복을 탈출한다.
→ 위 조건이 j의 범위에서 만족하지 않으면, asc_v[0]의 값을 cards[i]로 갱신한다.

**최장 감소 수열 만들기**

→ cards의 마지막 원소부터 첫 원소까지 조회하며, 최장 증가 수열과 마찬가지 방법으로 수열을 만든다.

**e.g.**

cards = [1, 3, 2]

| | i = 0 | i = 1 | i = 2 |
|---|---|---|---|
| **asc_v** | {1} (조건1) | {1, 3} (조건2) | {1, 2} (조건3) |
| **asc_count_v** | {1} | {1, 2} | {1, 2, 2} |
| **des_v (reverse)** | {2} | {3, 2} | {3, 1} |
| **des_count_v(reverse)** | {1} | {2, 1} | {2, 2, 1} |

→ asc_count_v + reverse(des_count_v) - 1 = {1, 2, 2} + {2, 2, 1} - 1 = {2, 3, 2}
→ 따라서 3이 중심카드가 되고 최대 카드 개수는 3이 된다.

## 3. Solution Analysis (N : 전체 카드 개수)

1) 시간 복잡도 : O(N^2) → N (입력시간) + N^2 (asc_count_list, des_count_list 만드는 시간, 최악의 경우 N개의 정보를 만드는데 앞선 N개의 정보를 다봐야 함) + N (asc_count_list, des_count_list 합치는 시간)
2) 공간 복잡도 : O(N) → N (cards) + N (asc_count_list, des_count_list)

## 4. Discussion

최장 증가(감소) 수열을 만들 때, i번째 원소 이전의 모든 카드를 검사하는 브루트포스 방법과 비교하여, 현재 방법은 불필요한 비교를 줄여 시간의 효율성을 개선시켰다.