

# Algorithm 4월 2주차

sort

# [프로그래머스] 인사고과

- 각 사원의 점수가 담긴 배열 `scores`가 주어짐.
- `scores[i][0]`은 근무 태도 점수, `scores[i][1]`는 동료 평가 점수
- 두 점수의 합의 크기에 따라 석차가 정해짐
- 완호의 점수는 `scores[0]`에 담김.
- 완호의 석차는 몇 등인지 리턴한다. 인센티브 못 받으면 -1을 리턴

## 인사고과

### 문제 설명

완호네 회사는 연말마다 1년 간의 인사고과에 따라 인센티브를 지급합니다. 각 직원마다 근무 태도 점수와 동료 평가 점수가 기록되어 있는데 만약 어떤 직원이 다른 임의의 직원보다 두 점수가 모두 낮은 경우가 한 번이라도 있다면 그 직원은 인센티브를 받지 못합니다. 그렇지 않은 직원들에 대해서는 두 점수의 합의 높은 순으로 석차를 내어 석차에 따라 인센티브가 차등 지급됩니다. 이때, 두 점수의 합의 동일한 직원들은 동석차이며, 동석차의 수만큼 다음 석차는 건너 뛩니다. 예를 들어 점수의 합의 가장 큰 직원이 2명이라면 1등이 2명이고 2등 없이 다음 석차는 3등부터입니다.

각 직원의 근무 태도 점수와 동료 평가 점수 목록 `scores` 이 주어졌을 때, 완호의 석차를 `return` 하도록 `solution` 함수를 완성해주세요.

# [프로그래머스] 시도 - 그냥 조건문 + 반복문

- 근무태도만 별도의 배열에 저장
- 동료평가만 별도의 배열에 저장
- 완호가 최악이면 return -1
- 그게 아니면 내림차순으로 정렬하여 석차를 리턴함.

```
def solution(scores):  
    attitude = [scores[i][0] for i in range(len(scores))]  
    colleague = [scores[i][1] for i in range(len(scores))]  
    if min(attitude) != max(attitude) or min(colleague) != max(colleague):  
        if scores[0][0] == min(attitude) and scores[0][1] == min(colleague):  
            return -1  
    total = [(scores[i][0] + scores[i][1]) for i in range(len(scores))]  
    value = total[0]  
    total.sort(reverse=True)  
    for i in range(len(total)):  
        if total[i] == value:  
            return i + 1  
    return 0
```

- 실패하는 케이스 속출

테스트 4	통과 (0.02ms, 10MB)
테스트 5	통과 (0.02ms, 10.3MB)
테스트 6	실패 (0.01ms, 10.1MB)
테스트 7	통과 (0.02ms, 10MB)
테스트 8	실패 (0.02ms, 10.1MB)
테스트 9	실패 (0.04ms, 10.2MB)

테스트 11	실패 (0.17ms, 10.1MB)
테스트 12	통과 (0.18ms, 10.1MB)
테스트 13	실패 (0.34ms, 10.4MB)
테스트 14	실패 (0.41ms, 10.4MB)
테스트 15	실패 (2.20ms, 10.9MB)
테스트 16	실패 (2.00ms, 10.8MB)

# [프로그래머스] 인사고과 - 문제에서 문제

- 문제를 자세히 읽어보면 어떤 사원이 임의의 사원보다 점수가 모두 낮은 경우가 있다면 그 사원은 인센티브를 받지 못한다.

- 꼴지가 인센티브를 못 받는 게 아님.

- [1, 6], [4, 4], [4, 5], [6, 3] 이 경우 2번째 사원은 꼴지는 아니지만 인센티브를 받지 못한다. 완호는 석차 3등
- [2, 3], [5, 2], [7, 1], [12, 17] 이 경우 1, 2, 3번째 사원은 인센티브를 받지 못한다.

완호네 회사는 연말마다 1년 간의 인사고과에 따라 인센티브를 지급합니다. 각 사원마다 근무 태도 점수와 동료 평가 점수가 기록되어 있는데 만약 어떤 사원이 다른 임의의 사원보다 두 점수가 모두 낮은 경우가 한 번이라도 있다면 그 사원은 인센티브를 받지 못합니다. 그렇지 않은 사원들에 대해서는 두 점수의 합이 높은 순으로 석차를 내어 석

# [프로그래머스] 풀이법 - 정렬을 두 번

- 정렬을 내림차순으로 한 번, 오름차순으로 한 번. 총 두 번 실행
- 예를 들어 다음과 같은 배열이 주어지면
- [[4, 8], [7, 9], [5, 2], [7, 3], [7, 2], [4, 6], [9, 3], [8, 6], [8, 2]]
- 정렬하고 나면
- [[9, 3], [8, 2], [8, 6], [7, 2], [7, 3], [7, 9], [5, 2], [4, 6], [4, 8]]
- 오름차순으로 정렬했음에도 불구하고 앞서 나온 점수들보다 더 작으면 인센티브에서 제외된다.
- (i) 첫번째 사람의 동료평가 3점이 기준이 됨 -> 두번째 사람 바로 제외, 세번째 사람은 제외 안 되고, 6점이 기준치로 새로 기록됨
- (ii) 기준치 6점보다 작은 네번째 다섯 번째 사람은 모두 제외. 여섯 번째 사람은 제외되지 않고 9점이 새로운 기준이 됨.
- 이런 식으로 계속 진행..

# [프로그래머스] 풀이법 - 정렬 두 번

- 정렬을 근무 태도 점수 기준 내림차순으로 정렬, 동시에 동료 평가 점수 기준 오름차순으로 정렬
- 파이썬 람다식을 활용하여 정렬
- 동료 평가 점수가 더 크다면 제외되지 않음
- Total 점수가 더 크다면 석차를 늘려야 한다.

```
1  def solution(scores):
2      answer = 1
3      WANHO_ATTJ = scores[0][0]
4      WANHO_COLL = scores[0][1]
5      WANHO_TOTAL = sum(scores[0])
6
7      scores.sort(key=lambda x: (-x[0], x[1]))
8      filtering = 0
9      for attitude, colleague in scores:
10         if WANHO_ATTJ < attitude and WANHO_COLL < colleague:
11             return -1
12         if filtering <= colleague:
13             if WANHO_TOTAL < attitude + colleague:
14                 answer += 1
15                 filtering = colleague
16
17     return answer
```