Algorithem 4월 2주차

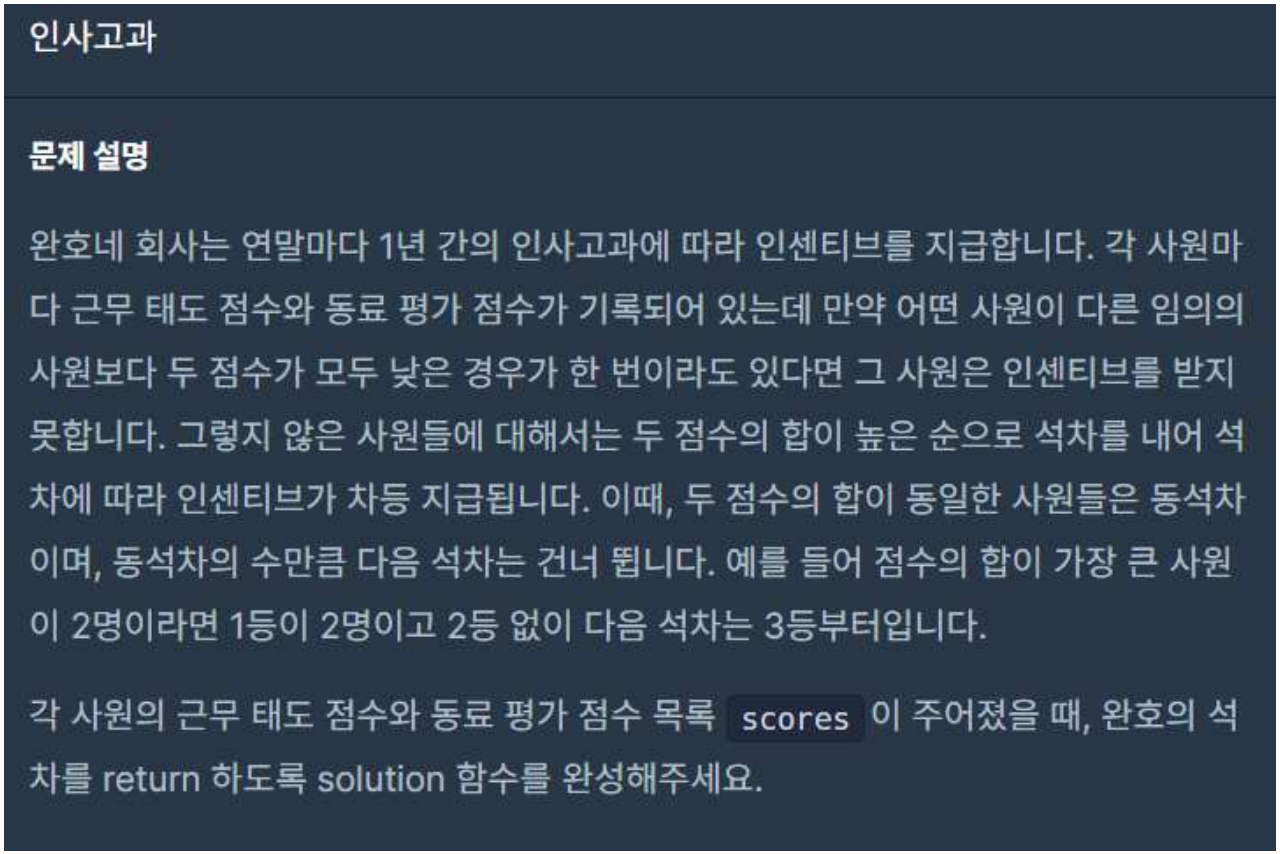
Sort, Brute force

Table of content

Sort

[프로그래머스] 인사고과

- 각 사원의 점수가 담긴 배열 scores가 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 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 석차에 따라 인센티브가 차등 지급됩니다. 이때, 두 점수의 합이 동일한 사원들은 동석차이며, 동석차의 수만큼 다음 석차는 건너 뜁니다. 예를 들어 점수의 합이 가장 큰 사원이 2명이라면 1등이 2명이고 2등 없이 다음 석차는 3등부터입니다.

각 사원의 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.

```python
def solution(scores):
    attitude = [scores[i][0] for i in range(len(scores))]
    collegue = [scores[i][1] for i in range(len(scores))]
    if min(attitude) != max(attitude) or min(collegue) != max(collegue):
        if scores[0][0] == min(attitude) and scores[0][1] == min(collegue):
            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.

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

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

[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

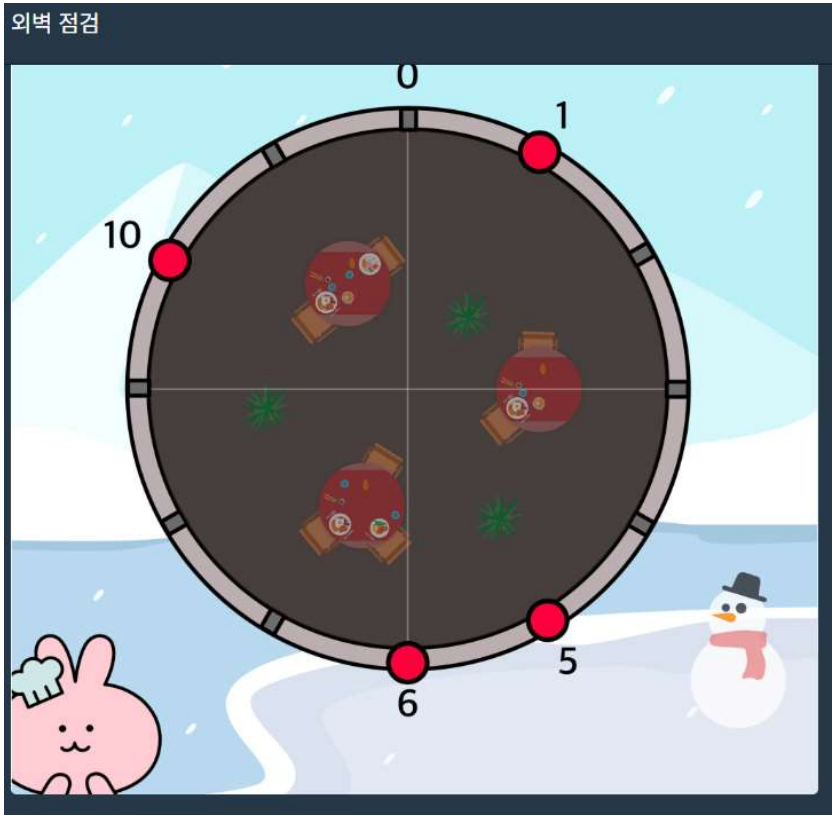
- 정렬을 근무 태도 점수 기준 내림차순으로 정렬, 동시에 동료 평가 점수 기준 오름차순으로 정렬
- 파이썬 람다식을 활용하여 정렬
- 동료 평가 점수가 더 크다면 제외되지 않음
- Total 점수가 더 크다면 석차를 늘려야 한다.

```python
def solution(scores):
    answer = 1
    WANHO_ATTI = scores[0][0]
    WANHO_COLL = scores[0][1]
    WANHO_TOTAL = sum(scores[0])

    scores.sort(key=lambda x: (-x[0], x[1]))
    filtering = 0
    for attitude, collegue in scores:
        if WANHO_ATTI < attitude and WANHO_COLL < collegue:
            return -1
        if filtering <= collegue:
            if WANHO_TOTAL < attitude + collegue:
                answer += 1
            filtering = collegue

    return answer
```

Brute force

[프로그래머스] 외벽 점검

- 전체 외벽 포인트 개수 n이 주어짐
- 점검해야 하는 취약 포인트가 번호 리스트로 주어짐
- 친구들이 외벽을 점검해주는데, 한 번에 점검할 수 있는 distance가 배열로 주어짐.
- 취약 포인트를 전부 점검하기 위해 투입해야 하는 친구의 수 최솟값을 반환해야 함.



n	weak	dist	result
12	[1, 5, 6, 10]	[1, 2, 3, 4]	2
12	[1, 3, 4, 9, 10]	[3, 5, 7]	1

[프로그래머스] 풀이법 – Brute force

- 제한사항의 weak의 길이와 dist의 길이가 충분히 짧으므로 Brute-force를 이용할 수 있다.
- 주어지는 친구 리스트가 짧으므로 순열을 사용하여 모든 경우를 테스트 할 수 있다. Dist가 [40, 30, 10, 5]인 경우와 [30, 40, 5, 10]인 경우는 결과가 다르다.
- 원형배열은 두배 늘려서 새 배열로 만들거나 맨 앞 요소를 맨 뒤에 붙여서 풀 수 있다. 주어진 weak의 길이가 짧으므로 두배 늘리면 됨
- 두배 배열은 처음 주어진 Weak의 길이만큼 잘라서 반복문에서 매번 사용하면 된다.

제한사항

- n은 1 이상 200 이하인 자연수입니다.
- weak의 길이는 1 이상 15 이하입니다.
  - 서로 다른 두 취약점의 위치가 같은 경우는 주어지지 않습니다.
  - 취약 지점의 위치는 오름차순으로 정렬되어 주어집니다.
  - weak의 원소는 0 이상 n - 1 이하인 정수입니다.
- dist의 길이는 1 이상 8 이하입니다.
  - dist의 원소는 1 이상 100 이하인 자연수입니다.
- 친구들을 모두 투입해도 취약 지점을 전부 점검할 수 없는 경우에는 -1을 return 해주세요.

[프로그래머스] 풀이법 – Brute force

- 모두 점검 못하는 경우를 대비해 answe에 len(dist)+1을 저장 (나중에 min함수를 사용할 것이므로)
- Weak의 배열을 두배 배열로 만들어 준다. (원형 배열 풀이를 위해)
- 원형 배열을 원래 주어진 weak 배열의 길이만큼 잘라서 매번 사용한다.
- 순열을 사용하여 친구 배열을 새로 만든다. 모든 경우의 수를 점검한다.

```python
from itertools import permutations
def solution(n, weak, dist):
    weak_length = len(weak)
    answer = len(dist) + 1

    for i in range(weak_length):
        weak.append(weak[i] + n)
```

```python
for start in range(weak_length):
    for friends in permutations(dist):
        count = 1

        position = weak[start] + friends[count - 1]
        for i in range(start, start + weak_length):
            if position < weak[i] :
                count += 1
                if count > len(dist):
                    break
                position = weak[i] + friends[count - 1]

        answer = min(count, answer)
```

[프로그래머스] 풀이법 – Brute force

- 위치(= 점검 시작점 + 친구가 갈 수 있는 거리)를 기록.
- 현재 잘라 놓은 배열의 취약 위치를 모두 검사한다. 만약 취약 위치보다 현재 위치가 더 작으면 필요한 사람 수를 증가시킨다.
- 만약 사람 수가 주어진 사람 수를 초과하면 반복문 break.
- 매 반복문에서 위치(= 점검 시작점 + 친구가 갈 수 있는 거리)를 새롭게 기록.
- 사람 수 count를 answer에 갱신
- 전부 검사 못하는 경우는 -1반환

```
for start in range(weak_length):
    for friends in permutations(dist):
        count = 1

        position = weak[start] + friends[count - 1]
        for i in range(start, start + weak_length):
            if position < weak[i] :
                count += 1
                if count > len(dist):
                    break
                position = weak[i] + friends[count - 1]

        answer = min(count, answer)

if answer > len(dist):
    return -1
return answer
```