PGS

#92342. 양궁대회

https://school.programmers.co.kr/learn/courses/30/lessons/92342

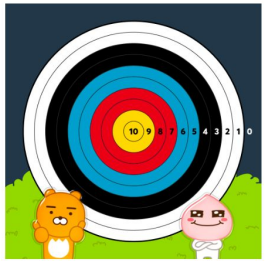
25.07.14

Problem

풀이시간: 1시간
힌트: 10시 20분부터

카카오배 양궁대회가 열렸습니다.
라이언은 저번 카카오배 양궁대회 우승자이고 이번 대회에도 결승전까지 올라왔습니다. 결승전 상대는 어피치입니다.
카카오배 양궁대회 운영위원회는 한 선수의 연속 우승보다는 다양한 선수들이 양궁대회에서 우승하기를 원합니다. 따라서, 양궁대회 운영위원회는 결승전 규칙을 전 대회 우승자인 라이언에게 불리하게 다음과 같이 정했습니다.

1. 어피치가 화살 n발을 다 쏜 후에 라이언이 화살 n발을 쏩니다.
2. 점수를 계산합니다.
   2-1. 과녁판은 아래 사진처럼 생겼으며 가장 작은 원의 과녁 점수는 10점이고 가장 큰 원의 바깥쪽은 과녁 점수가 0점입니다.



   2-2. 만약, k(k는 1~10사이의 자연수)점을 어피치가 a발을 맞혔고 라이언이 b발을 맞혔을 경우 더 많은 화살을 k점에 맞힌 선수가 k 점을 가져갑니다. 단, a = b일 경우는 어피치가 k점을 가져갑니다. k점을 여러 발 맞혀도 k점 보다 많은 점수를 가져가는 게 아니고 k점만 가져가는 것을 유의하세요. 또한 a = b = 0 인 경우, 즉, 라이언과 어피치 모두 k점에 단 하나의 화살도 맞히지 못한 경우는 어느 누구도 k점을 가져가지 않습니다.

- 예를 들어, 어피치가 10점을 2발 맞혔고 라이언도 10점을 2발 맞혔을 경우 어피치가 10점을 가져갑니다.
- 다른 예로, 어피치가 10점을 0발 맞혔고 라이언이 10점을 2발 맞혔을 경우 라이언이 10점을 가져갑니다.

   2-3. 모든 과녁 점수에 대하여 각 선수의 최종 점수를 계산합니다.

3.최종 점수가 더 높은 선수를 우승자로 결정합니다. 단, 최종 점수가 같을 경우 어피치를 우승자로 결정합니다.

현재 상황은 어피치가 화살 n발을 다 쏜 후이고 라이언이 화살을 쏠 차례입니다.
라이언은 어피치를 가장 큰 점수 차이로 이기기 위해서 n발의 화살을 어떤 과녁 점수에 맞혀야 하는지를 구하려고 합니다.

화살의 개수를 담은 자연수 n, 어피치가 맞힌 과녁 점수의 개수를 10점부터 0점까지 순서대로 담은 정수 배열 info가 매개변수로 주어집니다. 이때, 라이언이 가장 큰 점수 차이로 우승하기 위해 n발의 화살을 어떤 과녁 점수에 맞혀야 하는지를 10점부터 0점까지 순서대로 정수 배열에 담아 return 하도록 solution 함수를 완성해 주세요. 만약, 라이언이 우승할 수 없는 경우(무조건 지거나 비기는 경우)는 [-1]을 return 해주세요.

Hint

Step 1. 문제 분석

문제 요약

라이언과 어피치는 양궁 대회 결승전에 참여한다.

어피치가 먼저 n발의 화살을 쐈고, 라이언이 다음으로 n발을 쏘려 한다. 라이언이 가장 큰 점수차로 이기기 위해 어떤 과녁에 몇 발을 맞춰야 하는지 정수 배열로 반환하라

- k점(1~10)에 어피치가 a발을 맞췄고, 라이언이 b발을 맞췄을 때, b>a이면 라이언만 k점을 얻고, b<=a이면 어피치만 k점을 얻는다.
  a. 둘다 0발을 맞춘 경우 k점은 점수 계산에서 제외한다.
- 10점부터 1점까지 모든 점수에 대해 라이언과 어피치의 점수를 계산한다.
- 총 점수가 라이언이 어피치보다 크면 라이언이 우승이고, 라이언과 어피치가 같거나 라이언이 어피치보다 작으면 어피치가 우승이다.
- 라이언이 가장 큰 점수차로 이기기 위해 어떤 과녁에 몇 발을 맞춰야 하는지 정수 배열로 반환하라
  a. 라이언이 이길수 있는 경우가 없으면 -1을 출력하라
  b. 가장 큰 점수차로 이길 수 있는 배열이 여러개이면 가장 낮은 점수를 더 많이 맞춘 배열을 출력하라

목표는 (**모든 경우의 수 중 라이언이 어피치를 이길 수 있는 경우) & (라이언 총 점수) - (어피치 총 점수) 가 최대인 경우 & 결과 배열이 여러개라면, 가장 낮은 점수를 더 많이 맞춘 배열 선택. 이 3가지다.**

💡 Hint

Step 1. 문제 분석

💡 문제 유형

- "라이언이 어피치를 최대 점수차로 이길 수 있는 경우의 수를 어떻게 찾을지"가 핵심이다.
- **그리디한 방식(높은 점수를 무조건 어피치보다 많이 쏘는)은 안된다.**
  a. **문제 1번 예시)**
  b. [2,1,1,1,0,0,0,0,0,0,0] 어피치
  c. [0,2,2,0,1,0,0,0,0,0,0] 라이언
- 모든 경우의 수를 구하기 위해 DFS 알고리즘 사용
- 라이언은 화살을 쏠 때, 어피치가 k점을 맞춘 횟수보다 1만큼 더 쏘거나, 아예 쏘지 않아서 화살을 아끼는 두 가지 방향으로 DFS를 진행한다.

💡 Hint

Step 2. 접근 방식

💡 풀이 아이디어

- DFS에서 유지해야 할 상태는 무엇일까?
  a. 현재 어떤 점수(10점부터 1점까지)판을 기준으로 DFS를 진행하는지 알려주는 idx, 현재 남은 화살의 수인 cnt
- DFS의 종료 조건은 무엇일까?
  a. 현재 쏘는 점수가 0점인 경우(10점부터 내림차순으로 점수판 내려왔기 때문) OR 남은 화살 수가 0인 경우
- DFS는 어떻게 진행해야 할까?
  a. 현재 점수를 아예 안쏴서 화살을 아끼던가 현재 점수에서 어피치가 맞춘 화살의 수 +1 만큼 맞춰서 점수를 따오던가

💡 Hint

Step 2. 주의할 점

- 항상 DFS + backtracking시 backtracking이 잘 되었는지 주의하자.

- 같은 최대 점수가 나온 두 배열의 순위를 구하려면 무조건 화살을 다써야 한다. 즉 DFS 종료 시 남은 화살이 있다면 0점으로 몰아주자.

✨Solution

```
def solution(n, info):
    global max_gap, answer

    answer = [-1]
    score = [0]*11
    max_gap=0

    def is_winner_with_gap(score):
        a=0 # 어피치 점수
        b=0 # 라이언 점수

        for i in range(len(info)):
            if info[i] > 0 or score[i] > 0:
                if info[i]>=score[i]:
                    a += (10-i)
                else:
                    b += (10-i)
        return (b > a, abs(a-b))

    def dfs(L, cnt):
        global max_gap, answer
        if L == 11 or cnt == 0:
            is_winner, gap = is_winner_with_gap(score)
            if is_winner:
                if cnt >= 0: # 화살이 남은 경우
                    score[10] = cnt # 0점에 쏴도 이김

                if gap > max_gap: # 갭이 더 큰 경우로 업데이트
                    max_gap = gap
                    answer = score.copy()
```

💻 Assignment

https://www.acmicpc.net/problem/1230

문자열 거리