2022 HCPC 해설

HCPC | 2022
Hanyang Collegiate Programming Contest

Official Solutions

by

2022 ALOHA

2022년 12월 04일

2022 HCPC - 후원사

출제진

- 구재원 (jjaewon9) (한양대학교 컴퓨터소프트웨어학부 22학번 / ALOHA)
- 이성현 (hibye1217) (한양대학교 컴퓨터소프트웨어학부 22학번 / ALOHA)
- 조예진 (choyj427) (한양대학교 컴퓨터소프트웨어학부 20학번 / ALOHA)
- 문종건 (andrewmjk1) (한양대학교 컴퓨터소프트웨어학부 20학번 / ALOHA)
- 최철민 (effect2110) (한양대학교 컴퓨터소프트웨어학부 21학번 / ALOHA)

내부 검수진

- 김준서(junseo)
  (컴퓨터소프트웨어학부 22학번 / ALOHA)
- 신용명(tlsdydaud1)
  (컴퓨터소프트웨어학부 20학번)
- 최준익(nick832)
  (의예과 21학번 / ALOHA)

외부 검수진

- jthis
- jyheo98
- leejseo
- powergee
- shjohw12

Beginner Division

문제		의도한 난이도	출제자
BA	출제비 재분배	**Easy**	**hibye1217(이성현)**
BB	즉흥 여행 (Easy)	**Medium**	**hibye1217(이성현)**
BC	Wordle 찍기	**Medium**	**hibye1217(이성현)**
BD	세로 달력	**Easy**	**effect2110(최철민)**
BE	선물의 재분배	**Medium**	**hibye1217(이성현)**
BF	Identify, Sort, Index, Solve	**Easy**	**hibye1217(이성현)**
BG	NATO 음성 기호와 쿼리	**Hard**	**hibye1217(이성현)**
BH	플래피 버드 스코어링	**Medium**	**effect2110(최철민)**
BI	로하의 농사	**Medium**	**choyj427(조예진)**
BJ	한양 가왕	**Medium**	**andrewmjk1(문종건)**

Advanced Division

문제		의도한 난이도	출제자
AA	NATO 음성 기호와 쿼리	**Hard**	**hibye1217(이성현)**
AB	배수관 미스터리	**Medium**	**andrewmjk1(문종건)**
AC	나락도 락이다	**Medium**	**jjaewon9(구재원)**
AD	즉흥 여행 (Hard)	**Hard**	**hibye1217(이성현)**
AE	All Solve를 향해!	**Hard**	**hibye1217(이성현)**
AF	출제비 재분배	**Easy**	**hibye1217(이성현)**
AG	트리와 수열	**Medium**	**jjaewon9(구재원)**
AH	보물 찾기	**Challenging**	**hibye1217(이성현)**
AI	트리의 팔	**Challenging**	**effect2110(최철민)**
AJ	선물의 재분배	**Medium**	**hibye1217(이성현)**

BA, AF. 출제비 재분배

`implementation`, `simulation`, `arithmetic`
출제진 의도 – **Easy**

- 제출 24번, 정답 14팀 (정답률 62.50%) BEG
- 제출 42번, 정답 19팀 (정답률 47.62%) ADV
- 처음 푼 팀: **Keokiji_Anneun_Maum(BEG)**, 3분
- 처음 푼 팀: **Overfitting(ADV)**, 4분
- 출제자: hibye1217(이성현)

BA, AF. 출제비 재분배

- 문제에서 하라는 대로 하면 됩니다.
- 우선 각 출제자에게 출제비 S_i 를 더한 뒤, 각 i, j 쌍에 대해 i 번 출제자가 가지고 있는 돈에서 $T_{i,j}$ 를 뺀 뒤, j 번 운영자가 가지고 있는 돈에 $T_{i,j}$ 를 더해주면 됩니다.

BE, AJ. 선물의 재분배

`greedy`, `constructive`, `ad-hoc`
출제진 의도 – **Medium**

- 제출 2번, 정답 1팀 (정답률 50.00%) BEG
- 제출 18번, 정답 5팀 (정답률 27.78%) ADV
- 처음 푼 팀: **HaHaHoHo(ADV)**, 81분
- 처음 푼 팀: **Keokiji_Anneun_Maum(BEG)**, 75분
- 출제자: hibye1217(이성현)

- 여럿이 선물을 가지고 있는 상태에서 서로가 서로에게 나눠주는 건 너무 복잡하니, 차라리 한 명에게 몰아줬다가 나누는 방법을 생각해봅시다.
- 편의상 이 사람을 '중심'이라고 합시다.
- 중심에게 선물을 몰아주는 건 간단합니다. 중심을 제외한 사람들을 선물 순으로 정렬한 뒤, 차례대로 1명씩 중심에게 모든 선물을 옮겨주면 됩니다.

BG, AA. NATO 음성 기호와 쿼리

`dynamic-programming`, `string`, `binary-search`, `prefix-sum`
출제진 의도 – **Hard**

- 제출 5번, 정답 0팀 (정답률 00.00%) BEG
- 제출 17번, 정답 0팀 (정답률 00.00%) ADV
- 처음 푼 팀: , 분
- 출제자: hibye1217(이성현)

- NATO 쿼리에 대해 분석해봅시다.
- 우선 모든 NATO 음성 문자의 길이가 4 이상이므로, 문자열의 길이는 쿼리를 1번 적용시킬 때마다 최소 4배가 됩니다.
- 출력 쿼리로 주어지는 위치가 최대 10^{18} 이므로, 쿼리를 40번만 줘도 문자열의 뒷부분에 대한 정보는 필요가 없어짐을 생각해볼 수 있습니다.

- 또한 모든 NATO 음성 문자는 대응되는 글자로 시작하므로, 쿼리를 적용해도 바뀌지 않는 무언가가 있음을 기대해볼 수 있습니다.
- 실제로 그런 특징을 찾아볼 수 있으며, 아래와 같습니다.
- 편의상 S^k 를 S 에 NATO 쿼리를 k 번 적용시켰을 때의 문자열로 둡시다.
- 자명히, $S^0 = S$ 입니다.

- 이제 S^0에 쿼리를 한 번 적용해봅시다.
- NATO 음성 문자의 맨 앞글자는 항상 대응되는 글자와 동일하기 때문에, S^0과 S^1의 첫 글자는 동일하게 됩니다.
- 이제 S^1에 쿼리를 한 번 더 적용해봅시다.
- 그럼, S^0의 첫 1글자와 S^1의 첫 1글자가 동일하고, 모든 NATO 음성 문자의 길이가 최소 4 이상이기 때문에, S^1과 S^2의 첫 4글자는 반드시 동일하게 됩니다.
- 이런 식으로, S^i과 S^{i+1}의 첫 4^i 글자는 반드시 동일함을 알아낼 수 있습니다.
- 출력 쿼리로 주어지는 위치가 최대 10^{18}이므로, 쿼리를 40번만 줘도 실질적으로 신경써야 하는 부분은 전혀 바뀌지 않음을 알 수 있습니다.

- 이제 이를 토대로 DP를 세워봅시다.
- $LEN_{i,c}$를 글자 c에 NATO 쿼리를 i번 적용시킬 때의 결과 문자열의 길이라고 하면, $LEN_{i,c} = \sum_{c' \in NATO[c]} LEN_{i-1,c'}$ 이 됩니다.
- 가능한 알파벳의 종류는 26가지고, 신경써야 하는 NATO 쿼리의 횟수는 40번 정도가 되기 때문에, 이 DP의 공간복잡도는 $O(26 \times 40)$ 이 됩니다.

- 이제, 지금까지 NATO 쿼리가 k 번 적용되었다고 할 때,
- DP_i 를 현재까지 적용된 쿼리를 기준으로, 초기 문자열 S^0 의 첫 i 글자가 차지하는 글자 수라고 하면, $DP_i = DP_{i-1} + LEN_{k,S_i^0}$ 가 됩니다.
- 실제로 출력할 때는 위 DP를 토대로 초기 문자열의 몇 번째 글자를 펼쳐야 하는지 이진 탐색으로 알아낸 다음, 이에 해당되는 실제 글자를 LEN 을 역추적해나가면서 찾아주면 됩니다.

BB. 즉흥 여행 (Easy)

`depth-first-search`
출제진 의도 – **Medium**

- 제출 3번, 정답 1팀 (정답률 33.33%)
- 처음 푼 팀: **CCD**, 103분
- 출제자: hibye1217(이성현)

BB. 즉흥 여행 (Easy)

- 모든 정점에서 다른 모든 정점으로 갈 수 있는 경로가 있다면 YES이고, 아니면 NO입니다.
  이유는 다음과 같습니다.
- 만약 모든 정점에서 다른 모든 정점을 방문할 수 있다면, 시작점이 어떻게 잡히든 거기서
  목표점을 잡고 이동한 뒤, 거기서 또 하나의 목표점을 잡고, ... 을 반복하면 됩니다.
- 하지만 그렇지 못한 정점 쌍이 있다면, 그 정점 쌍의 한 쪽에서 출발하면 반드시 다른 한 쪽으로
  이동할 수 없습니다.

- 하지만 이대로 구현하면 $O(N^2)$ 으로 시간 초과를 받게 됩니다.
- 다행히도, 모든 정점에서 다른 모든 정점을 방문할 수 있음을 더 빠르게 판별할 수 있습니다.
- 바로 1번 정점에서 다른 모든 정점으로 갈 수 있는지와 그 반대 방향도 성립하는지만 보면 됩니다.

- 만약 이게 성립한다면, 임의의 정점 쌍 (v, w)에 대해 $v \to 1 \to w$의 경로를 찾을 수 있습니다.
- 그렇지 않다면, 이미 방문할 수 없는 정점 쌍을 찾은 거나 마찬가지입니다.
- 그러니 1번 정점에서 dfs를 돌리고, 그래프를 뒤집은 뒤 1번 정점에서 다시 dfs를 돌려서 두 경우 모두 모든 정점이 방문되었는지 체크하면 됩니다.

BC. Wordle 찍기

`ad-hoc`, `constructive`, `case-work`

출제진 의도 – **Medium**

- 제출 3번, 정답 1팀 (정답률 33.33%)
- 처음 푼 팀: **Keokiji_Anneun_Maum**, 57분
- 출제자: hibye1217(이성현)

- 일단 불가능한 경우를 생각해봅시다.
- 이는 초록색이 4개 + 노란색이 1개인 경우가 유일합니다.
- 불가능한 이유는, 노란색으로 칠해진 글자가 매치될 수 있는 위치가 자기 자신밖에 남지 않는데, 그렇게 되면 노란색이 아니라 초록색이 칠해져야 하기 때문입니다.

BC. Wordle 찍기

- 나머지 경우는, 아래와 같이 경우를 나눠서 가능한 경우를 만들어줄 수 있습니다.
- 편의상, 정답 단어를 ABCDE라고 합시다.
- 회색 글자는 Z 등 의미없는 글자로 채워넣으면 됩니다.
- 초록색 글자는 정답 단어와 동일하게 채워넣으면 됩니다.
- 노란색 글자는 아래와 같이 처리하면 됩니다.
- 노란색 글자가 여럿이라면, 그들간의 자리를 바꿔주면 됩니다.
- 노란색 글자가 하나 있다면, 회색 글자의 위치에 있던 걸 넣어주면 됩니다.
- 만약 노란색은 하나인데 회색 글자가 없다면, 이는 아까 우리가 본 불가능한 경우에 해당되므로 IMPOSSIBLE을 출력하면 됩니다.

BD. 세로 달력

`implementation`, `math`

출제진 의도 – **Easy**

- 제출 17번, 정답 13팀 (정답률 76.47%)
- 처음 푼 팀: **Keokiji_Anneun_Maum**, 25분
- 출제자: effect2110(최철민)

- 잘 생각해보면 첫 시작일이 달라진다고 해서 세로 달력 개수가 바뀌지는 않습니다.
- 따라서, 총 개수는 고정되어 있음을 알 수 있고 이 개수는 윤년에만 바뀝니다.
- 결론적으로는 윤년인지만 판단해주면 됩니다.

BF. Identify, Sort, Index, Solve

`implementation, sort`

출제진 의도 – **Easy**

- 제출 20번, 정답 13팀 (정답률 75.00%)
- 처음 푼 팀: **Keokiji_Anneun_Maum**, 8분
- 출제자: hibye1217(이성현)

- 문제에서 나온 그대로 구현하면 됩니다.
- 번호 순으로 정렬한 뒤, 문자열의 D번째 글자를 차례로 출력하면 됩니다.
- 소문자는 대문자로 바꿔 출력해야 함에 주의하세요.

BH. 플래피 버드 스코어링

`binary-search`
출제진 의도 – **Medium**

- 제출 63번, 정답 4팀 (정답률 6.44%)
- 처음 푼 팀: **Keokiji_Anneun_Maum**, 19분
- 출제자: effect2110(최철민)

- Naive 하게 풀기에는 쿼리의 수가 너무 많습니다.
- 따라서, 하나의 쿼리를 처리하는데 걸리는 시간을 줄일 필요가 있습니다.
- 장애물 틈새가 이전보다 커지는 것은 의미가 없습니다. 틈새가 반대로 이전보다 작아지는 것만 의미가 있습니다.
- 장애물을 훑으면서 각 지점까지의 틈새의 최솟값을 구한 값을 이용하면, 쿼리를 $O(logN)$ 에 처리할 수 있습니다.
- 해당 값은 단조 감소하기 때문에 std::upper_bound 를 이용하여 값을 찾아낼 수 있습니다.
- 따라서, $O(QlogN)$ 에 문제를 해결할 수 있습니다.

BI. 로하의 농사

`brute-force`, `backtracking`
출제진 의도 – **Medium**

- 제출 7번, 정답 0팀 (정답률 00.00%)
- 처음 푼 팀: -, -분
- 출제자: choyj427(조예진)

- 간단한 브루트포스, 백트래킹 문제입니다.
- 이전에 탐색했던 방향과 같은 방향을 탐색하면 파이프를 1개 늘리고, 다른 방향을 탐색한다면 파이프를 2개 늘려야 합니다.
- 시간복잡도는 $O(3^p)$ 이고 p가 20 이하이므로 2초 안에 해결 가능합니다.

BJ. 한양 가왕

`math, simulation`
출제진 의도 – **Medium**

- 제출 30번, 정답 3팀 (정답률 10.00%)
- 처음 푼 팀: **ITBT_MeN**, 73분
- 출제자: andrewmjk1(문종건)

- 주어진 총 라운드의 수 M 이 최대 10^9 입니다. 실제 모든 라운드를 시뮬레이션하면 $O(NM)$ 의 시간복잡도를 갖게 되어, 시간 초과를 피하기 힘듭니다.
- 유독 1번 노래방 기계에서만 패배자와 승리자가 반대로 움직입니다. 모든 라운드를 시뮬레이션하지 않아도 마지막 결과를 알 수 있는 방법이 있을까요?
- 최대 $2N$ 라운드 이후, 참가자들의 위치가 N라운드마다 반복되는 주기성을 가진다는 사실을 알 수 있습니다. 이 사실을 증명해 봅시다.
- $2N$ 명의 참가자들 중 $N+1 \sim 2N$ 의 실력 점수를 가진, 나머지 절반보다 강한 참가자들을 "우승 후보"라고 해봅시다.

- 최대 $N-1$ 라운드 이후, 실력 점수 $2N$ 을 가진 참가자는 반드시 1번 노래방 기계로 와서 고정된 위치를 갖게 됩니다. 가장 높은 실력 점수를 가졌기 때문에 모든 승부에서 승리할 것이고, 그렇기 때문에 시작 위치와 관계없이 반드시 $N-1$ 라운드 이후 1번 노래방 기계로 오게 됩니다.
- 이렇게 실력 점수 $2N$ 의 참가자가 1번 노래방 기계로 오게 되면, 이후 N 라운드 이내로 우승 후보안 참가자들(실력 점수 $N+1 \sim 2N$)이 모든 노래방 기계에 골고루 분포하게 됩니다. 우승 후보인 참가자가 하나의 노래방 기계에 위치하는 순간 그 노래방 기계는 반드시 우승 후보인 참가자가 항상 최소 1명 존재하게 된다는 것을 감안하면, 이 사실은 귀류법으로 증명할 수 있습니다.
- 즉 $2N$ 라운드 이내로 모든 노래방 기계에는 우승 후보와 우승 후보가 아닌 참가자가 쌍을 이룬 형태로 배정됨을 알 수 있습니다. 한번 이런 형태의 배치가 이루어지고 나면, 우승 후보가 아닌 참가자들은 절대 움직일 일이 없게 됩니다.

- 이로써 최대 $2N$ 라운드 이후 반드시 참가자들의 이동에 주기성이 생긴다는 사실을 증명할 수 있습니다.
- 따라서 총 라운드의 수 M이 2N보다 클 경우, $M' = 2N + (M \mod N)$ 으로 대체하여 M' 라운드만 시뮬레이션 하면 답을 얻을 수 있습니다!

AB. 배수관 미스터리

`disjoint-set`, `offline-query`
출제진 의도 – **Medium**

- 제출 20번, 정답 5팀 (정답률 25.00%)
- 처음 푼 팀: **HaHaHoHo**, 80분
- 출제자: andrewmjk1(문종건)

- 임계 확률에 대한 질의가 들어올 때마다, 해당 확률보다 연결 확률이 높은 배수관들을 전부 연결했을 때의 집합의 개수를 구해야 합니다.
- 즉, 배수관들의 연결 방식과 상관없이 각 배수관들이 어느 집합에 속해 있는지만 구분하여 개수를 세어주면 됩니다.
- 따라서, 분리 집합 자료구조를 사용하여 집합의 개수를 세어줄 수 있습니다.
- 단, 매 질의마다 모든 배수관의 연결 확률을 확인하며 이어주게 되면 질의 하나당 $O(MlogN)$ 의 시간복잡도가 됩니다. 모든 질의에 대해 다 답하고 나면 $O(QMlogN)$ 의 시간복잡도가 되기 때문에, 최적화할 방법을 찾아봅시다.

- 모든 질의를 미리 입력받고, 임계 확률이 높은 순서대로 질의 목록을 정렬해 줍시다. 마찬가지로 배수관들의 목록 또한 연결 확률이 높은 순서대로 정렬해주면, 이후 임계 확률이 높은 순서대로 배수관의 연결 여부를 살펴보면 됩니다.
- 위와 같은 순서대로 배수관들의 연결을 진행하면서 순차적으로 각 질의에 대한 답을 저장해놓으면 한 질의에 대한 답을 얻은 뒤 다음 질의에 대해 답하기 위해 분리 집합을 초기화할 필요가 없습니다.
- 따라서 $O(QlogQ + MlogM + MlogN)$의 시간 복잡도가 되고, 이는 시간 내에 문제를 풀기에 적합한 시간복잡도입니다.

AC. 나락도 락이다

`dp`, `prefix-sum`, `combinatorics`
출제진 의도 – **Medium**

- 제출 52번, 정답 7팀 (정답률 15.39%)
- 처음 푼 팀: **HaHaHoHo**, 11분
- 출제자: jjaewon9(구재원)

- $K[i] = S[i...N)$ 중 "K" 부분열의 개수
- $CK[i] = S[i...N)$ 중 "CK" 부분열의 개수
- $OCK[i] = S[i...N)$ 중 "OCK" 부분열의 개수로 정의합시다. (0-based)
- $S[i] =$ 'R'인 모든 i에 대해, $2^i \cdot OCK[i]$ 의 합이 문제에서 요구하는 답입니다.
- 각 배열은 역순으로 $O(N)$에 채워줄 수 있으므로, $O(N)$에 전체 문제가 해결됩니다.

AD. 즉흥 여행 (Hard)

`strongly-connected-component`
출제진 의도 – **Hard**

- 제출 17번, 정답 2팀 (정답률 11.77%)
- 처음 푼 팀: **nan_da_joa**, 60분
- 출제자: hibye1217(이성현)

AD. 즉흥 여행 (Hard)

- SCC의 정의 상, 하나의 SCC에 속한 두 정점 v, w 사이에는 자유롭게 다닐 수 있는 경로가 존재합니다.
- 또한, SCC를 하나의 정점으로 취급하면, DAG가 만들어지게 됩니다.
- DAG에서 모든 정점을 방문할 수 있는 경로가 존재하려면, DAG가 일직선으로 나열될 수 있어야 합니다.
- 이는 DAG의 위상 정렬 순서가 유일한지 판별하면 됩니다.
- 정렬 순서가 유일하다면, 맨 앞에 있는 SCC에서 모든 정점을 방문할 수 있으므로, 맨 앞에 있는 SCC에 속한 정점들을 출력하면 됩니다.
- 그렇지 않다면, 어느 위치에서도 모든 정점을 방문할 수 없으므로 답은 0이 됩니다.

AE. All Solve를 향해!

`binary-search`
출제진 의도 – **Hard**

- 제출 19번, 정답 2팀 (정답률 10.53%)
- 처음 푼 팀: **HaHaHoHo**, 149분
- 출제자: hibye1217(이성현)

- 문제를 번호 순으로 스위핑해봅시다.
- D_i를 i번째 날에 푼 문제 중 가장 어려웠던 문제라고 하고, P_i를 i번째 날에 푼 문제 수라고 합시다.
- 조건의 특성상 가장 어려웠던 문제와 가장 최근에 푼 문제는 동일하기 때문에, D에 대해 크게 신경써야 할 부분도 없습니다.
- 특정 문제가 페이지에 보이는 날은 연속된 구간으로 표현되기 때문에, 스위핑과 함께 관리할 두 포인터 p, q를 준비해봅시다.
- 이는 현재 보는 문제가 p일에서 q일까지 페이지에 보였음을 의미합니다.

- 이제 각각의 값을 스위핑하면서 어떻게 업데이트해야 할까요?
- 우선, D_i 는 $[p, q]$ 구간에서 감소합니다. (그렇지 않다면, 그 문제를 더 일찍 풀었어야 하므로)
- 그러니, D 에서 이진탐색을 돌리면 새로운 문제를 풀게 되는 날짜가 나오게 됩니다.
- 그 날짜에 문제를 풀었음을 D 와 P 에 업데이트한 뒤, 만약 이 날짜가 새로운 날이라면 q 에 1 을 더해줍니다.
- 이제, 만약 건너뛴 문제의 수가 k 가 되었다면, 즉 $\sum_{i=p+1}^{\infty} P_i = k$ 가 되었다면, p 에 1 을 더해줍니다.
- 위 시뮬레이션을 빠르게 돌려주면 됩니다.

AE. 트리와 수열

`tree-dp`, `greedy`, `sorting`
출제진 의도 – **Medium**

- 제출 28번, 정답 7팀 (정답률 28.57%)
- 처음 푼 팀: **HaHaHoHo**, 39분
- 출제자: jjaewon9(구재원)

- $\frac{N(N-1)}{2}$ 개의 경로를 하나씩 고려하는건 불가능합니다.
- 각 간선을 지나는 경로가 몇개인지 구해봅시다.
- 임의의 정점을 루트로 잡고, 모든 i에 대해 i번 정점을 루트로 하는 서브트리의 크기를 구해 이를 $sz[i]$라고 합시다.
- i번 정점과 i번 부모 정점을 잇는 간선을 지나는 경로의 개수는 $(N - sz[i])sz[i]$와 같습니다.
- a_i를 주어진 수열, b_i를 $(N - sz[i])sz[i]$라고 합시다. $\sum dist(i,j) = \sum a_i b_i$ 입니다.
- a_i와 b_i를 각각 적절히 재배열하여 $\sum a_i b_i$의 최솟값을 구하는 문제가 되었습니다.

- 위 값은 a 를 오름차순으로, b 를 내림차순으로 정렬했을 때가 최솟값임이 알려져 있습니다. (재배열 부등식)
- b_i 를 tree dp 를 이용해 계산하는데 $O(N)$, a 와 b 를 정렬하는 데에 $O(NlogN)$ 이 소요되므로 총 $O(NlogN)$ 에 문제가 해결됩니다.
- 주의해야하는 점으로 b_i 를 64bit 자료형으로 선언해야 한다는 것, 그리고 b_i 를 계산 후 정렬한 뒤에 모듈러를 취해야 한다는 것이 있습니다.

AH. 보물 찾기

`geometry`, `ccw`, `convex-hull`, `point-in-convex-polygon`, `binary-search`, `case-work`

출제진 의도 – **Challenging**

- 제출 2번, 정답 0팀 (정답률 00.00%)
- 처음 푼 팀: -, -분
- 출제자: hibye1217(이성현)

AH. 보물 찾기

- 시작점과 보조점으로 볼록 껍질을 만들어봅시다.
- 끝점이 이 볼록 껍질 밖에 있으면, 자명히 도달할 수 없습니다.
- 아닌 경우, 아래 4가지 경우로 나눌 수 있습니다.
  1. 볼록 껍질이 점인 경우
  2. 선인 경우
  3. 삼각형인 경우
  4. 나머지 경우 (사각형 또는 그 이상)

- 1번 경우(점)는 아무 움직임도 할 수 없는 상태입니다. 그러니, 끝점이 시작점과 동일하다면 YES, 아니면 NO입니다.
- 모든 점의 위치가 동일하기 때문에, 움직임은 의미가 없습니다.
- 2번 경우(선)는 그 선 위에서는 자유롭게 움직일 수 있습니다. 그러니, 끝점이 그 선 위에 있으면 YES, 아니면 NO입니다.
- 선의 한쪽 끝으로 이동한 뒤, 반대쪽 끝을 바라보고 끝점까지 적당히 움직이면 됩니다.

- 3번 경우(삼각형)는, 우선 시작점을 P_1, 두 중간점을 각각 P_2, P_3 이라고 해봅시다.
- 목표점 P_4 와 P_3 을 잇는 직선은 P_1 과 P_2 를 잇는 선분과 최소 한 점에서 만나게 됩니다. 그 점을 Q 라고 한 뒤, P_1 에서 Q 로 이동합니다.
- 이제 Q, P_4, P_3 가 일직선에 있으니, 2번 경우와 비슷하게 문제를 풀면 됩니다.
- 4번 경우(기타)는 끝점을 포함하는 삼각형을 아무거나 찾은 뒤 3번 경우와 동일하게 풀면 됩니다.

AI. 트리의 팔

`dfs`, `fft`, `prefix-sum`
출제진 의도 – **Challenging**

- 제출 4번, 정답 1팀 (정답률 25.00%)
- 처음 푼 팀: -, -분
- 출제자: effect2110(최철민)

- ”$A[i]$ = i 깊이의 리프노드 개수”라고 합시다.
- 각 팔의 조합을 구해야합니다.
- 잘 생각해보면, A 배열을 자기 자신과 이산 합성곱해주면
  필요한 값을 얻어올 수 있음을 알 수 있습니다.
- 합성곱은 FFT를 이용하면 $O(N \log N)$에 구할 수 있습니다.
- 이때, 범위의 합을 구해야하므로 prefix-sum을 사용할 수 있습니다.
- 쿼리에 $O(1)$ 시간이 걸리고 전처리가 $O(N \log N)$이므로,
  전체 시간복잡도 $O(N \log N + Q)$에 해결할 수 있습니다.