목차

문제		의도한 난이도	출제자
A	인간은 무엇인가	**Beginner**	김형진plast
B	아카라카 2	**Easy**	김형진plast
C	흑백 요리사	**Easy**	김태윤ystaeyoon113
D	게임 오브 데쓰 (Easy)	**Medium**	김형진plast
E	출구가 바뀌는 미궁	**Medium**	김형진plast
F	트리의 루트를 찾아라	**Medium**	김형진plast
G	게임 오브 데쓰 (Hard)	**Medium**	김형진plast
H	오장원전	**Hard**	김태윤ystaeyoon113
I	돌 게임	**Hard**	김형진plast
J	빨간점, 파란점 3	**Challenging**	이국렬lky7674

출제진 소개

- ✓ 김형진 (plast) - 연세대학교
  - 2023 삼성전자 대학생 프로그래밍 대회 (SCPC) 4 등상
  - 2023 현대모비스 프로그래밍 경시대회 우수상 (14 등)
  - 2024 신촌지역 대학생 프로그래밍 대회 (SUAPC) 1st
- ✓ 김태윤 (ystaeyoon113) - 연세대학교
  - 2023 삼성전자 대학생 프로그래밍 대회 (SCPC) 4 등상
  - 2024 ICPC Asia Pacific Championship 21th
  - 2023/2024 신촌지역 대학생 프로그래밍 대회 (SUAPC) 1st

출제진 소개

- 이국렬 (lky7674) - 연세대학교, 삼성전자
  - 2019 ICPC Seoul Regional 3rd
  - 2020 ICPC World Finalist

검수진 소개

- ✓ 곽재혁(dreami63) - 연세대학교
- ✓ 장래오(leo020630) - 포항공과대학교
  - 2024 ICPC World Finalist
  - 명예 연세인
- ✓ 정용진(playsworld16) - 연세대학교
  - 2020 연세대학교 프로그래밍 경진대회 1st
  - 2023/2024 신촌지역 대학생 프로그래밍 대회(SUAPC) 1st
- ✓ 정현서(jhwest2) - 서울대학교
  - 2024 ICPC World Finalist, 9th place(Bronze Medal)

검수진 소개

- dabbler1
- hamuim
- sjhi00
- yup0927
- 모든 검수진 분들께 감사드립니다.

A. 인간은 무엇인가

`input-output, if`
출제진 의도 – **Beginner**

- ✓ 제출 29번, 정답 18명 (정답률 62.069%)
- ✓ 처음 푼 사람: **박건휘**, 0분
- ✓ 출제자: **김형진(plast)**

A. 인간은 무엇인가

- 10만 이하의 2024의 배수인지 여부를 판정하는 문제입니다.
- $N \leq 100,000$과 N % 2024 = 0 두가지 조건문을 이용해 검사하면 판정할 수 있습니다.

B. 아카라카 2

`string`, `greedy`
출제진 의도 – **Easy**

- 제출 20번, 정답 18명 (정답률 90.000%)
- 처음 푼 사람: **박건휘**, 1분
- 출제자: **김형진 (plast)**

B. 아카라카 2

- ✓ AKARAKA가 정확히 K번 등장하는 문자열 중 가장 짧은 문자열은, AKA 뒤에다가 RAKA를 정확히 K개 이어 붙인 문자열입니다.
- ✓ 증명은 궁금하시면 읽어보세요.

B. 아카라카 2

- ✓ AKARAKA가 정확히 K번 등장하는 문자열 중 가장 짧은 문자열은, AKA 뒤에다가 RAKA를 정확히 K개 이어 붙인 문자열(이때 길이는 4K+3)로 유일함을 보이겠습니다.
- ✓ S가 AKARAKA가 정확히 K번 등장하는 문자열이라고 가정합니다. 만약 K=1 이면, AKARAKA의 길이가 7이므로 S의 길이는 7 이상이며, S의 길이가 7일 때 S는 AKARAKA 임이 자명합니다.

B. 아카라카 2

- 이제 K>1 일때를 확인하겠습니다. S의 x번째 문자부터 x+6번째 문자로 이루어진 부분문자열이 AKARAKA일 때, S와 AKARAKA가 x에서 매칭된다고 하겠습니다. S와 AKARAKA가 매칭되는 위치를 $a_1, a_2, \ldots, a_K$ (단, $a_1 < a_2 < \ldots < a_K$) 라고 하겠습니다. 만약 어떤 i에 대하여 $a_{i+1} - a_i \leq 3$ 이면, 모순임을 확인할 수 있습니다.
- $a_{i+1} - a_i = 1$ 인 경우 a_{i+1} 번째 문자가 A인 동시에 K여야 해서 모순, $a_{i+1} - a_i = 2$ 인 경우 $a_{i+1} + 1$ 번째 문자가 K인 동시에 R여야 해서 모순, $a_{i+1} - a_i = 3$ 인 경우 a_{i+1} 번째 문자가 A인 동시에 R여야 해서 모순입니다.

B. 아카라카 2

- ✓ 따라서 모든 $i(1 \leq i < K)$에 대하여 $a_{i+1} - a_1 \geq 4$ 이므로, $a_K - a_1 \geq 4(K-1)$ 입니다. S의 길이는 $a_K + 6$ 이상이어야 하므로, S의 길이는 4K+3 이상임을 알 수 있습니다.
- ✓ 만약 S의 길이가 정확히 4K+3 이면, $a_K - a_1 = 4(K-1)$ 이므로, 모든 $i(1 \leq i < K)$ 에 대하여 $a_{i+1} - a_i = 4$ 이며, 이를 만족하는 문자열은 AKA 뒤에다가 RAKA를 정확히 K개 이어 붙인 문자열로 유일함을 확인할 수 있습니다.

C. 흑백 요리사

`math`, `number theory`
출제진 의도 – **Easy**

- ✓ 제출 16번, 정답 12명 (정답률 75.000%)
- ✓ 처음 푼 사람: **박동언**, 3분
- ✓ 출제자: **김태윤(ystaeyoon113)**

C. 흑백 요리사

- 어떤 시간 T가 주어진 조건을 만족하는지 생각해 봅시다.
- 먼저, 어떤 고기 i에 대해 T는 한 면을 온전히 구워야 하므로 A_i의 배수여야 합니다.
- 그리고, T는 앞면과 뒷면을 각각 같은 횟수로 구워야 하므로, $2A_i$의 배수여야 합니다.
- 따라서, T는 모든 고기에 대해 $2A_i$의 배수여야 합니다. 즉, T는 모든 $2A_i$의 공배수여야 합니다.
- 이중, 가장 최소가 되는 T는 모든 $2A_i$들의 최소공배수입니다.

C. 흑백 요리사

- ✓ 이때, 두 숫자 a와 b의 최소 공배수 $lcm(a,b)$를 구하는 방법은 다음과 같습니다.
- ✓ 먼저, a와 b의 최대공약수 $gcd(a,b)$를 유클리드 알고리즘으로 구할 수 있습니다.
- ✓ 이후, $gcd(a,b) * lcm(a,b) = ab$라는 수학적인 사실을 이용하여, $lcm(a,b)$를 구할 수 있습니다.
- ✓ 그런데, 세 개 이상의 숫자들의 lcm은 어떻게 구할까요? 이는 다음 사실로부터 구할 수 있습니다. $lcm(a,b,c) = lcm(lcm(a,b),c)$
- ✓ 즉, 세 개 이상의 숫자들의 lcm은 먼저 2가지 숫자의 lcm을 구하고, 그 결과와 나머지 하나 숫자의 lcm을 취하면 됩니다. 이를 이용하면 N개 숫자의 lcm도 같은 원리로 구할 수 있습니다.
- ✓ 추가로, 유클리드 알고리즘은 대부분의 언어에 gcd 함수를 이용하여 구현되어 있습니다. 따라서 직접 구현해서 사용해도 좋고, 라이브러리에 내장된 함수를 사용해도 좋습니다.

D. 게임 오브 데쓰 (Easy)

BFS, DP
출제진 의도 – **Medium**

- ✓ 제출 7번, 정답 7명 (정답률 100.000%)
- ✓ 처음 푼 사람: **김지호**, 11분
- ✓ 출제자: **김형진 (plast)**

D. 게임 오브 데쓰 (Easy)

- 만취되어 있는 사람은 누구에게 전달할지 모르기 때문에, 모든 상황을 가정해야 하는 이 문제 특성상 둘 모두에게 전달할 수 있다고 보아도 무방합니다.
- 이렇게 생각하면, 각 턴에 술래는 자신이 지목한 사람들 모두에게 다음 턴에 술래를 넘길 수 있습니다. 즉, 술래일 가능성이 하나라도 있는 모든 플레이어들은 다음 턴에 자신이 지목할 수 있는 모든 사람에게 술래를 넘길 수 있게 됩니다.
- i번째 턴에 j번째 플레이어가 술래일 가능성이 있는 경우, i + 1번째 턴에 L[j], R[j]번째 플레이어는 모두 술래가 될 가능성이 있다고 볼 수 있습니다.
- i번째 턴에 j번째 플레이어가 술래일 가능성이 있으면 1, 없으면 0과 같이 상태를 정의하면 이 문제를 일종의 그래프 문제로 모델링할 수 있게 됩니다.

D. 게임 오브 데쓰 (Easy)

- ✓ 초기에는 0번째 턴에 1번 플레이어만 술래일 가능성이 있습니다.
- ✓ i번째 턴에는, 각각의 j번째 플레이어에 대해서, 만약 술래일 가능성이 있다면 i + 1번째 턴의 L[j], R[j] 번째 플레이어는 술래일 가능성이 있도록 표시를 해둡니다.
- ✓ 이런 식으로 생각하면 각 턴에 어느 플레이어가 술래일 가능성이 있는지 정보를 정리할 수 있게 됩니다.
- ✓ 10 - 99번째 턴에 대해서, 1번째 플레이어가 술래일 가능성이 없는 턴을 찾고 출력합니다.

E. 출구가 바뀌는 미궁

`shortest path`, `math`

출제진 의도 – **Medium**

- 제출 1번, 정답 1명 (정답률 100.000%)
- 처음 푼 사람: **이재열**, 22분
- 출제자: **김형진 (plast)**

E. 출구가 바뀌는 미궁

- 1번 정점에서 출발해 각 정점까지 도달하는 최단 시간을 $D(i)$ 로 정의합시다.
- 각 정점까지의 최단 시간은 dijkstra algorithm을 이용해 $O(MlogM)$ 에 구할 수 있습니다.
- 형진이가 E_j 라는 출구를 이용하기로 결정했을 때, 다음 전략이 최적의 전략입니다.
- 먼저 E_j 출구에 최단 경로를 이용해 도달합니다. 이후 해당 정점에서 기다리다가, 출구가 열리자마자 빠져나옵니다.

E. 출구가 바뀌는 미궁

- E_j 출구는 $[(j-1)K, jK)$ 초에 열리고, $[(j-1)K + XK, jK + XK)$ 초에 열리고... 이후 XK 초마다 반복됩니다. 즉, $[(j-1)K, jK)\ (mod\ XK)$ 초마다 열리게 됩니다.
- E_j 출구에 최단 경로로 도달하는 시간을 t 라고 합시다. $t = D(E_j)$ 입니다.
- 이때 형진이는 $t \leq T$ 이고, $(j-1)K \leq T\ mod\ XK < jK$ 인 최소 T 를 구하고 싶습니다.
- 편의상 $t\ mod\ XK = q$ 로 정의하겠습니다.
- 만약 $(j-1)K \leq q < jK$ 이면 $T = t$ 입니다.
- 만약 $q < (j-1)K$ 이면 $T = t + (j-1)K - q$ 입니다.
- 만약 $q \geq jK$ 이면 $T = t + (j-1)K - q + XK$ 입니다. 왜냐하면 다음 출구가 열리는 시간까지 기다려야 하기 때문입니다.

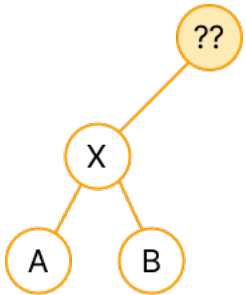
F. 트리의 루트를 찾아라

`tree`, `dfs`, `bfs`

출제진 의도 – **Medium**

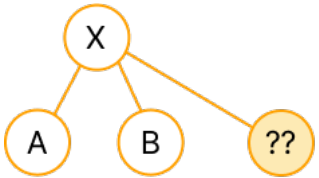
- 제출 0번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: —, –분
- 출제자: **김형진(plast)**

F. 트리의 루트를 찾아라



- 문제 상황을 생각해 보면, 다음과 같이 색칠한 정점은 LCA(A, B) = X를 만족하는 루트가 됩니다.
- 위 그림을 조금 돌려서 생각해 봅시다.

F. 트리의 루트를 찾아라



- 색칠한 정점들은 X를 루트로 두었을 때, A와 B의 정점이 포함된 서브트리가 아닌 나머지 서브트리에 있는 모든 정점이 됩니다.
- 문제에서 LCA(A, B) = X 를 만족하는 루트가 항상 존재한다고 하기 때문에, 예외 상황을 고려하지 않고 X번 정점과 해당하는 조건의 서브트리를 모두 구하면 됩니다.

F. 트리의 루트를 찾아라

- ✓ A, B 정점에서 각각 DFS / BFS를 진행해 각 정점간의 거리를 미리 구해놓습니다.
- ✓ X번 정점에서 시작해 A, B 정점 모두와 거리가 멀어지는 방향으로만 트리를 탐색하면, 방문한 정점들의 개수가 정답이 됩니다.

G. 게임 오브 데쓰 (Hard)

DP

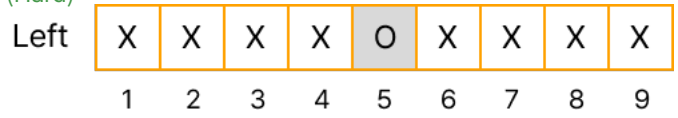
출제진 의도 – **Hard**

- 제출 0번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: —, –분
- 출제자: **김형진(plast)**

G. 게임 오브 데쓰 (Hard)

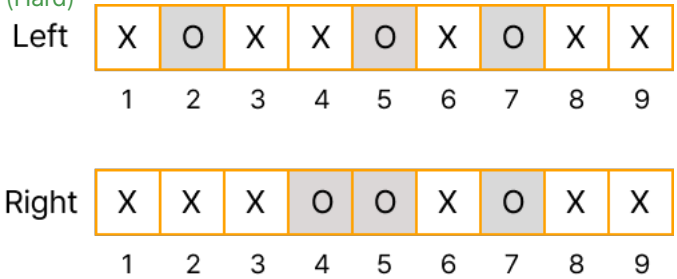
- ✓ 이전의 게임 오브 데쓰 (Easy) 문제를 생각해 봅시다. 게임 오브 데쓰 (Easy)와 (Hard)의 다른 점은 형진이가 자신이 걸리지 않도록 선택할 수 있다는 점입니다.
- ✓ 즉, 형진이가 어느 쪽을 선택하더라도 자신에게 걸릴 가능성이 있는지를 판단해야 합니다.
- ✓ 우선 첫 번째 플레이어가 왼쪽을 지목하는 경우, 오른쪽을 지목하는 경우 각각을 기준으로 어느 턴에 형진이 자기 자신이 걸릴 수 있는지 계산해 둡니다.

G. 게임 오브 데쓰 (Hard)



- 쉽게 확인할 수 있는 것은, 예를 들어 왼쪽을 지목하더라도 5번째 턴에 형진이가 걸릴 수 있게되며 동시에 오른쪽을 지목하더라도 5번째 턴에 형진이가 걸릴 수 있게 되면 형진이는 어느 쪽을 지목하더라도 5번째 턴에 자기 자신이 걸리게 됩니다.
- 즉, 왼쪽을 지목하더라도 오른쪽을 지목하더라도 i번째 턴에 자기 자신이 걸리면, 이는 형진이가 무슨 짓을 하더라도 i번째 턴에는 자기 자신이 지목당할 수 있습니다.

G. 게임 오브 데쓰 (Hard)



- 두 번째로, 예를 들어 다음과 같은 상황을 가정해봅시다. 여기서 9번째 턴에 주목해 봅시다.
- 형진이가 왼쪽을 지목하면 2턴만에 자기 자신의 차례로 돌아올 수 있습니다. 이 경우 7턴이 남게 되며, 이제 형진이는 무슨 짓을 하더라도 자기 자신이 걸릴 가능성이 생깁니다.
- 형진이가 오른쪽을 지목하더라도 4턴만에 자기 자신의 차례로 돌아올 수 있습니다. 이 경우 5턴이 남게 되며, 이제 형진이는 무슨 짓을 하더라도 자기 자신이 걸릴 가능성이 생깁니다.

G. 게임 오브 데쓰 (Hard)

- ✓ 따라서, 왼쪽, 오른쪽 어느 쪽을 고르더라도 i번째 턴에 형진이가 걸릴 가능성이 있는지 여부를 A[i]라고 하겠습니다. A[i]는 곧 j + k = i, l + m = i인 어떠한 j, k, l, m에 대해, A[j] = true이며, Left[k] = true이며, A[l] = true이며, Right[m] = true라면 A[i]는 true라고 볼 수 있게 됩니다.

H. 오장원전

`game theory`, `dp`, `greedy`, `binary search`

출제진 의도 – **Hard**

- 제출 0번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: —, –분
- 출제자: **김태윤(ystaeyoon113)**

H. 오장원전

- ✓ $dp(K, N)$ 를 제갈량이 N 만큼 보급해야 하고, 사마의가 K 번 방해할 수 있을 때 제갈량이 목표 달성을 위해 사용해야 하는 최소 cost라고 정의합시다.
- ✓ 이때 제갈량이 i 만큼 보급을 시도한다면 사마의는 이를 방해할 수도 있고, 방해하지 않을 수도 있습니다. 사마의는 2가지 선택지의 결과 중 제갈량에게 더 많은 cost를 발생시키는 선택지를 고를 것입니다.
- ✓ 사마의가 방해한다면 cost는 $dp(K-1, N) + i + C$ 이고, 방해하지 않는다면 cost는 $dp(K, N-i) + i + C$ 입니다. 중복되는 항이 보입니다.
- ✓ 따라서 제갈량이 i 만큼의 보급을 시도한다면 cost는 $max(dp(K-1, N), dp(K, N-i)) + i + C$ 입니다.

- 반면에, 제갈량 또한 사마의가 그렇게 행동할 것을 알고 있기 때문에, 자신이 i 만큼의 보급을 시도할 때의 cost를 계산할 수 있습니다. 따라서 제갈량은 자신이 시도할 수 있는 모든 i 중에서 최소의 cost를 주는 i를 선택할 것입니다.
- 이를 통해 $dp(K, N)$의 결과를 수식으로 나타낸다면 다음과 같습니다.
- $dp(K, N) = \underset{i:1 \to N}{min}\{max(dp(K-1, N), dp(K, N-i)) + i + C\}$
- 위 수식을 그대로 dynamic programming을 이용해 구현하면 $O(N^2K)$에 값을 계산할 수 있지만, 문제 제한이 크므로 추가적인 관찰이 필요합니다.

H. 오장원전

- 직관적으로 생각했을 때, 제갈량이 무리하게 보급을 시도하면 사마의가 방해하고, 제갈량이 다소 소심하게 보급을 시도하면 사마의가 방해하는 카드를 아끼는게 이득일 것 같습니다.
- 실제로 이 생각은 문제의 복잡도를 줄이는 핵심적인 생각이고, 수식으로 증명할 수 있는 옳은 주장입니다.
- 먼저 $dp(K, N) \leq dp(K, N+1)$ 은 자명합니다. 제갈량이 총 $N+1$ 의 보급을 시도할 때 N 의 보급을 시도하는 전략보다 작은 cost로 성공할 수는 없습니다.
- $\min_{i:1 \to N} \{max(dp(K-1, N), dp(K, N-i)) + i + C\}$ 이라는 수식을 생각해 봅시다.
  $dp(K-1, N)$ 은 i 와 무관하므로 고정된 값 Y 를 가집니다. 반면 $dp(K, N-i)$ 는 i 에 따라 단조감소합니다.

H. 오장원전

- ✓ 따라서 어떤 i에 대해서 $Y \geq dp(K, N-i)$ 이게 된다면 그것보다 큰 i에 대해서는 모두 Y가 크게 됩니다. 즉, 어떤 기준이 되는 i 이상으로는 사마의가 방해하는 전략이 우월전략이 됩니다. 그런 i를 i^*라고 부르겠습니다.
- ✓ 제갈량은 i^* 보다 큰 보급을 시도하는 것이 무의미합니다. 어차피 사마의는 방해할 것이고, 어차피 방해될 것이라면 cost라도 작은 i^*를 시도하는 것이 우월하기 때문입니다.
- ✓ i^*는 이분탐색으로 구할 수 있고, 제갈량에게 i^*를 시도하는 전략은 사마의의 방해 카드를 소모하는 하나의 전략이 됩니다.

H. 오장원전

- 그렇다면 i^* 보다 작은 값에 대해서는 어떤 i를 시도해야 할까요? 이때는 사마의가 무조건 방해하지 않을 것이기 때문에, 가능한 큰 i를 시도하는 전략이 좋아보입니다.
- $i < j$라고 합시다. i를 시도하는 것과 j를 시도하는 전략의 cost를 비교하려면, $dp(K, N-i) + i + C$와 $dp(K, N-j) + j + C$를 비교해야 합니다.
- 그런데 이 비교는 쉽지 않기 때문에, i를 쓰는 전략보다 $i+1$을 쓰는 전략이 우월하다는 것을 보여서 결론적으로는 i를 쓰는 전략보다 j를 쓰는 전략이 우월하다는 것을 귀납적으로 증명할 수 있습니다.
- 양변을 적당히 소거하면 $dp(K, N-i)$와 $dp(K, N-i-1) + 1$을 비교해야 합니다. 그런데 $N-i$에서 $N-i-1$로 가기 위해 반드시 1 이상의 cost가 필요함을 생각하면, $dp(K, N-i) \geq dp(K, N-i-1) + 1$임을 어렵지 않게 보일 수 있습니다.

H. 오장원전

- 결국, 사마의가 방해하지 않을 때는 가능한 가장 많은 보급을 시도하는 것이 우월전략입니다.
- 따라서 주어진 상황에서 제갈량의 우월한 2가지 선택지는 i^* 를 시도해서 사마의의 방해카드를 소모시키거나, 혹은 $i^* - 1$ 을 시도해서 방해받지 않으면서 가장 많이 보급을 시도하는 것입니다.
- 제갈량은 둘 중에서 작은 cost를 주는 선택지를 시도하게 됩니다.
- 각 $dp(K, N)$ 마다 i^* 를 이분탐색으로 구할 수 있습니다. 이는 $O(NKlogN)$ 의 복잡도로 동작합니다.
- 혹은 i^* 의 단조성까지 관찰하면 투 포인터를 이용하여 $O(NK)$ 의 복잡도로 동작합니다.
- 이 문제는 이분탐색을 활용한 log 가 붙는 풀이까지는 정답으로 인정했습니다.

I. 돌 게임

`ad-hoc, game theory`
출제진 의도 – **Hard**

- 제출 0번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: —, –분
- 출제자: **김형진(plast)**

- ✓ 순서대로 2개씩의 돌을 짝지어주었을 때, 보드판의 상태를 (1번째 쌍의 돌 사이의 거리, 2번째 쌍의 돌 사이의 거리, ..., K번째 쌍의 돌 사이의 거리) 형태로 표현을 해보겠습니다.
- ✓ 각 쌍의 거리가 전부 0일 때 후공에게 필승 전략이 있습니다. 후공 입장에서, 무조건 선공의 제일 최근 move에 갖다 붙이는 전략을 취하면 선공은 반드시 돌을 한 쪽 방향으로 움직여야 하기 때문에 게임은 반드시 종료하고 후공이 더 이상 시행할 수 없는 상태인 BWBW.... 같은 모양은 후공의 차례에는 절대 생기지 않기 때문에 후공이 승리합니다.
- ✓ 따라서 게임의 목표를, 각 쌍 사이의 거리를 전부 0으로 만드는 걸로 바꿔도 게임의 결과는 변하지 않습니다.

- ✓ 돌 쌍의 거리를 늘리는 시행은 무의미합니다. 다음 플레이어는 거리를 원상복구시킬 수 있으며, 마찬가지로 거리를 늘리는 시행은 반드시 돌을 한 쪽 방향으로 움직여야 하기 때문에 반드시 돌 쌍의 거리를 더이상 못 늘리는 순간이 오게 됩니다.
- ✓ 돌 쌍의 거리를 가까워지게 하는 시행을 생각해 봅시다. 두 돌의 쌍의 거리를 줄이는 과정은 하나의 쌍에 대해 원하는 만큼 거리를 줄여나갈 수 있는데, 이 연산을 님게임과 완전히 같은 문제로 모델링 할 수 있게 됩니다. 님게임에서 하나의 무더기에 원하는 만큼 돌을 빼내는 것과 완전히 똑같은 연산이기 때문입니다.

- 님 게임에서의 필승전략은 스프라그-그런디 정리로 잘 알려져 있지만, 간단하게나마 설명하겠습니다. 모든 더미의 돌의 개수의 xor을 0으로 만든 사람이 승리합니다.
- 님 게임에서 모든 더미의 돌의 개수의 xor이 0만 아니면 무조건 xor을 0으로 만들 수 있습니다.
- 전체 xor한 값 S의 max bit를 잡고, 그 bit가 켜져있는 원소 x를 잡아서 x를 $x \oplus S$로 낮추면 됩니다.
- 따라서, xor을 0으로 만든 사람은 항상 돌의 xor을 0을 유지할 수 있고, 최종적으로 모든 더미의 돌이 0이 되도록 만들 수 있습니다.

- ✓ 따라서, 님 게임의 원리를 이 문제의 모델링에 적용하면, 거리의 xor이 0이 아닌 경우, 선공은 xor을 0으로 만들도록 거리를 좁혀나가는 전략을 취하면 반드시 승리하게 됩니다.
- ✓ 만약 거리의 xor이 0인 경우, 후공은 선공이 거리를 늘린다면 거리를 원상복구시키고, 선공이 거리를 줄인다면 거리를 더 줄여서 xor을 0으로 만들게 되면 반드시 승리하게 됩니다.

J. 빨간점, 파란점 3

`geometry`, `convexhull`, `binary_search`, `sweeping`

출제진 의도 – **Challenging**

- 제출 0번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: —, –분
- 출제자: **이국렬 (lky7674)**

J. 빨간점, 파란점 3

- ✓ 상당히 어려운 문제입니다.
- ✓ ~~거하 문제 출제 너무 고통스러워요. 살려주세요.~~

J. 빨간점, 파란점 3

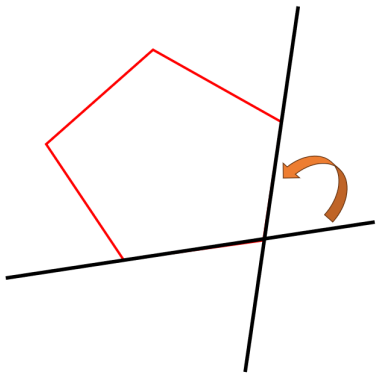
- ✓ 모든 빨간점들이 같은 반평면에 있어야 한다. → 빨간점들을 포함하는 최소 convexhull을 해당 반평면이 포함하고 있어야 한다.
- ✓ 따라서 한 convexhull을 포함하면서, 포한된 파란점들의 가중치의 합을 최소화하는 문제로 볼 수 있습니다.
- ✓ convexhull 안에 있는 파란점들은 무슨 짓을 해도 격리가 불가능하니 따로 제외합니다. (이는 이분탐색 또는 스위핑으로 체크할 수 있다.)

J. 빨간점, 파란점 3

- ✓ 반평면을 나누는 직선은 convexhull과 만나는 것이 이득.
- ✓ 만나지 않는 optimal한 직선이 있다면, 해당 직선을 (반평면에 포함된 파란점의 가중치가 증가하지 않도록) 움직여서 convexhull과 만나는 optimal한 직선을 만들 수 있음.
- ✓ 따라서 convexhull에 접한 직선만 고려하면 된다.

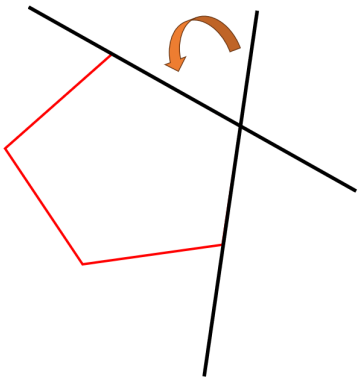
J. 빨간점, 파란점 3

- 다음과 같이 convexhull에 접하면서 직선을 돌리는 방법을 생각해볼 수 있다.



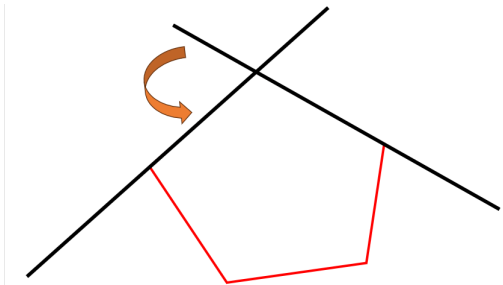
J. 빨간점, 파란점 3

✓ 다음과 같이 convexhull에 접하면서 직선을 돌리는 방법을 생각해볼 수 있다.



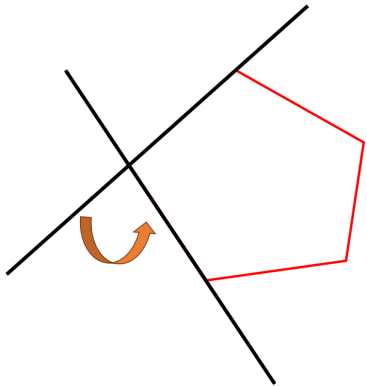
J. 빨간점, 파란점 3

✓ 다음과 같이 convexhull에 접하면서 직선을 돌리는 방법을 생각해볼 수 있다.



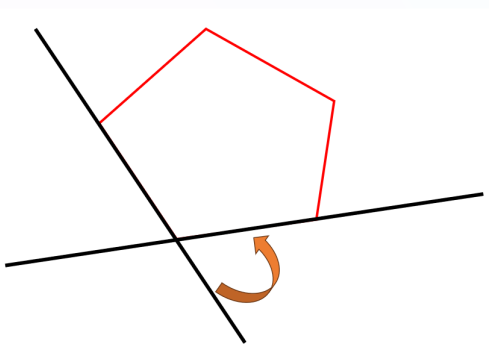
J. 빨간점, 파란점 3

- ✓ 다음과 같이 convexhull에 접하면서 직선을 돌리는 방법을 생각해볼 수 있다.



J. 빨간점, 파란점 3

✓ 다음과 같이 convexhull에 접하면서 직선을 돌리는 방법을 생각해볼 수 있다.



J. 빨간점, 파란점 3

- convexhull의 각 점을 기준으로 직선을 밖으로 돌릴때, 만나는 파란점들을 동적 배열로 저장. 저장한 점들을 각도 기준으로 정렬.
- 이때, 각 파란점이 convexhull에 접선을 그었을 때의 두 접점에서 직선을 회전시킬 때 만나는 것을 알 수 있다.
- 각 파란점에 대한 접점은 이분 탐색 + CCW로 $O(\log N)$에 구할 수 있다.
- 스위핑을 하면서 만나는 점들의 포함/배제 여부를 바꾸면서 격리 가능한 파란점의 가중치 합의 최댓값을 구하면 된다.

J. 빨간점, 파란점 3

- ✓ 시간복잡도
  - convexhull 구하기 - $O(N \log N)$
  - 각 파란점의 convexhull 포함 여부 + 접점 구하기 - $O(M \log N)$
  - convexhull의 각 점에 대해 각도 정렬 후 스위핑 - $O(N + M \log M)$
- ✓ 주의사항
  - $N = 1$인 경우는 따로 처리해줘야 한다.
  - 이는 그냥 각도 정렬 후 스위핑을 해주면 된다.
  - 구현 방법에 따라 $N = 2$인 경우도 따로 빼줘야 하는 상황이 생길 수 있다.

J. 빨간점, 파란점 3

- ✓ 관련 추천 문제
  - 22962 신촌방위본부 (**P1**)
  - 24895 다트 (**D4**)
  - 12801 중계신호 (**D2**)