Semi-Game Cup 2021

Official Solutions

by

stonejjun 게임 공장

2021년 8월 8일

문제		의도한 난이도	출제자
A	인증된 쉬운 게임	Easy	stonejjun03
B	루미너스와 모험 중 마주친 퍼즐게임	Challenging!!	blackking26
C	조화로운 마법 농구 게임	Medium	stonejjun03
D	수식 알아맞히기 퀴즈 게임	Hard	ahgus89
E	초콜릿 쪼개기 게임	Medium	stonejjun03
F	흑왕과 어둠의 게임 대진표	Hard	stonejjun03
G	다리 건너기 게임	Medium	jhwest2
H	기둥과 성벽 디펜스 게임	Challenging	stonejjun03
I	고장난 계산기 (Calculator) 게임	Challenging	blackking26
J	고인물의 두번째 리듬게임	Hard	jhwest2

목차

A. 인증된 쉬운 게임	4
B. 루미너스와 모험 중 마주친 퍼즐게임	7
C. 조화로운 마법 농구 게임	56
D. 수식 알아맞히기 퀴즈 게임	62
E. 초콜릿 쪼개기 게임	69
F. 흑왕과 어둠의 게임 대진표	74
G. 다리 건너기 게임	80
H. 기둥과 성벽 디펜스 게임	86
I. 고장난 계산기 (Calculator) 게임	96
J. 고인물의 두번째 리듬게임	101

A. 인증된 쉬운 게임

game_theory
출제진 의도 – **Easy**

- 제출 546번, 정답 95명 (정답률 21.25%)
- 처음 푼 사람 : **kimjihoon**, 4분
- 출제자 : stonejjun

- 이러한 게임은 '상태'를 생각할 수 있습니다. 이 문제에서는 칠판에 쓰여진 수가 상태를 나타내게 됩니다.
- 그리고 이 게임에서 모든 상태는 선공이 이기는 상태와 후공이 이기는 상태로 분류할 수 있습니다.
- 어떤 상태에서 갈 수 있는 모든 상태가 선공이 이기는 상태이면 그 상태는 후공이 이기는 상태이며, 그렇지 않으면 선공이 이기는 상태인 것을 생각할 수 있습니다.
- 이 문제에서 상태의 변화는 약수를 더하는 과정을 의미하며 K 부터 내려가면서 DP를 채우는 방식으로 진행을 하면 됩니다.
- 1 부터 N 까지 약수를 미리 구하는 것으로 $\mathcal{O}\left(N \log N\right)$ 의 시간복잡도로 문제를 해결할 수 있습니다.

- 위의 과정을 통해서 각 수별로 누가 승리하는 지를 출력해보면 거의 Ringo가 승리한다는 것을 알 수 있습니다.
- 실제로 $K = 2, 6$을 제외한 범위내의 나머지 모든 수에서 Ringo가 승리하게 되고, 이를 이용하여 코드를 간단하게 작성할 수도 있습니다.

B. 루미너스와 모험 중 마주친 퍼즐게임

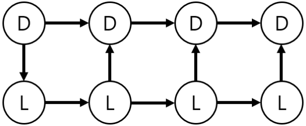
ad_hoc, constructive
출제진 의도 – **Challenging**

- 제출 3번, 정답 0명 (정답률 0.00%)
- 처음 푼 사람 : None
- 출제자 : blackking26
- **주의!**

- $N \times M$ 크기의 게임판의 각 칸에 빛의 함정 또는 어둠의 함정이 있습니다.
- 어둠의 함정을 없애고 인접한 함정들의 속성을 뒤집는 시행을 할 수 있습니다.
- 적절한 순서로 시행을 실시해서 모든 함정을 없앨 수 있는 방법이 있는지 찾아야 합니다.

Part 0. 기우성

- 함정을 정점으로 하고 인접한 함정 사이에 간선을 그어서 그래프를 그려 봅시다.
- 모든 함정을 없앨 수 있는 방법이 있다고 합시다. 그런 방법 중 하나를 골라, 각 간선이 잇는 두 함정 중 먼저 없어지는 함정에서 나중에 없어지는 함정 쪽으로 간선의 방향을 정합니다.
- 각 함정의 반전횟수는 indegree와 같습니다. 따라서 빛의 함정 (이하 L) 의 indegree는 홀수여야 하고, 어둠의 함정 (이하 D) 의 indegree는 짝수여야 합니다.



- 간선의 수는 $N(M-1)+M(N-1)$ 이고 이 값과 L 의 기우성이 같아야 합니다.

- $N=(\text{홀수}), M=(\text{홀수})$ 일 때: $N(M-1)+M(N-1)$ 는 짝수이고, D 는 홀수 개 있어야 합니다.
- $N=(\text{홀수}), M=(\text{짝수})$ 일 때: $N(M-1)+M(N-1)$ 는 홀수이고, D 는 홀수 개 있어야 합니다.
- $N=(\text{짝수}), M=(\text{짝수})$ 일 때: $N(M-1)+M(N-1)$ 는 짝수이고, D 는 짝수 개 있어야 합니다.

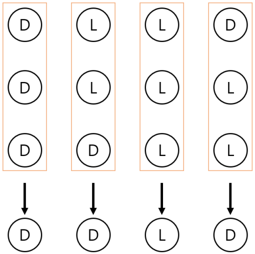
- 위 기우성 조건을 만족하지 않으면 반드시 Run입니다. 이제부터는 기우성 조건을 만족하는 경우만 생각합시다.

Part 1. $1 \times N$

- 수학적 귀납법을 사용합시다. 1×1일 때는 기우성 조건에 의해 유일한 함정이 D이므로 그냥 지우면 됩니다.
- 기우성 조건에 의해 D가 적어도 하나 존재합니다. 따라서 가장 왼쪽에 있는 D를 선택할 수 있고, 가장 왼쪽에 있는 D를 지우는 것으로 시작해서 그 왼쪽에 있는 L들을 전부 없애면 $1 \times N' \, (N' < N)$이 남습니다.
- 규칙대로 함정을 제거할 때 (남은 간선 수) $+$ (L의 개수)의 기우성은 불변량입니다. 따라서 남은 $1 \times N'$ 게임판도 기우성 조건을 만족함은 자명합니다.

Part 2. $(\text{홀수}) \times N$

- Part 1에서 확인한 바에 따르면, D가 홀수 개 있는 열은 완전히 지울 수 있습니다. 이런 열을 $D-$열이라고 합시다. 반대로 D가 짝수 개 있는 열은 $L-$열이라고 합시다.



- $D-$ 열을 완전히 지우고 나면 주변의 열은 어떻게 될까요? 열의 길이가 홀수이므로 지워진 열의 양옆에 있던 열 중 $D-$ 열이었던 것은 $L-$ 열로, $L-$ 열이었던 것은 $D-$ 열로 변하게 됩니다.
- $D-$ 열을 D로, $L-$ 열을 L로 보면, 놀랍게도 이 상황은 $1 \times N$에서 문제를 푸는 것과 완전히 동일합니다!
- 기우성 조건에 의해 D는 홀수 개이므로 $D-$ 열도 홀수 개입니다. 따라서 Part 1의 방법대로 각 열을 제거할 순서를 정해주면 문제를 풀 수 있습니다.

- 물론 $N \times$ (홀수) 의 경우에도 문제를 풀 수 있습니다.

Part 3. $(\text{짝수}) \times (\text{짝수})$

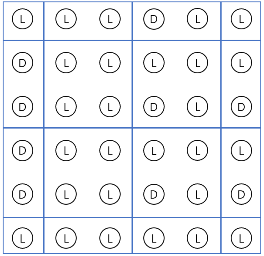
- Part 2에서 확인했듯이 $(\text{홀수}) \times N$ 의 경우에는 기우성 조건만 만족하면 모두 `Solve`입니다. 그러나 $(\text{짝수}) \times (\text{짝수})$ 의 경우에는 기우성 조건을 만족하더라도 Run일 수 있습니다.
- 예를 들어 모든 함정이 L 일 경우는 기우성 조건을 만족하지만 자명하게 Run이고, 모든 함정이 L 이 아니더라도 다음과 같은 경우가 Run입니다.

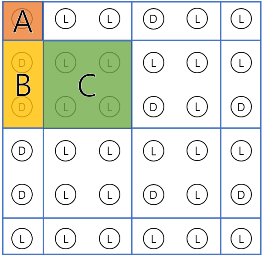
LDDL

LLLL

B. 루미너스와 모험 중 마주친 퍼즐게임

- 앞으로의 논의에서는 다음 그림과 같이 구역을 나눠서 생각하겠습니다.

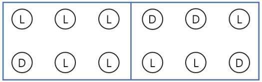


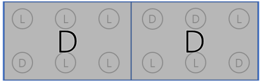


- $2i$ 번째 행과 $2i+1$ 번째 행을 묶고, $2i$ 번째 열과 $2i+1$ 번째 열을 묶으면 왼쪽 그림과 같이 구역이 나누어집니다.
- 또 오른쪽 그림과 같이 한 구역에 함정이 1개 들어가는 구역을 $A-$ 구역, 2개 들어가는 구역을 $B-$ 구역, 4개 들어가는 구역을 $C-$ 구역 이라고 합시다.

Part 3-1. $(\text{홀수}) \times N$ 두 개로 나누기

- 전체 게임판을 적당한 홀수 k에 대해 $N \times k$와 $N \times (M - k)$로 나누었을 때 양쪽에 D가 홀수 개씩 포함되는 경우를 생각합시다.
- Part 2에 의해 $N \times k$ 게임판을 없앨 수 있고, N이 짝수이므로 $N \times k$ 게임판을 없앤 후에도 $N \times (M - k)$ 게임판에 포함된 D의 개수는 여전히 홀수 개입니다. 따라서 $N \times (M - k)$ 게임판도 Part 2의 방법으로 마저 없앨 수 있습니다.

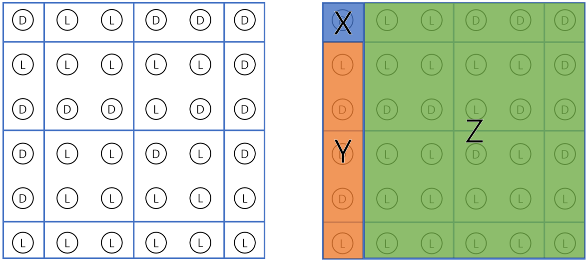




- 물론 $k \times M$과 $(N - k) \times M$으로 게임판을 나눌 수 있는 경우에도 문제를 해결할 수 있습니다.
- 이제부터는 Part 3-1의 방법으로 문제를 해결할 수 없는 경우에 대해서만 생각합시다.

Part 3-2. $A-$ 구역에 D가 있을 때

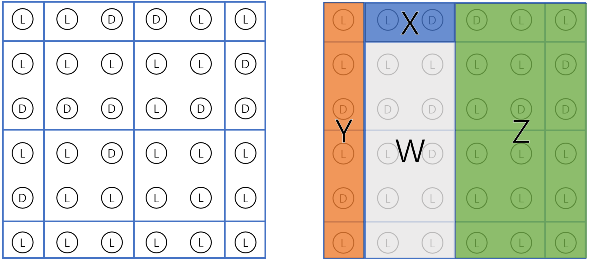
✓ 다음 그림과 같이 구역을 나눠서 생각합시다.



- 먼저 X 를 제거합니다.
- Part 3-1로 문제를 해결할 수 없었으므로 최초에 Z 에 포함된 D 의 개수는 짝수입니다. 여기서 X 를 제거하면 Z 에 포함된 D 의 개수는 홀수가 되므로 Part 2의 방법을 통해 Z 를 제거할 수 있습니다.
- 마지막으로 Y 가 기우성 조건을 만족함은 불변량에 의해 보장되므로 Y 또한 제거할 수 있습니다.
- 이제부터는 Part 3-2의 방법으로 문제를 해결할 수 없는 경우에 대해서만 생각합시다.

Part 3-3. $B-$구역에 L과 D가 1개씩 있을 때

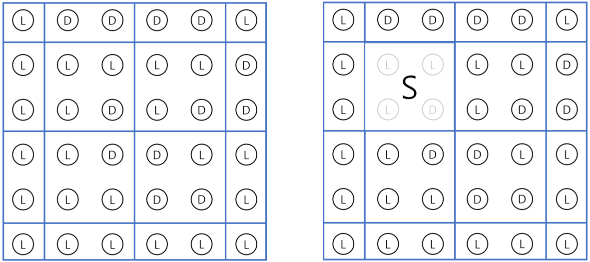
- 다음 그림과 같이 구역을 나눠서 생각합시다.



- 먼저 X 를 제거합니다.
- Part 3-1로 문제를 해결할 수 없었으므로 최초에 Y, Z 에 포함된 D 의 개수는 각각 짝수입니다. 여기서 X 를 제거하면 Y, Z 에 포함된 D 의 개수는 각각 홀수가 되므로 Part 2의 방법을 통해 Y, Z 를 제거할 수 있습니다.
- 마지막으로 W 가 기우성 조건을 만족함은 불변량에 의해 보장되므로 W 또한 제거할 수 있습니다.
- 이제부터는 Part 3-3의 방법으로 문제를 해결할 수 없는 경우에 대해서만 생각합시다.

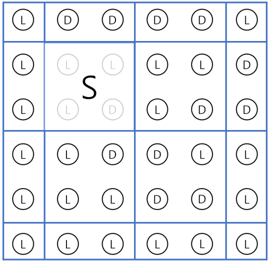
Part 3-4. $C-$구역에 D가 홀수 개 있을 때

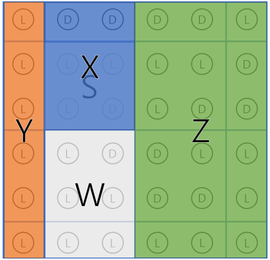
✓ D가 홀수 개 포함된 $C-$구역 중 가장 위에 있는 구역을 S라고 합시다.



B. 루미너스와 모험 중 마주친 퍼즐게임

- S 보다 위에 있는 구역에는 모두 D 가 짝수 개 속합니다.
- S 보다 위에 있는 구역 중 $C-$ 구역에는 S 의 정의에 의해서 D 가 짝수 개만 속하고, $A-$ 구역이나 $B-$ 구역에는 Part 3-2,3으로 문제를 해결할 수 없었기 때문에 D 가 짝수 개만 속합니다.



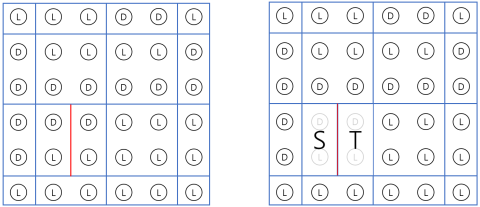


- 따라서 위 그림과 같이 구역을 나누면 X 에는 홀수 개의 D 가 속하게 됩니다.

- 먼저 Part 2의 방법을 통해 X 를 제거합니다.
- Part 3-1로 문제를 해결할 수 없었으므로 최초에 Y, Z 에 포함된 D 의 개수는 각각 짝수입니다. 여기서 X 를 제거하면 Y, Z 에 포함된 D 의 개수는 각각 홀수가 되므로 Part 2의 방법을 통해 Y, Z 를 제거할 수 있습니다.
- 마지막으로 W 가 기우성 조건을 만족함은 불변량에 의해 보장되므로 W 또한 제거할 수 있습니다.

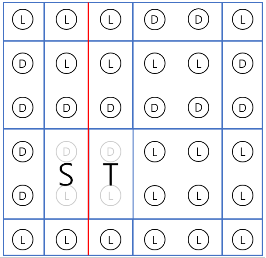
Part 3-5. $C-$ 구역에 D가 2개 있을 때

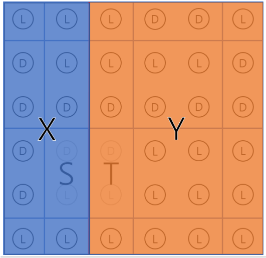
- D가 2개 포함 된 $C-$ 구역에서 D는 각 행에 하나씩 포함되거나, 각 열에 하나씩 포함되거나, 혹은 둘 다입니다.
- D가 각 행에 하나씩 포함되는 경우에는 $C-$ 구역을 가로로 자르고, D가 각 열에 하나씩 포함되는 경우에는 $C-$ 구역을 세로로 자릅니다.



- 그러면 위 그림과 같이 S, T에 D가 각각 한 개씩 들어가게 됩니다.

- 자르는 선을 연장하여 게임판 전체를 두 부분으로 나누어 봅시다.





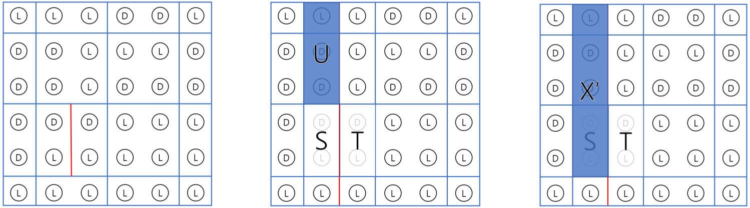
- 이제 X와 Y에 포함된 D가 각각 짝수 개일 때와 홀수 개일 때로 다시 경우를 나누어 생각합시다.

Part 3-5-1. X, Y **에** D **가 각각 짝수 개씩 있을 때**

- X 를 전체 게임판으로 생각하면 S 는 $B-$ 구역이 됩니다.
- X 는 $(\text{짝수}) \times (\text{짝수})$ 게임판이고, D 를 홀수 개 포함하며 S 에 L 과 D 가 1개씩 있으므로 Part 3-3의 방법을 통해 X 를 제거할 수 있습니다.
- X 를 제거한 후에도 T 에 L 과 D 가 1개씩 있게 되고, 같은 방법으로 Y 를 제거할 수 있습니다.

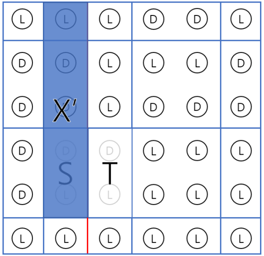
Part 3-5-2. X, Y 에 D 가 각각 홀수 개씩 있을 때

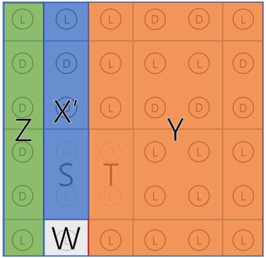
- 다음 그림과 같이 S 위쪽의 구역을 U 라 합시다.



- 만약 U 에 D 가 홀수 개 포함되어 있다면 X' 을 U 로 정의하고, U 에 D 가 짝수 개 포함되어 있다면 X' 을 $U + S$ 로 정의합니다.

- 둘 중 어느 경우든, X' 의 길이는 홀수이며, D 를 홀수 개 포함하고 있고, S 를 완전히 포함하거나 완전히 포함하지 않습니다.
- 이제 다음 그림과 같이 구역을 나눠서 생각합시다.

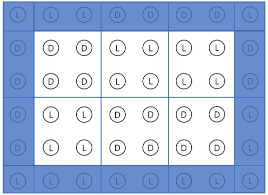


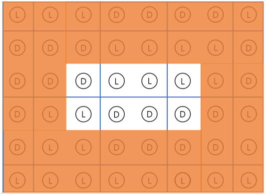


- 먼저 Part 1의 방법을 통해 X'을 제거합니다.
- 최초에 Y에 포함된 D의 개수는 홀수입니다. X'의 길이가 홀수이므로 X'를 제거한 후 Y에 포함된 D의 개수는 홀수가 됩니다.
- 또 X'가 S를 완전히 포함하거나 완전히 포함하지 않으므로 X'을 제거한 뒤에도 T에는 L과 D가 1개씩 있게 됩니다. 따라서 Part 3-2의 방법을 통해 Y를 제거할 수 있습니다.
- Part 3-1로 문제를 해결할 수 없었으므로 최초에 Z에 포함된 D의 개수는 짝수입니다. 여기서 X'를 제거하면 Z에 포함된 D의 개수는 홀수가 되므로 Part 2의 방법을 통해 Z를 제거할 수 있습니다.
- 마지막으로 W가 기우성 조건을 만족함은 불변량에 의해 보장되므로 W 또한 제거할 수 있습니다.

Part 3-6-1. 모서리

- 다음 case를 설명하기 위해 잠시 다른 이야기를 하겠습니다.
- 왼쪽 그림과 같이 $A -$ 구역과 $B -$ 구역 들의 합집합을 $1 -$ 모서리라고 합시다.
- 모서리를 제외한 나머지 부분을 '내부'라고 합시다.
- 오른쪽 그림과 같이 $1 -$ 모서리와, 내부의 $1 -$ 모서리의 합집합을 $2 -$ 모서리라고 합시다.





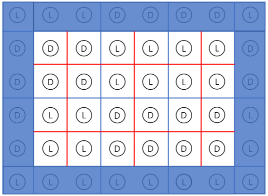
- Part 3-2,3 으로 문제를 해결할 수 없었으므로 $1-$ 모서리에 포함된 D의 개수는 짝수입니다.
- $1-$ 모서리에 포함된 D의 개수가 0 초과의 짝수일 경우에는 다음 방법으로 $1-$ 모서리를 제거할 수 있습니다.

- $1-$ 모서리에 포함된 아무 D를 골라서 제거하면 $1-$ 모서리에 포함된 D의 개수는 홀수 개가 됩니다.
- 또한 $1-$ 모서리의 남은 함정은 함정이 직선형으로 배열되어 있을 때와 같은 연결구조를 가지게 됩니다. 따라서 Part 1의 방법을 통해 남은 함정을 제거할 수 있습니다.

Part 3-6-2. 기본 조건

- 설명에 앞서 Part 3-1~5로 문제를 해결할 수 없는 조건을 정리해봅시다.
  1. A – 구역에는 L만 포함되어야 합니다.
  2. B – 구역에는 동일한 속성의 함정이 포함되어야 합니다.
  3. C – 구역에는 동일한 속성의 함정이 포함되어야 합니다.
- 게임판이 이 세 가지 조건 중 하나라도 만족하지 않는다면 반드시 `Solve`입니다.
- 이제 이 세 가지 조건을 '기본 조건'이라고 부르겠습니다.

- 다음 그림과 같이 내부를 하나의 게임판으로 생각해서 기본 조건을 만족하는지 확인할 수 있습니다.



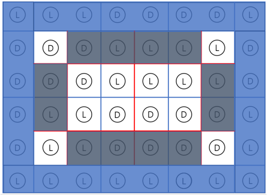
- 내부 게임판이 기본 조건을 만족하지 않는다면 내부를 제거할 수 있습니다.
- 따라서 내부와 $1-$ 모서리를 따로 처리하는 방식으로 문제를 해결할 수도 있을 것 같습니다.

Part 3-6-3. 새로운 전략

- 전체 게임판은 기본 조건을 만족하지만 내부 게임판은 기본 조건을 만족하지 않는다고 합시다.
- 우리는 Part 3-6 에서 $1-$ 모서리에 D 가 0 개 초과의 짝수 개만큼 포함되어 있으면 $1-$ 모서리를 없앨 수 있음을 확인했습니다.
- 전체 게임판이 기본 조건을 만족하기 때문에 $1-$ 모서리에는 D 가 짝수 개 포함되어 있습니다. 따라서 $1-$ 모서리에 D 가 하나라도 있다면 $1-$ 모서리를 제거할 수 있습니다.

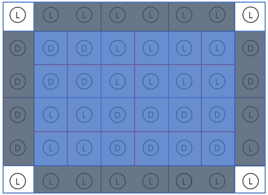
B. 루미너스와 모험 중 마주친 퍼즐게임

- 만약 $1-$ 모서리에 D가 하나라도 있다면, $1-$ 모서리를 먼저 제거합니다.
- $1-$ 모서리를 제거하고 나서 상태가 반전되는 함정은 다음 그림에 표시된 것과 같이 내부 게임판의 $B-$ 구역에 해당하는 함정들입니다.



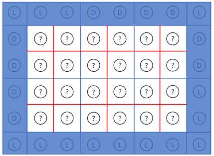
- 따라서 처음에 내부가 기본 조건을 만족하지 않았다면, $1-$ 모서리를 제거한 후에도 여전히 내부는 기본 조건을 만족하지 않을 것입니다.
- 따라서 내부를 마저 제거하고 문제를 해결할 수 있습니다.

- $1-$ 모서리에 D가 하나도 없는 경우에는 내부를 먼저 제거하면 됩니다.
- 내부를 제거하면 $1-$ 모서리에 D가 생기므로, 모서리를 마저 제거하고 문제를 해결할 수 있습니다.



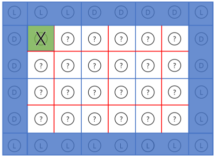
Part 3-6-4. $C-$구역에 D가 있을 때

- 이제 전체 게임판과 내부 게임판 중 하나라도 기본 조건을 만족하지 않는다면 문제를 해결할 수 있다는 것을 확인했습니다.
- "전체 게임판과 내부 게임판 모두 기본 조건을 만족한다" 라는 말은 어떤 의미일까요?
- 다음 게임판이 전체 게임판과 내부 게임판 모두 기본 조건을 만족하는 게임판이라고 합시다.

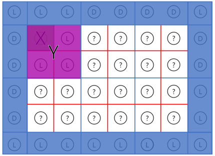


B. 루미너스와 모험 중 마주친 퍼즐게임

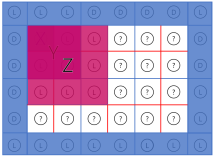
- 먼저, X 는 내부 게임판에서 $A-$ 구역 에 해당하므로 X 에 속한 함정은 L 이어야 합니다.



- 이번에는 전체 게임판의 관점에서 보면, Y 는 전체 게임판에서 $C-$ 구역 에 해당하므로 Y 에 속한 함정은 모두 속성이 같아야 합니다.
- 그런데 이미 X 에 속한 함정이 L 이므로, Y 에 속한 함정은 전부 L 이어야 합니다.



- 다시 내부 게임판으로 돌아오면, 내부 게임판의 $B-$구역과 $C-$구역에 속한 함정은 각각 서로 속성이 같아야 하기 때문에 Z에 속한 함정은 전부 L이 되고...



- 이렇게 내부 게임판과 전체 게임판의 구역이 한 칸씩 어긋나 있기 때문에, 결국에는 내부에 속한 모든 함정이 L이어야 한다는 것을 알 수 있습니다.
- 다시 말해, $C-$구역에 D가 하나라도 있다면 Part 3-1∼5의 방법 혹은 Part 3-6-3의 방법 중 적어도 하나를 통해 문제를 해결할 수 있습니다.

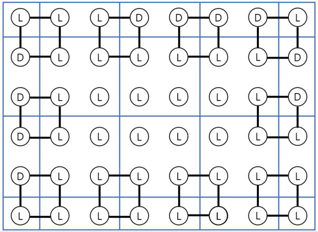
Part 4. 불가능 조건의 증명

- 설명에 앞서 Part 3-1~6으로 문제를 해결할 수 없는 조건을 다시 한번 정리해봅시다.
  1. $A-$ 구역에는 L만 포함되어야 합니다.
  2. $B-$ 구역에는 동일한 속성의 함정이 포함되어야 합니다.
  3. $C-$ 구역에는 L만 포함되어야 합니다.
- 결론부터 말하자면, 이 조건들을 모두 만족하면 Run입니다.

- Part 0에서 봤던 그래프를 다시 꺼내옵시다.
- 올바른 제거 순서를 나타내는 그래프는 사이클이 없어야 하고, indegree의 기우성이 함정의 속성과 일치해야 합니다.
- 앞으로는 indegree가 홀수인 정점을 $L-$ 정점, indegree가 짝수인 정점을 $D-$ 정점 이라고 하겠습니다.
- 저희가 증명할 것은 다음 네 가지 조건을 모두 만족하는 그래프가 없다는 것입니다.

1. $A-$ 구역 에는 $L-$ 정점 만 포함됩니다.
2. $B-$ 구역 에는 동일한 속성의 정점이 포함됩니다.
3. $C-$ 구역 에는 $L-$ 정점 만 포함됩니다.
4. 그래프에 사이클이 없습니다.

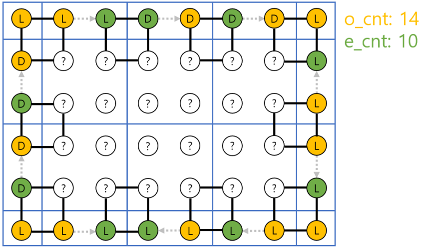
- 전체 그래프에서 $2-$ 모서리에 속하는 정점들만 사용하여 induced subgraph를 만듭니다.
- 그 후 같은 구역에 속한 정점끼리 잇는 간선을 추가로 삭제하여 얻는 그래프를 '기본 그래프'라고 부릅시다.



- 기본 그래프에서 indegree가 홀수인 정점을 $o-$ 정점, 기본 그래프에서 indegree가 짝수인 정점을 $e-$ 정점이라고 합시다.

- 이제 기본 그래프에서 $1-$ 모서리에 속한 정점 중 $o-$ 정점의 개수를 o_{cnt} 라고 하고, $1-$ 모서리에 속한 정점 중 $e-$ 정점의 개수를 e_{cnt} 라고 하겠습니다.
- 저희가 보일 것은 두 가지입니다.
  1. 위에서 언급한 네 가지 조건 중 1번과 2번 조건을 만족하는 그래프는 $e_{cnt} < o_{cnt}$ 를 만족합니다.
  2. 위에서 언급한 네 가지 조건 중 3번과 4번 조건을 만족하는 그래프는 $o_{cnt} \leq e_{cnt}$ 를 만족합니다.
- 위 두 명제를 증명하면 네 가지 조건을 모두 만족하는 그래프가 없음을 증명할 수 있습니다.

- 먼저 1번과 2번 조건을 만족하는 그래프를 생각합시다.

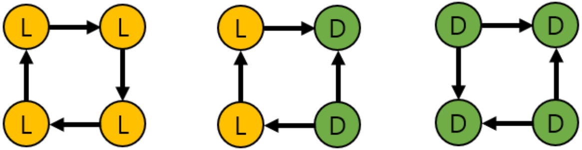


- 앞으로 그림에서 노란색 정점은 $o-$ 정점 을 의미하고, 초록색 정점은 $e-$ 정점 을 의미합니다.
- $A-$ 구역 에 속한 정점에 연결된 간선은 모두 기본 그래프에도 속합니다. 1번 조건에서 $A-$ 구역 에 속한 정점은 $L-$ 정점 이므로 동시에 $o-$ 정점 입니다.

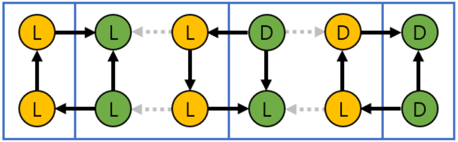
- $B-$ 구역에 속한 정점 사이에 연결된 간선은 기본 그래프에는 속하지 않습니다.
- 따라서 $B-$ 구역에 속한 두 정점 중 하나는 전체 그래프에서 indegree와 기본 그래프에서 indegree의 기우성이 같을 것이고, 다른 하나는 기우성이 다를 것입니다.
- 2번 조건에서 같은 $B-$ 구역에 속한 정점은 둘 다 $L-$ 정점이거나 둘 다 $D-$ 정점이므로, 반대로 기본 그래프의 관점에서 보면 같은 $B-$ 구역에 속한 정점은 하나는 $o-$ 정점이고 하나는 $e-$ 정점이 됩니다.
- 정리하면, $B-$ 구역에 속한 $o-$ 정점과 $e-$ 정점의 개수는 같고, $A-$ 구역에는 $o-$ 정점만 속하기 때문에 $e_{cnt} < o_{cnt}$ 입니다.

- 다음으로 3번과 4번 조건을 만족하는 그래프를 생각합시다.
- 여기서는 $\min(N, M)$에 대한 수학적 귀납법을 사용하여 $2 \times N$ 게임판에서 claim을 증명한 후 $N \times M$ 게임판에서의 claim이 $(N-2) \times (M-2)$ 게임판에서의 claim으로 환원됨을 보이겠습니다.

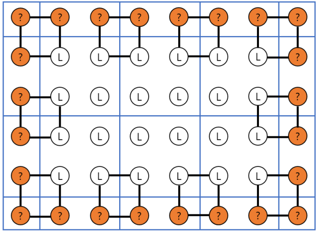
- 앞으로의 논의에서 기본이 될 2×2 게임판에 대해 생각해 봅시다.
- 모든 정점의 indegree의 합은 4이므로, $o-$정점의 개수는 0개, 2개, 4개 중 하나입니다.
- 그런데 o-정점이 4개 있을 때는 왼쪽 그림과 같이 항상 사이클이 만들어지게 됩니다.
- 따라서 4번 조건을 만족하기 위해서는 가운데 그림이나 오른쪽 그림과 같이 $o-$정점의 개수가 0개 또는 2개가 되어야 합니다.
- 즉, 2×2 게임판에서 $o_{cnt} \leq e_{cnt}$ 입니다.



- 다음으로 $2 \times N$ 게임판에 대해 생각해 봅시다.
- $2 \times N$ 게임판의 기본 그래프는 다음 그림과 같이 2×2 그래프 여러 개로 이루어져 있기 때문에 $o_{cnt} \leq e_{cnt}$가 성립합니다.

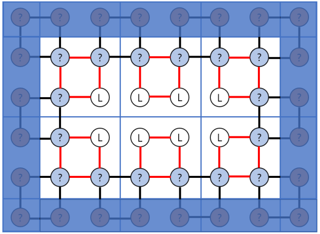


- 이제 $N \times M(N \geq 4, M \geq 4)$ 게임판에 대해 생각해 봅시다.
- $N \times M$ 게임판의 기본 그래프 또한 2×2 그래프 여러 개로 이루어져 있기 때문에 $e-$ 정점 의 수는 $o-$ 정점 의 수보다 많거나 같습니다.



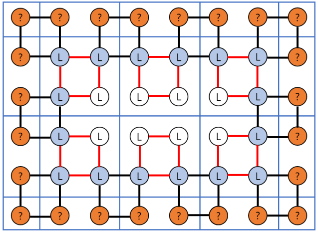
- 그러나 저희는 $1-$ 모서리 에 속한 정점의 속성에 관심이 있으므로 내부의 $1-$ 모서리 에 속한 정점은 제외하고 생각해야 합니다.

- 전체 그래프에서 내부 정점만 이용하여 만든 induced subgraph를 '내부 그래프'라고 합시다.
- 내부의 내부 정점 (i.e. $2-$ 모서리의 여집합) 은 여전히 $L-$ 정점 입니다.
- 그러나 내부의 $1-$ 모서리에 속하는 정점은 $1-$ 모서리 쪽으로 가는 간선 (그림에서 검은색 선) 이 내부 그래프에 속하지 않기 때문에 indegree의 기우성를 알 수 없습니다.



- 하지만 3번 조건은 내부 정점만 고려하므로 내부 그래프가 3번 조건을 만족함은 보장됩니다. 물론 내부 그래프는 4번 조건도 당연히 만족합니다.
- 따라서 수학적 귀납법 가정에 의해 내부 그래프에서 $o_{cnt} \leq e_{cnt}$가 성립합니다.
- 이는 즉, 내부의 $1-$ 모서리에 속한 정점 중 내부 그래프의 기본 그래프 (위 그림에서 빨간색 선)에 속한 간선에 의한 indegree가 짝수인 정점의 개수가 홀수인 정점의 개수보다 많거나 같다는 뜻입니다.

- 그런데 다음 그림에서 볼 수 있듯이, 내부의 $1-$ 모서리에 속한 정점에 연결된 간선은 전체 그래프의 기본 그래프에 속해 있거나, 내부 그래프의 기본 그래프에 속해 있거나, 둘 중 하나입니다.



- 3번 조건에 의해 내부 정점은 모두 $L-$ 정점 이므로 전체 그래프에서 indegree가 홀수입니다.
- 따라서 전체 그래프에서 $o-$ 정점 인 정점은 내부 그래프에서 $e-$ 정점 이 되고, 전체 그래프에서 $e-$ 정점 인 정점은 내부 그래프에서 $o-$ 정점 이 됩니다.

- 내부 그래프에서 $o_{cnt} \leq e_{cnt}$ 임은 아까 확인했습니다.
- 따라서 전체 그래프에서 내부의 $1-$ 모서리 에 속하는 정점을 생각하면 반대로 $o-$ 정점 의 개수가 $e-$ 정점 의 개수보다 많거나 같을 것입니다.
- $2-$ 모서리 에 속한 정점 모두를 생각하면 $e-$ 정점 의 개수가 $o-$ 정점 의 개수보다 많거나 같음도 아까 확인했습니다.
- 따라서 $1-$ 모서리 에 속한 정점만 생각하면 $e-$ 정점 의 개수가 $o-$ 정점 의 개수보다 많거나 같습니다. 즉 $o_{cnt} \leq e_{cnt}$ 입니다. ■

- Part 4의 내용을 정리해봅시다. 만약 어떠한 N과 M이 짝수인 $N \times M$ 게임판이 아래의 3가지 속성을 가진다고 가정합시다.
  1. 게임판의 $A-$구역에는 L만 포함되어 있습니다.
  2. 게임판의 각 $B-$구역은 동일한 속성의 함정으로 구성되어 있습니다.
  3. 게임판의 $C-$구역에는 L만 포함되어 있습니다.
- 만약 이러한 게임판을 해결 할 수 있는 방법이 있다면, 즉 `Solve`라면, 이 게임판을 그래프로 치환했을 때 4가지 조건을 만족해야 합니다.

- 그러한 4가지 조건은 아래와 같습니다.
  1. $A-$ 구역에는 $L-$ 정점만 포함됩니다.
  2. $B-$ 구역에는 동일한 속성의 정점이 포함됩니다.
  3. $C-$ 구역에는 $L-$ 정점만 포함됩니다.
  4. 그래프에 사이클이 없습니다.
- 하지만 그러한 그래프는 $e_{cnt} < o_{cnt}$ 이면서 $o_{cnt} \leq e_{cnt}$ 이어야 하기 때문에 모순이 발생해 존재할 수 없습니다.
- 이는 가정의 거짓을 의미하고 전 슬라이드의 3가지 속성을 가지고 있는 게임판은 해결할 수 없음을 의미합니다.
- 따라서 그러한 게임판에 대해서는 Run을 출력해야 함이 증명이 되었습니다.

C. 조화로운 마법 농구 게임

`prefix_sum` , `two_pointer`
출제진 의도 – **Medium**

- 제출 25번, 정답 4명 (정답률 28.00%)
- 처음 푼 사람 : **kimjihoon**, 196분
- 출제자 : `stonejjun`

Part 1

- 문제를 정확히 이해하고 리나와 루나의 전략을 정확히 생각해야 합니다.
- 루나는 $1 \ldots a-1$ 라운드에 축복으로 얻을 수 있는 점수를 최대화하기 위해 노력합니다.
- 또한 루나는 $b \ldots K$ 라운드에는 끝까지 축복으로 얻을 수 있는 점수를 최대화하여 최종 점수 X 점을 달성하거나, 반대로 축복을 전혀 쓰지 않고 최종 점수 Y 점을 달성합니다.
- 이때 최종점수는 $\max(|X|, |Y|)$ 점이며, 리나는 이 값이 최소가 되도록 a, b를 잘 고릅니다.

Part 2. 각 라운드에서 얻을 수 있는 최대 점수 구하기

- Part 1에서 각 라운드에서 루나가 축복을 잘 써서 얻을 수 있는 최대 점수와 축복을 안 쓸 때의 점수를 구해야 한다는 사실을 짚었습니다.
- 축복을 안 쓸 때의 점수는 L_j 부터 R_j 까지의 구간에 S_j 의 값을 더해주는 것을 반복하여 각 위치에 대한 값을 구하면 됩니다.
- 이 과정은 세그먼트 트리나 Lazy propagation을 이용해도 되지만, 시간 제한에 편하게 들어오기 위해서는 prefix sum을 이용해야 합니다.

- 핵심 관찰은 골대가 늘어났다가 줄어드는 과정이 특별하여, 한 점은 연속으로 많아야 3번 골대에 포함된다는 점입니다.
- 따라서 보통은 계속 축복을 계속 쓸 수 있으나, $2k, 2k+1, 2k+2$번 상태가 모두 포함된 경우 $S_{2k} + S_{2k+2} < S_{2k+1}$인 경우 $2k+1$번 상태만 선택해야 합니다.
- 따라서, 모든 짝수 j에 대해서 축복을 쓴다고 가정하고 $S_{2k+1} - S_{2k} - S_{2k+2} > 0$인 경우에 이 세 구간이 모두 포함되는 구간 $[l, r]$에 이 값을 더하여 각 라운드별 최대 점수를 구할 수 있습니다.
- 이 또한 prefix sum을 이용하여 선형 시간에 해결할 수 있습니다.

Part 3. 리나의 전략 세우기

- Part 1에서 정의한 X, Y 에 대해, $P(a,b) = \max(|X|, |Y|)$ 라 정의합시다.
- 고정된 a 에 대해, $P(a,b)$ 는 b 가 증가함에 따라 아래로 볼록한 그래프를 갖습니다. 따라서 삼분탐색을 이용하여 문제를 해결할 수도 있습니다.
- 하지만 추가적인 관찰로, a 가 증가함에 따라 $P(a,b)$ 를 최소화하는 b 가 **단조증가합니다**. 이 사실을 이용하면 $\mathcal{O}(K)$ 시간에 이 부분을 해결할 수 있습니다.

Epilogue. 시간 복잡도

- 의도한 풀이는 prefix sum과 two pointers를 이용한 $\mathcal{O}(N+K)$의 선형 시간 풀이입니다.
- 하지만, Part 3에서 삼분탐색을 활용하거나 Part 2에서 Fenwick tree를 사용하는 등 빠른 $\mathcal{O}(K\log K)$ 혹은 $\mathcal{O}(N\log N)$ 풀이는 시간 내에 통과할 수도 있습니다.
- 다만 Part 2에서 Lazy propagation을 사용하거나, 의도한 관찰 없이 DP를 세그먼트 트리로 이용하는 트릭[1]을 쓰는 느린 $\mathcal{O}(N\log N)$ 풀이는 통과하지 못하도록 제한을 설정하였습니다.

D. 수식 알아맞히기 퀴즈 게임

FFT, Number Theory

출제진 의도 – **Hard**

- 제출 8번, 정답 4명 (정답률 62.5%)
- 처음 푼 사람 : **sebinkim**, 719분
- 출제자 : ahgus89

- 소수 p가 주어집니다.
- 모든 연산은 mod p에서 이루어집니다. 즉, $x = y$는 암묵적으로 $x \equiv y \pmod p$를 의미합니다.
- $x = 1, \cdots, p-1$에 대해 $f(x) = \sum_{i=1}^{p-1} a_i \cdot i^x$이 입력으로 주어집니다.
- 이 때, 모든 a_i를 빠른 시간 안에 구해야 합니다.

D. 수식 알아맞히기 퀴즈 게임

- $x \neq 0$일 때 다음 식이 성립하는 데 주목합시다.

$$\sum_{i=1}^{p-1} x^i = \begin{cases} -1 & x = 1 \\ 0 & \text{otherwise} \end{cases}$$

- 이를 바탕으로 다음과 같은 결론을 얻을 수 있습니다.

$$\sum_{x=1}^{p-1} f(x) j^{-x} = \sum_{x=1}^{p-1} \sum_{i=1}^{p-1} a_i i^x j^{-x} = \sum_{i=1}^{p-1} a_i \sum_{x=1}^{p-1} (ij^{-1})^x$$

- $ij^{-1} = 1$, 곧 $i = j$ 인 경우에만 $\sum_x (ij^{-1})^x \neq 0$ 이므로 위 식은 놀랍게도 $-a_j$ 가 됩니다.

- 따라서, 모든 $j = 1, \cdots, p-1$에 대해 $\sum_{x=1}^{p-1} f(x) j^x$를 빠르게 계산할 수 있으면 됩니다.
- 이는 j에 대한 $p-1$차 다항식으로, 결국 mod p 계수 다항식에 $1, \ldots, p-1$을 대입한 결과를 빠르게 알 수 있으면 됩니다.
- 이산 푸리에 변환 (DFT) 를 이용하면, N차 다항식 $g(x)$와 $\omega^{2^k} \equiv 1 \pmod p$을 만족하는 ω에 대해 $g(1), g(\omega), g(\omega^2), \cdots g(\omega^{2^k-1})$을 $\mathcal{O}(Nk)$ 시간에 계산할 수 있습니다.

- 이러한 ω를 찾는 방법은 $p-1=2^k\cdot d$인 홀수 d와 $\bmod p$에 대한 원시근 γ에 대해 $\omega=\gamma^d$를 쓰는 방법입니다.
- k가 충분히 크고 d가 충분히 작다면, $g(x)$를 $g(x)=g_0(x^d)+xg_1(x^d)+\cdots x^{d-1}g_{d-1}(x^d)$와 같이 쪼개어 쓸 수 있으므로 $O(dNk)$ 시간에 문제를 해결할 수 있습니다.
- 하지만 당장 $k=1$인 소수만 해도 너무 많기 때문에, 다른 방법을 생각해야 합니다.

- 편의상 $g(x) = \sum_{i=1}^{p-1} g_i x^{-i}$ 라고 두고, 원시근을 γ 라 합시다. $g(\gamma^k) = \sum_{i=1}^{p-1} g_i \gamma^{-ik}$ 입니다.
- ik 를 $i, k, i+k$ 에 관련된 항으로 분리해 봅시다.
  - $\sigma_n = n(n+1)/2$ 로 두면, $-ik = \sigma_i + \sigma_k - \sigma_{i+k}$ 입니다.
- 따라서, $g(\gamma^k) = \gamma^{\sigma_k} \cdot \sum_{i=1}^{p-1} g_i \gamma^{\sigma_i} \gamma^{-\sigma_{i+k}}$ 입니다. $a_s = g_s \gamma^{\sigma_s}, b_s = \gamma^{\sigma_{2p-s}}$ 으로 두면, $g(\gamma^k)$ 는 수열 a 와 b 의 convolution 으로 구할 수 있습니다.
- 따라서 이를 FFT로 구해주면 $O(p \log p)$ 시간에 문제를 해결할 수 있습니다.

- 소개한 풀이는 Bluestein's algorithm으로 불리는 FFT의 일반화된 형태 중 하나입니다.
- Convolution에 사용하는 FFT는 실수를 사용하므로, 오차에 주의해야 합니다.
- 일반적으로 n차 다항식에서 q개의 점을 대입한 결과를 알아내는 문제 (multi-point evaluation)은 $\mathcal{O}\left((n+q)\log^2(n+q)\right)$ 시간에 수행할 수 있습니다. 다만 이 풀이에 기반한 $\mathcal{O}\left(p\log^2 p\right)$ 코드는 통과하지 못하도록 제한을 조정하였습니다.

E. 초콜릿 쪼개기 게임

`game_theory`, `precomputation`
출제진 의도 – **Medium**

- 제출 103번, 정답 24명 (정답률 24.27%)
- 처음 푼 사람 : **Green55**, 59분
- 출제자 : `stonejjun`

- 이 문제의 풀이를 이해하기 위해서는 스프라그-그런디 수에 대해서 알고 계셔야 합니다.
- 풀이 설명은 스프라그-그런디 수에 대한 이해를 하고 있다는 가정하에 진행합니다.

- $g[i][j]$ 를 크기 $i \times j$ 짜리 초콜릿으로 게임할 때의 스프라그-그런디 수로 정의하면 작은 값부터 순서대로 구하는 것으로 DP 테이블을 채우듯이 값을 채워넣을 수 있습니다.
- $g[n][m]$ 을 계산하여 값이 0 이라면 후공의 승리, 그렇지 않다면 선공의 승리입니다.
- 하지만 테이블을 채우는 데 $\mathcal{O}\left((n+m)^3\right)$ 시간이 필요하므로 시간초과가 나게 됩니다.
- 따라서 추가적인 관찰이 필요합니다.

- $g[i][j]$ 값을 출력해보면 34를 주기로 그런디 수의 패턴이 반복되는 것을 알 수 있습니다. 이를 토대로 앞의 일부 수에 대해서만 예외처리를 해주고 그 이상의 수에 대해서는 mod 34로 처리를 해주면 문제를 sublinear time에 해결할 수 있습니다.

- 또는 규칙을 찾지 않고 그런디 수에 대한 추가적인 관찰을 통해서 문제를 해결할 수도 있습니다. $f[n] := g[1][n]$ 이라 할 때,

$$g[n][m] = \begin{cases} 1 & f[n] \neq 0,\ f[m] \neq 0 \\ f[n] + f[m] & \text{otherwise} \end{cases}$$

- 위 사실을 이용해 $f[n]$의 값을 모두 구하고 DB에 저장하여 해결할 수 있습니다.

- 관찰에 대한 증명 – by Aeren
  - https://pastebin.com/7f2nFcpM

F. 흑왕과 어둠의 게임 대진표

ad_hoc
출제진 의도 – **Hard**

- 제출 18번, 정답 2명 (정답률 11.11%)
- 처음 푼 사람 : **ainta**, 986분
- 출제자 : `stonejjun`

Part 1. K번 플레이어가 우승할 조건.

- 일반성을 잃지 않고 $M = 2^N, K = 1$, 그리고 1번 플레이어는 가위라고 가정합니다.
- 1번 플레이어가 우승하기 위해서는 바위와 만나지만 않으면 됩니다. 즉, 모든 바위를 보자기로 없애버릴 수 있으면 우승시킬 수 있습니다
- 또한 바위와 바위가 만나도 바위 한 개가 탈락하기 때문에 바위의 수를 줄일 수 있습니다.
- 따라서 바위를 한쪽에 최대한 모는 것이 이득입니다.

- 바위가 1개 이상이면 보자기도 1개 이상 필요합니다.
- $2^{N-1}-1$ 개의 바위까지는 한쪽에 몰아서 보자기 1개로 커버가 가능합니다.
- 바위가 2^{N-1} 개면 보자기가 2개 이상 필요합니다.
- $2^{N-1}+2^{N-2}-2$ 개의 바위까지는 보자기 2개로 커버가 가능합니다.
- 이를 일반적으로 확장하여 j개의 보자기로는 $\sum_{i=1}^{j}(2^{N-i}-1)$ 개의 바위까지 커버할 수 있으며,

  이 사실을 통해 각 고유 모양의 갯수를 세는 것으로 대진표 완성 여부를 판단할 수 있습니다.

Part 2. 각 사람의 고유 모양 알아내기.

- a의 고유 모양을 알고 있을 때 b의 고유 모양을 알기 위해서는 a와 b를 두 번 대결시켜야 합니다. 정확히는 a vs b, b vs a의 결과로 a와 b의 상성 관계를 알 수 있습니다.
- 이때 질문 '? 1 a b c', '? c b a 1'을 한다고 가정합시다.
- 우리는 두 질문의 4강전 결과에서 1과 a, b와 c의 상성을 각각 알 수 있습니다.

- 결승전과 3-4위전을 합쳐서 최종전이라고 명명하겠습니다.
  - 두 질문에서 1의 최종전 상대 x가 같다면, 1과 x의 상성을 알 수 있습니다.
  - 다르다면 1과 a, 혹은 b와 c의 고유 모양이 동일합니다.
- 최종전과 4강전의 대칭성을 생각하면, 항상 1과 최종전을 붙은 적이 동일한 것처럼 정보를 얻을 수 있습니다. 이렇게 얻은 정보들을 종합하면 2번의 질문으로 3명의 플레이어의 모양을 알 수 있습니다.

- $n \le 2^{16}$ 이고, 따라서 필요한 질문의 횟수는 $2^{16} \cdot \frac{2}{3} \approx 43700$ 으로 문제를 해결하기에 충분합니다.
- Part 2에서 모든 플레이어의 모양을 구하고, Part 1에서 세운 승리 기준을 판단하여 문제를 해결할 수 있습니다.

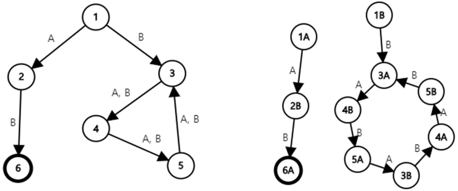
G. 다리 건너기 게임

`game_theory`
출제진 의도 – **Medium**

- 제출 22번, 정답 7명 (정답률 31.82%)
- 처음 푼 사람 : **namnamseo**, 79분
- 출제자 : `jhwest2`

- 단순 방향 그래프 G가 주어지고, 각 방향 간선 (u, v)는 A만 이용할 수 있거나, B만 이용할 수 있습니다.
- s번 정점에 위치한 하나의 말을 A와 B가 간선을 따라 번갈아가면서 옮기는데, N번 정점에 도달하는 이동을 한 사람이 승리합니다.
- 서로가 최선을 다해 게임을 할 때 각 시작 정점마다 필승 전략이 존재하는 사람 또는 무승부 여부를 구하는 것이 목표입니다.

G. 다리 건너기 게임



- 그래프를 왼쪽 그림에서 오른쪽 그림과 같이 변형해 줍시다.
  - 각 정점을 A번과 B번 두 개로 분리하여, A 간선은 A 정점에서 B 정점으로 가도록 잇고, B 간선은 B 정점에서 A 정점으로 가도록 이어줍니다.
  - 그림에는 없지만 움직이지 않는 행동에 대한 간선도 추가해야 한다는 점에 유의합니다. 같은 번호에서 나온 A 정점과 B 정점을 서로 이어주면 됩니다.

- 이제 간선의 종류를 신경 쓸 필요가 없습니다! A와 B는 새로 생긴 그래프에서 매 턴 한번씩 **반드시** 간선을 따라서 말을 이동시키고, N에 해당하는 두 정점 중 한 곳에 먼저 위치시키는 사람이 승리합니다.

- 각 정점은 선공이 이기는 시작점 (W), 후공이 이기는 시작점 (L), 무승부가 발생하는 시작점 (D) 중의 한 가지입니다.
- N 번 정점에서 시작해 역방향 그래프를 따라 탐색하면서, 모든 정점을 저 세 가지 종류 중 하나로 분류하는 것이 목표입니다.

- N에 해당하는 정점은 W 정점입니다.
- 어떤 정점이 W이기 위해서는 그 정점에서 인접한 L 정점이 존재해야 합니다.
- 어떤 정점이 L이기 위해서는 그 정점에서 인접한 모든 정점이 W 정점이여야 합니다.
- N으로 도달할 수 없거나, 위 두 조건을 만족하지 않는 정점은 D 정점입니다.

- 위 사실들을 이용하여 모든 정점의 종류를 구하는 과정을 Queue를 이용하여 $\mathcal{O}(N+M)$에 작동하도록 구현할 수 있습니다.

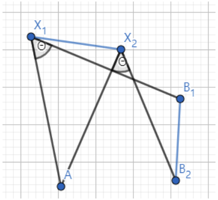
H. 기둥과 성벽 디펜스 게임

`math, geometry`
출제진 의도 – **Challenging**

- 제출 2번, 정답 2명 (정답률 100.00%)
- 처음 푼 사람 : **andyjung**, 121분
- 출제자 : `stonejjun03`

Part 0.

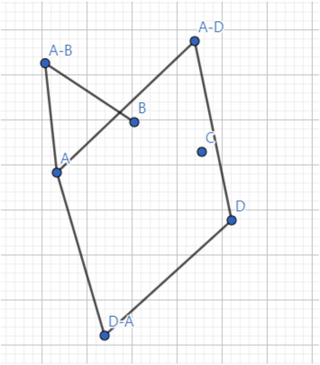
- 아래의 그림처럼 점 X_1, X_2가 각각 점 (A, B_1)과 점 (A, B_2)로부터 확장권을 사용해 얻어진 점일 때 $\frac{AB_1}{AX_1} = \frac{AB_2}{AX_2}, \frac{AX_2}{AX_1} = \frac{AB_2}{AB_1}$ 로부터 $\triangle AX_1X_2 \sim \triangle AB_1B_2$ (SAS 닮음).
- 따라서 $\frac{X_1X_2}{B_1B_2} = \frac{AX_1}{AB_1} = \frac{1}{2\sin\frac{\theta}{2}}$.



Part 1.

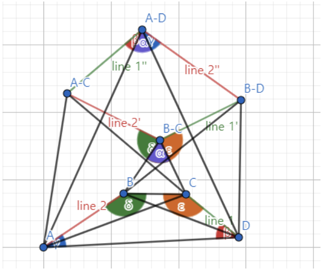
- 확장권을 쓴 후의 성벽 (볼록 껍질) 을 이루는 기둥 (점) 은 확장권을 쓰기 이전에 성벽을 이루던 두 기둥으로 부터 만들어진 기둥임을 관찰해야 합니다.
- 다시 말해 매 상태마다 성벽을 이루지 않는 점들은 아예 없애버린 후 생각을 해도 결과가 변하지 않습니다.

- 아래 그림과 같이 점 A와 B에서 확장권을 사용하여 만든 새로운 점을 $A-B$라고 정의합시다.

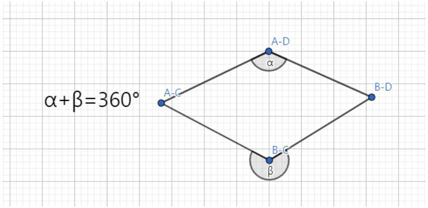


Part 2. 확장권 사용 이후 성벽을 이루는 기둥

- 확장권 사용 이전, 성벽에서 점 A, B, C, D가 순서대로 인접해 있었다고 가정합시다. 확장권을 쓴 후 $A - C, B - D$가 인접할 수 있는지 알아봅시다.



- 전 슬라이드의 그림에서 같은 색의 각을 끼고 있는 모든 두 삼각형은 Part 0의 관찰에 의해 닮음입니다.
  - $\triangle ABC \sim \triangle(A-C)(B-C)C, \triangle BCD \sim \triangle B(B-C)(B-D)$
  - $\triangle ACD \sim \triangle A(A-C)(A-D), \triangle DBA \sim \triangle D(B-D)(A-D)$
- 이때 파랑, 주황, 초록, 빨강의 네 각이 사각형을 이루므로 각도의 합이 360° 이며, 이는 그림 위쪽의 각 $(A-C)(A-D)(B-D)$ (= 빨강 + 보라 + 파랑) 와 중앙의 큰 각 $(A-C)(B-C)(B-D)$ (= 초록 + 주황 − 보라) 의 합이 360° 가 됨을 의미합니다.



- 위의 그림과 식을 만족할 때, 점 $A-C$ 와 점 $B-D$ 를 잇는 직선 위, 혹은 그 직선으로 구분된 위쪽 반평면 위에 적어도 하나의 점이 존재하게 됩니다.
- 따라서 확장권을 쓴 이후 점 $A-C$ 와 점 $B-D$ 가 모두 성벽에 있다면, 반드시 $A-D$ 혹은 $B-C$ 가 성벽 위에 존재하게 됩니다.

- 이와 비슷한 방식으로 점 $A-B, C-D$ 사이에는 $B-C$가, $A-B$와 $B-C$ 사이에는 $A-C$ 혹은 B가 존재해야 함을 보일 수 있습니다.
- 이를 종합하면, 확장 이후 기둥 $i-j$에 시계방향으로 인접한 점은 $i-(j+1)$ 혹은 $(i+1)-j$만 가능하다는 것을 알 수 있습니다.
  - $i+1, j+1$은 각각 i와 j에 시계방향으로 인접한 점입니다.

- 새로운 성벽을 이루는 기둥은 기존 성벽을 two pointer처럼 한바퀴 돌리면서 생성하게 되며, 이 과정에서 기존 성벽의 각 부분이 회전하고 $\frac{1}{2\sin\frac{\theta}{2}}$ 배 확장하여 새로운 성벽의 각 부분을 2번씩 이루게 됩니다.
- 따라서 확장권을 쓸 때마다 성벽의 길이는 $\frac{1}{\sin\frac{\theta}{2}}$ 배가 됩니다.
- 확장권 사용 순서는 상관이 없으며, 처음 주어진 점의 볼록 껍질을 구해 성벽의 길이를 구한 후 확장권의 수 만큼 각 값을 곱해나가는 것으로 문제를 해결할 수 있습니다.

- 문제 해결 과정에서 세 점이 한 직선 위에 있을 수 있다는 점 빼고는 최대한 예외처리를 하지 않아도 되도록 문제를 만들었습니다.
- 실수오차가 두렵다면 python을 사용해도 되지만, C++의 실수 자료형을 사용해도 실수오차에 대한 걱정을 하지 않도록 최대한 문제의 조건들을 조정했습니다.

I. 고장난 계산기 (Calculator) 게임

`segment_tree, case_work`
출제진 의도 – **Challenging**

- 제출 41번, 정답 2명 (정답률 4.88%)
- 처음 푼 사람 : **kimjihoon**, 1086분
- 출제자 : blackking26

- 덧셈과 곱셈이 포함된 길이 N 의 수식에서 구간 업데이트 쿼리를 처리하면서 수식의 값을 계산해야 합니다.
- 쿼리는 수식을 구성하는 숫자에 일정한 값을 더하는 것입니다.
- 덧셈을 10, 곱셈을 11의 숫자로 생각하여 각 숫자는 12의 주기로 순환하게 됩니다.

- 숫자가 12의 주기로 순환하므로 구간 전체에 쿼리를 적용해서 얻을 수 있는 서로 다른 수식의 값은 최대 12개입니다.
- 세그먼트 트리의 각 노드마다 현재 계산기의 식 전체에 i를 더했을 때 수식의 상태를 나타내는 12개의 값을 관리하면 lazy propagation으로 쿼리를 처리할 수 있습니다. $(0 \leq i < 12)$
- 문제는 수식의 상태를 어떻게 간단하게 표현할 것인지, 그리고 두 노드를 어떻게 합칠지입니다.

✓ 수식의 상태를 다음 네 가지로 분류합니다.

1. 숫자가 없을 때: (op)
2. 숫자가 있고 연산기호가 없을 때: $(op)a(op)$
3. 숫자와 곱셈이 있고 덧셈이 없을 때: $(op)a * b * c(op)$
   a, c는 숫자로만 이루어집니다.
4. 숫자와 덧셈이 있을 때: $(op)a * b + c + d * e(op)$
   a, e는 숫자로만 이루어지고, b, d는 숫자와 곱셈으로만 이루어집니다.

(op) 자리에는 연산 기호가 오거나 아무것도 오지 않을 수 있습니다.

- 이제 (op) 에 해당하는 연산기호 2개와 a, b, c, d, e 에 해당하는 수식을 계산한 값 5개, 여기에 더해서 a 의 길이까지 관리하면 두 노드를 합칠 수 있습니다.
- 양쪽 노드의 상태에 따라, 그리고 두 노드 사이에 오는 연산 기호에 따라 $4 \times 3 \times 4 = 48$ 가지 경우의 수가 있습니다.
- 기본적으로 케이스 워크가 심합니다. 적당히 항등원을 이용하거나 비슷한 경우를 묶어서 코드 길이를 줄일 수 있습니다.

J. 고인물의 두번째 리듬게임

dp, two_pointers
출제진 의도 – **Hard**

- 제출 64번, 정답 5명 (정답률 14.29%)
- 처음 푼 사람 : **kimjihoon**, 59분
- 출제자 : jhwest2

- 노트의 등장 시각 T_i, 점수 A_i, 에너지 B_i가 주어집니다.
- 에너지를 X 만큼 모으면 임의의 시간에 시작해 Y 초동안 점수가 2배가 되도록 할 수 있습니다.
- $Q \leq 10$ 가지의 X, Y 값에 따른 최대 점수를 구해야 합니다.

- 답에서 모든 노트의 점수를 빼면, 피버 타임 내에 처리한 노트의 점수의 합이 됩니다.
- 피버 타임 내에 처리한 노트의 점수의 최대 합을 구하는 문제를 해결하면 됩니다.

- $Dp[i]$ 를 1부터 i 번째 노트까지 고려했을 때 얻을 수 있는 최대 점수로 정의합니다.
- 이전 상태에서 $Dp[i]$ 의 값을 계산할 때 i 번째 노트까지 포함해서 피버 타임을 만드는 경우를 생각해주어야 합니다.

- i 번째 노트가 피버 타임의 마지막 노트가 되도록 할 때, 피버 타임의 첫 노트의 번호로 가능한 가장 작은 것을 L_i 라고 합시다.
- 이런 L_i 와 R_i 들의 값은 투 포인터 또는 이분 탐색을 이용해서 구할 수 있습니다.
- 이때, i 번째 노트가 피버 타임의 마지막 노트가 되도록 하는 첫 노트의 범위는 $[L_i, L_{i+1}]$ 로 주어집니다.

- $[l, r]$ 번째 노트에 피버 타임을 적용시키기 위해서는 $l-1$ 번째 노트부터 앞의 몇 개의 노트는 피버 타임 없이 에너지를 모으는 데에 사용되어야 합니다.
- i 번째 노트에 대해서 왼쪽으로 적어도 몇 번째 노트까지 에너지를 모아야 X 보다 커지는 지를 계산해주고, 이 값을 C_i 라고 합시다.
- 이런 C_i 역시 투 포인터 또는 이분 탐색을 이용해서 구할 수 있습니다.

- 전처리한 정보들을 이용해서 Dp 식을 세워볼 수 있습니다.
- $Dp[i] = \max\left(Dp[i-1], \max_{L_i \le j \le L_{i+1}}\left(Dp[C_{j-1}-1] + \sum_{x=j}^{i} A_x\right)\right)$
- 오른쪽 합을 누적합으로 처리하면 $Dp[i]$ 를 $O(N)$ 에 모두 구할 수 있습니다.

- 시간 복잡도는 $\mathcal{O}(N \log N + QN)$ 또는 $\mathcal{O}(N \log N + QN \log N)$ 입니다. 구현에 따라 후자에 해당하는 풀이는 시간 초과를 받을 수 있습니다.