Codejam 문제로 공부하는 Nimber

Problem

Attempt 1

일단 문제를 읽으면서 떠올린 아이디어는

- 게임이네? 일단 Game tree를 적용해 볼까?
- 박테리아를 놓아서 갈라진 보드는 완전히 독립적인 Subproblem이 되는군

Game tree를 적용하기 위해 이런 함수를 정의해봤다.

임의의 상태 G에 대해 반드시 이길 방법이 존재한다면 $f(G) := 1$, 아니라면 $f(G) := 0$으로 정의하자. 내 move m에 의해 갈라진 두 개의 상태 G_m^1, G_m^2에 대해 대충 이렇지 않을까? $f(G) = max(\forall m\{f(G_m^1)\{XNOR\}f(G_m^2)\})$, 왜냐하면 두 board의 상태값이 동일하면 상대 턴에 어떤 move가 나와도 내가 그것을 상쇄할 수 있는 다른 move를 할 수 있을지 않을까 하는 정도의 추측만 가지고 일단 구현을 해 보았다.

하지만 여지없이 Fail...

Attempt 2

일단 망했기 때문에 Small을 푸는 것만 목적으로 다돌리기 코드를 짜서 넣어보았다. 역시나 기대했던 대로 (Pass, TLE)

Counter example

Attempt 1과 Attempt 2를 비교해서 결과가 다른 데이터를 보면 Attempt 1이 왜 망했는지 알 수 있겠다. 랜덤 데이터를 넣어 돌려봤을 때 걸린 데이터 중 하나는 다음과 같았다.

```
.$..
.$#.
.$..
.$##
```

여기서 Attempt 1은 2열을 채우는 선택을 해도 이길 수 있다고 판단했고 Attempt 2는 그렇지 않다고 판단했다. 2열을 채울 때 $f(col1) = 1$ (왜냐하면 한번에 다 채울 수 있으니까), $f(col3, 4) = 1$(왜냐하면 valid한 move를 하는 경우 무조건 3턴이 걸림)이라서 Attempt 1은 이것도 승리 move라고 판단했던 건데 여기서 $f(col1) = 1$만으로는 정보값이 부족했다.

```
$$..  (1)$$..  (2)$$..  (3)$$..
.$#.     $$#.     .$#.     .$#.
.$..     $$..     .$..     $$..
.$##     $$##     $$##     $$##
```

상대가 이렇게 플레이해서 다시 내 턴으로 돌아오면 f값이 여전히 1이 유지된다. 오른쪽은 무조건 3회 move가 강제된 상태인데, 왼쪽은 내가 어떻게 플레이를 해도 상대가 홀수번 move를 강제할 수 있기 때문이다. (1과 같이 3칸을 닫으면 1회 move니까 fail, 2와 같이 1칸을 닫는 플레이를 하는 경우 상대가 마찬가지로 1칸만 닫으면 3회 move가 강제되니까 fail)

Attempt 3

멘탈이 터져서 f를 땜방칠 궁리를 하기 시작했다. 대충 홀수번 move를 강제할 수 있느냐, 짝수번 move를 강제할 수 있느냐를 function의 결과값에 넣고 조합식을 어떻게 해볼까 하다가 둘 다 강제할 수 없는 경우 어떻게 처리해야 하나에서 멘탈이 수습할 수 없게 터져서 GG를 치고 Analysis를 봤다.

Nimber?

Analysis에서는 이 게임의 State가 Nim 게임의 state로 표현 가능하다고 한다. 게임 자체가 뭔가 남아있는 칸을 지우는 게임이니만큼 그런것 같기도 하다. 그리고 게임의 상태는 nimber라는 non-negative integer로 표현 가능하다고 한다.

그러나 수알못은 위키피디아 설명도 너무 어려워서 버벅이면서 읽어야 했다. 제대로 이해한건지 아직도 잘 모르겠지만. 어쨌든 임의의 상태 G에 대해 Nimber $g(G) := v$란, $\forall 0 \leq t < v, \exists m\ g(G_m) = t$, 즉 v 미만의 모든 상태로 이동 가능하다는 뜻이다. 만약 $g(G) = 0$이라면 최선의 플레이를 해도 반드시 지는 경우가 존재한다는 뜻.

처음에는 이런 식으로 이해해보려고 노력해 봤다. 짝수번 만큼 turn이 진행되는 것이 강제되는 상태를 nimber 0, 홀수번 만큼 turn을 진행하는 것을 강제할 수 있는 상태를 nimber 1, 홀수, 혹은 짝수번 만큼 turn을 진행하는 것을 강제할 수 있는 상태를 nimber 2라는 식으로. 뭐 nimber 3부터는 이런식으로는 설명이 좀 힘들어지긴 하지만, 어쨌든 홀짝 개념에 매몰되어 있던 사람이 빠져나오는 데에는 이런 삽질도 필요했다.

여기서 중요한 것은 두 개의 subproblem으로 쪼개진 상태를 하나의 nimber로 표현하는 방법이었는데, 이게 또 골때리게 $g(G_m) = g(G_m^1)\{XOR\}g(G_m^2)$ 이다. 처음 nimber를 제대로 이해하지 못하고 수식을 읽었을 때에는 왜 여기서 XOR이 나오지? 하고 막막했었는데 나중에 좀 이해가 가고나니, XOR 값 미만의 모든 값으로 이동하는 방법이 존재한다는 뜻이었다.

그러므로 임의의 상태 G에 대해, $g(G) = mex(\forall m\, G_m)$, 즉 어떤 move로도 도달할 수 없는 가장 작은 숫자가 G의 nimber 되시겠다. 여기까지 이해가 되었다면 위키피디아 문서를 읽는 편이 더 쉬울 것 같다.

Solution

즉, 이 문제는 초기상태 G가 주어졌을 때 $g(G_m) = 0$ 이 되는 m의 갯수를 찾아 출력하는 문제다. 구현 자체는 간단한 편이었다.