H - Advance or Eat Editorial by en_translator

Grundy Number

Although it is not directly related to the original problem, we will first review Grundy Number for comparison. Remember, Grundy Number can be used for the following problem:

```
There is a DAG (Directed Acyclic Graph), and some vertices have pieces.  (A ver…
ay have multiple pieces.)
Two players play in turn alternately.  When one's turn, the player chooses a piece a
nd move it along an edge of the DAG.
The player who is unable to make a move will lose.  Which player will win?
```

Copy

How to find Grundy Number

We define a non-negative integer called Grundy Number to each vertex of the DAG. The Grundy Number of a vertex v is $mex(a_1, ..., a_k)$, where $a_1, ..., a_k$ are the Grundy Numbers assigned to the heads of the outgoing edges of vertex v; i.e. the smallest non-negative integer that does not appear in $a_1, ..., a_k$.

How to determine the winner

If the xor of Grundy Numbers of vertices where pieces are placed is 0, then the second player will always win; otherwise the first player will always win.

Proof

As if often the case with game theory problems, it is sufficient to check the following two cases.

- For any winning state, it can be transitioned to a losing state by choosing an appropriate move.
- It is impossible to transition from a losing state to another losing state.

Suppose that we have a state where pieces are placed in vertices with Grundy Numbers $g_1, ..., g_k$. Let $x$be their xor.

- When $x = 0$ (winning state), we can choose g_i which has the most significant bit of x, then move the piece to the vertex whose Grundy Number is $g_i \oplus x$ (which is possible since $g_i \oplus x < g_i$ and by the definition of Grundy Number), resulting in a losing state.
- When $x = 0$ (losing state), by the definition of Grundy Number, no matter which piece is moved to where, the Grundy Number of that piece will change, and so will the xor, so xor become non-0, always resulting in a winning state.

Impartial game

If a problem can be boiled down to the problem above by regarding a state of board as a vertex and a transition as a edge, then it can be treated with Grundy Number. Note however that it relies on the property that possible moves do not depend on whose turn is it (this kind of game is called impartial games). If not, for instance if each player can only move pieces with specific color, then other methods are required.

How to handle with Partisan Games

Consider the following problem as a example of Partisan Game (a game where possible moves depends on players).

```
    There is a DAG (Directed Acyclic Graph), and some vertices have pieces.  (A ver
    ay have multiple pieces.)
    Each edge of the DAG is painted either black or white.
    Two players play in turn alternately.  When one's turn, the player chooses a piece a
    nd move it along an edge of the DAG.
    Here, the first player can only use white edges; the second player can only use blac
    k edges.
5. The player who is unable to make a move will lose.  Which player will win?
```

Unfortunately, this problem cannot be generally solved as easy as for the case of Grundy Number, but we will introduce an efficient way to solve it for special kind of problems.

Intuitively, we treat “a game in which players is reluctant to make their moves.” We define real-numbered evaluation value for each vertex. The more advantageous for the first player the state is, the larger the evaluation value will be; the more advantageous for the second, the smaller it will be. Moreover, the evaluation value will decrease when transitioning along a white edge, and will increase when transitioning a black edge, so that every turn make the player sad.

How to determine the evaluation value?

The evaluation value of each vertex v is determined in the topological order of the DAG as follows.

- If the head of any outgoing edge from v has an undefined evaluation value, then the evaluation value of v itself will also be undefined.
- Otherwise, let $a_1, ..., a_k$ be the evaluation values of heads of white outgoing edges from v, and $b_1, ..., b_l$ be those of black outgoing edges from v, then the evaluation value of v, $eval(v)$, is determined so that $\max\{a_1, ..., a_k\} < eval(v) < \min\{b_1, ..., b_l\}$.
- If there does not exist such real number $eval(v)$, then the evaluation value of v is undefined.
- If there exists such real number $eval(v)$, then the evaluation value is determined (by the following rules uniquely) as:
    - an integer with minimum absolute value if possible; or otherwise
    - a rational number whose denominator is a power of 2 (when expressed as a irreducible fraction with a positive denominator) with minimum denominator.

(Example)

- If $-3.125 < eval(v) < 1.5$, then $eval(v) = 0$
- If $-5.5 < eval(v) < -2$, then $eval(v) = -3$
- If $1.625 < eval(v) < 1.78125$, then $eval(v) = 1.75$

How to determine the winner?

Calculate the sum of evaluations values corresponding to vertices where each piece is. If it is positive, then the first player will always win; if 0 or negative, then the second player will always win.

Proof

This time, it is sufficient to check the following two cases.

- If the sum of evaluation values are positive, then it is possible to choose an appropriate white edge and move a piece along it so that the sum of evaluation values become 0 or positive.
- If the sum of evaluation values are 0 or negative, then no matter which white edge is chosen, after moving a piece along it, the sum of evaluation values will become negative.

Currently we focus on the first player, so we care whether the sum of evaluation values is positive or not, but the next turn will be the second player's, so we then have to care whether the sum of evaluation values if negative or not.

The latter is obvious, so we will show the former. Let s ($s > 0$) be the current sum of evaluation values.

- When s is an integer Since the sum of evaluation values is positive, there is a piece on a vertex with positive evaluation value, so after moving such piece along such white edge that decreases the evaluation value by at most 1 (which is always possible by the definition of the evaluation value), the sum of evaluation values will still be non-negative.
- When s is not an integer Let 2^t be its denominator. Then, there is a piece on a vertex whose evaluation value has a denominator of 2^t or more, so after moving such piece along such white edge that decreases the evaluation value by at most $1/2^t$ (which is always possible by the definition of the evaluation value), the sum of evaluation values will still be non-negative.

Regarding with this problem

Prepare a DAG with 3^N vertices and place N pieces. Each vertex of DAG correspond to a state of a single column. There is a white edge between the states that can be transitioned by advancing a white piece or eating a black piece, and the same applies to black edges. Place a piece to the vertex corresponding to each column in the initial state. Now, the method above can be applied.

It needs to be proved that the evaluation value will never be undefined. We will prove that the evaluation value can be defined by induction in the topological order of DAG. Suppose that we are now focusing on vertex v, and all vertices before it in the topological order has a evaluation value defined. Then, it is sufficient to prove that, if there exist a white edge $v \to a$ and a black edge $v \to b$, then $eval(a) < eval(b)$.

- If both $v \to a$ and $v \to b$ corresponds to "eat", let c be the state transitioned from v by eating both of them, then there exists a black edge $a \to c$ and a white edge $c \to b$, and by the assumption of induction, $eval(a) < eval(c) < eval(b)$.
- Similarly, if two operations do not interfere with each other, then there exists such c described above, so it can be proved similarly.
- The only exceptional case is where $v \to a$ corresponds to eating a black piece and $v \to b$corresponds to advancing that piece (or the corresponding case for white), but in this case there exists a white edge $b \to a$, so $eval(a) < eval(b)$.

Therefore, the problem has been solved by actually finding the evaluation values of all 3^N vertices and take the sum.