12th August 2014

Grundy numbers for competitive programming

Consider a simple game which two players can play. There are N coins in a pile. In each turn, a player can choose to remove one or two coins. The players keep alternating turns and whoever removes the last coin from the table wins.

Once you start playing this game, you will realise that it isn't much fun. You will soon be able to devise a strategy which will let you win for certain values of N. Eventually you will figure out that if both players play smartly, the winner of the game is decided entirely by the initial value of N. How does this work? Suppose that N were 1 or 2. The first player could just remove all coins and win straightaway. If N were 3, whatever the first player did, the second player would end up getting 1 or 2 coins and win in their move. If N were 4 or 5, the first player could remove the required number of coins to make the second player receive 3 coins and lose. Continuing in this fashion, we find that the first player is guaranteed a win unless N is a multiple of 3. The *winning strategy* is to just remove coins to make N a multiple of 3.

Cool and easy, right? Now what if I make a simple modification to the problem? The game now starts with K piles of coins. Initially there are $N_1,N_2…N_K$ coins in the piles. Each player is allowed to remove one or two coins from a single pile in their turn. Once again, the player who makes the last move of removing the last coin from the last existing pile wins the game. Is there a strategy now?

Not so easy anymore right? However, it turns out that there is a strategy in this case as well. However, to find it and to deal with more general games, let us try to identify the salient features of the above games.

- The games are **sequential** [https://en.wikipedia.org/wiki/Sequential_game] . The players take turns one after the other, and there is no passing
- The games are **impartial** [https://en.wikipedia.org/wiki/Impartial_game] . Given a state of the game, the set of available moves does not depend on whether you are player 1 or player 2.
- Both players have **perfect information** about the game. There is no secrecy involved
- The games are **guaranteed to end** in a finite number of moves.
- In the end, the **player unable to make a move loses**. There are no draws. (This is known as a normal game. If on the other hand the last player to move loses, it is called a **misère** [http://en.wikipedia.org/wiki/Mis%C3%A8re] game)

Let's call the games satisfying these properties as **solvable games**. For such games, there is a theory to find the winning strategy. This involves the **Sprague–Grundy theorem** [http://en.wikipedia.org/wiki/Sprague%E2%80%93Grundy_theorem] , which reduces every such game to a game of **Nim** [http://en.wikipedia.org/wiki/Nim] .

So first, let us analyse the game of Nim. In this game, there are K piles of coins containing $N_1,N_2…N_K$ coins. In a turn, a player chooses a pile and removes any number of coins from it. Once again, the player who manages to remove the last coin from the last remaining pile wins.

It turns out that the first player loses Nim iff $xor(N_1,N_2…N_K)=0$. The winning strategy is thus to ensure that the set of piles passed on to the opponent has xor=0. This seems an extremely strange result, so we will derive it here to convince you.

The current state of the game in Nim can be uniquely specified by the ordered K-tuplet of remaining coins in each pile : $(n_1,n_2…n_K)$. Out of all such K-tuplets, some states are winning states - that is, if the current player receives that state, they will win if they play optimally. The remaining states are losing states - whatever the player does, the other player can manage to win. If we can identify the winning and losing states in Nim, we can find a strategy to play the game as well. The best way to do this is to identify a property P such that:

- The eventual losing states (the ones from which no further move can be made) satisfies P. In Nim, there is only one eventual losing state - when all piles become empty.
- From a state that satisfies P, it is impossible to move to another state that satisfies P
- From any state that does not satisfy P, it is possible to move to some state that satisfies P.

Then it is easy to see that a state is a losing state iff it satisfies P.

In Nim, the supposed losing states are exactly those with $xor(n_1,n_2…n_K)=0$. Does this satisfy the three criteria?

- The eventual losing state has $n_1=n_2=…n_K=0$. Hence $xor(n_1,n_2…n_K)=0 \Rightarrow$ The eventual losing state is a P state
- Suppose that $xor(n_1,n_2…n_K)=0$. If we remove some coins from pile 1 to reduce the number of coins to n_1'. The the new result is $xor(n_1',n_2…n_K)=xor(n_1,n_2…n_K) \oplus n_1 \oplus n_1' = n_1 \oplus n_1' \neq 0$. That is, it is not possible to move from a P state to another P state.
- Proving the third criterion is slightly tougher. Let $s=xor(n_1,n_2…n_K)$. Consider the leading bit of s. There should be some n_i with this bit set in it. WALOG, let it be n_1. Let $n_1' = s \oplus n_1$. It is easy to see that $n_1'<n_1$. $xor(n_1',n_2…n_K)=xor(n_1,n_2…n_K) \oplus n_1 \oplus n_1' = s \oplus n_1 \oplus s \oplus n_1 = 0$. Thus reducing n_1 to n_1' allows us to move to a P state.

With Nim thus solved fully, we can move on to the general case of solvable games. Sprague-Grundy theorem reduces the state of every such game to a single Nim pile of a certain size. This pile size is called the **nimber/Grundy number** [http://en.wikipedia.org/wiki/Nimber] of the state. The theorem then essentially states that the state of a game is a winning/losing position iff the corresponding Nim pile is a winning/losing state. The power of the theorem is that it makes a game composed of a collection of independently played solvable games solvable as well. That is, suppose that the game is made up of subgames such that each player can choose to make a move in exactly one of the subgames at each step. A simple example would be K independent piles of Nim. A game composed of K solvable subgames with Grundy numbers G_1, G_2 … G_K is winnable iff the Nim game composed of Nim piles with sizes G_1, G_2 … G_K is winnable.

To apply the theorem on arbitrary solvable games, we need to find the Grundy number associated with each game state. This is how it is done:

- The final losing position corresponds to the empty Nim pile and has Grundy number 0.
- Suppose that we have calculated the Grundy numbers of all states we can move to from the state currently under consideration. Then the Grundy number of the current state is the *smallest whole number which is not the Grundy number of any state that can be reached in the next step* (This is in direct analogy with what happens in Nim. If a Nim pile has n coins, we can remove coins to get 0,1…n-1 coins in the next step. The lowest number not reachable is n, thus a single

Nim pile with n coins is equivalent to current state). Mathematically, if s_1, s_2 … s_k are the game states directly reachable from s,

$$Grundy(s) = min(\{0, 1, \ldots\} - \{Gundy(s_1), Grundy(s_2) \ldots Grundy(s_k)\})$$

The smallest non-negative integer which is not present in a set S is denoted by mex{S}.

- The Grundy numbers for each state can thus be enumerated recursively and the losing states are exactly those with Grundy number equal to 0.

To see how this works, let us apply it in the original game where we have a pile of coins and can remove one or two at a time. 0 coins corresponds to a Grundy number of 0. If we have 1 coin, the only state we can reach is 0 and thus the corresponding Grundy number is 1. With 2 coins, we can reach 0 and 1 - thus the Grundy number is 2. The difference comes once we move to 3 coins. Now we can only reach 1 and 2 coins - with their Grundy numbers 1 and 2 as well - and hence the Grundy number of this state is 0. In general, we can see that the state with n coins has Grundy number n mod 3, and hence the losing states are exactly those cases which have n as a multiple of 3.

The power of the Sprague-Grundy theorem comes when we have multiple subgames to choose from. In the game of Nim, we can choose which pile to play. The winnability is decided by the xor of Nim pile sizes. In the same way, in any game where you have multiple independent subgames to play, the winnability is decided by the xor of the Grundy numbers of the current states of the subgames.

This is the resolution of the modified 1-2 coin removal problem with multiple piles. Suppose that we have K piles initially with coins N_1,N_2…N_K and we are playing 1-2 coin removal game with them.

We just found that the Grundy number for each pile is given by N_i mod 3. Sprague-Grundy theorem tells us that the state of piles is a losing state iff the xor of the Grundy numbers is 0. That is, iff (N_1 mod 3) $\oplus$ (N_2 mod 3) … (N_K mod 3) = 0. Thus (3,4,5) is a winning state whereas (3,4,4) is a losing state.

So that's the theory. Problems about impartial games and Grundy numbers appear often in programming contests. We will now have a look at a few such problems and how to utilise the theory to solve them.

Problem 1 : Matrix Game [http://www.spoj.com/problems/MATGAME/]

The first thing we realise looking at the constraints is that anything like doing a DP on the states of all cells/rows in the matrix is simply not going to work. But then, the game on each row is independent of the game on other rows. Thus we can consider each row as a subgame, and this really looks like a problem where Sprague-Grundy theorem could be useful. So the question is, how do we assign a Grundy number to each row?

The empty row is a losing state since no further move can be made on it. Thus it has Grundy number 0. What if there is only one element in the row? Then reducing it is equivalent to removing coins from a Nim pile, and its Grundy number is just the value of the element. Now suppose there are two elements in the row (a_l,a_r). If a_l=1, the only move we can make is reduce it and get to a_r. Thus the Grundy number of (1,a_r) is 0. Continuing along this, the Grundy number of (a_l,a_r) can be found to be a_l-1. But there is a catch. If a_l=a_r+1, the Grundy number has to be a_l instead of a_l-1 since the Grundy number of (0, a_r), which is reachable from (a_l, a_r), is a_r=a_l-1. Thus the Grundy number for (a_l,a_r) = a_l-1

if $a_l \leq a_r$ and a_l otherwise.

Now, notice that we have used only the Grundy number of a_r in this description and not the value of the element directly. This means that we can keep adding elements to the left following the same procedure. Let G(i) be the Grundy number calculated for the *rightmost* i elements in the row and a_i be the i^{th} element from the right. Then $G(i+1)=a_{i+1}-1$ if $a_{i+1} \leq G(i)$ and a_{i+1} otherwise.

Having calculated the Grundy number for each row in this fashion, the first player loses iff the xor of all the Grundy numbers is 0.

Problem 2 : A String Game [http://www.codechef.com/problems/ASTRGAME]

In this problem, every time we remove a word from the string, the string gets divided into two substrings. Now, the further moves on these substrings are independent and can be thought of as separate subgames - thus the Grundy number of the substring pair is just the xor of the Grundy numbers of the individual substrings.

We can thus calculate the Grundy numbers by dynamic programming. Let f(i,j) be the Grundy number corresponding to the substring S[i…j]. The set of Grundy numbers of all states reachable from the current string is given by $S_{reachable}=\{f(i,k-1) \oplus f(l+1,j)\}$ for all S[k…l] which are substrings of S[i…j] and valid words at the same time. Then $f(i,j)=mex(S_{reachable})$. The base case for the DP is that Grundy number for an empty string is 0.

Tracy wins iff the Grundy number for the entire string is zero.

Practice Problems

- WPLAY [http://www.codechef.com/problems/WPLAY]
- G3 [http://www.codechef.com/SEP09/problems/G3]

***PS**: Thanks to Utkarsh Lath for helping out significantly with this post. Please comment with any queries or suggestions you have regarding this post, and if you would like to discuss any further problems solvable using Grundy numbers.*

Posted 12th August 2014 by Raziman T.V.

Labels: competitive programming, impartial games, nimber

View comments