Introduction to Combinatorial Games on Graphs

Combinatorial games are two-person games with perfect information and no chance moves, and with a win-or-lose outcome. Such a game is determined by a set of positions, including an initial position, and the player whose turn it is to move. Play moves from one position to another, with the players usually alternating moves, until a terminal position is reached. A terminal position is one from which no moves are possible. Then one of the players is declared the winner and the other the loser.

This theory may be divided into two parts, impartial games in which the set of moves available from any given position is the same for both players, and partizan games in which each player has a different set of possible moves from a given position. Games like chess or checkers in which one player moves the white pieces and the other moves the black pieces are partizan.

In this ZPC we are only concerned with impartial games in which the **last player** to make a **valid move** is the **winner**, and the other player thus is the loser. A **winning position** is a position which in an optimally played game, guarantees the victory of the player whose turn it currently is to move. Any state from which no valid moves are possible, thus becomes a losing position.

P-positions, N-positions:
All positions that are winning for the player that just made the move are called P-Positions or Previous-positions, while all those that are winning for the Next player to make the move are called N-positions. So a loser will end up in a P-position at the start of his/her turn.

In impartial combinatorial games, one can find in principle which positions are P-positions and which are N-positions by (possibly transfinite) induction using the following labeling procedure starting at the terminal positions. We say a position in a game is a terminal position, if no moves from it are possible.

Step 1: Label every terminal position as a P-position.
Step 2: Label every position that can reach a labelled P-position in one move as an N-position. Step 3: Find those positions whose only moves are to labelled N-positions; label such positions as P-positions.
Step 4: If no new P-positions were found in step 3, stop; otherwise return to step 2.

It is easy to see that the strategy of moving to P-positions wins. From a P-position thus, your opponent can move only to an N-position Then you may move back to a P-position. Eventually the game ends at a terminal position and since this is a P-position, you win.

Games Played on Directed Graphs:
We first give the mathematical definition of a directed graph.
Definition: A directed graph, G, is a pair (X, F) where X is a nonempty set of vertices (positions) and F is a function that gives for each $x \in X$ a subset of X, $F(x) \subset X$. For a given $x \in X$, F(x) represents the positions to which a player may move from x (called the followers of x). If F(x) is empty, x is called a terminal position.

A two-person win-lose game may be played on such a graph G = (X, F) by stipulating a starting position $x_0 \in X$ and using the following rules:

(1) Player I moves first, starting at x_0.
(2) Players alternate moves.
(3) At position x, the player whose turn it is to move chooses a position $y \in F(x)$.
(4) The player who is confronted with a terminal position at his turn, and thus cannot move, loses.

As defined, graph games could continue for an infinite number of moves. To avoid this possibility and other problems, we restrict attention to graphs that have the property that no matter what starting point x_0 is used, there is a number n, possibly depending on x0, such that every path from x_0 has length less than or equal to n. (A path is a sequence $x_0, x_1, x_2,\ldots,x_m$ such that $x_i \in F(x_i-1)$ for all i = 1,...,m, where m is the length of the path.) Such graphs are called progressively bounded. (If X itself is finite, this merely means that there are no circuits. A circuit is a path, $x_0, x_1,\ldots,x_m$, with x0 = xm and distinct vertices $x_0, x_1,\ldots,x_m-1$, $m \geq 1$.)

As an example let us consider a game with rules:

(1) There are two players. We label them I and II.
(2) There is a pile of n chips in the center of a table.
(3) A move consists of removing 1, 2, or 3 chips from the pile. At least one chip must be removed, but no more than three may be removed.
(4) Players alternate moves with Player I starting.
(5) The player that removes the last chip wins.

Here X = {0, 1,...,n} is the set of vertices. The empty pile is terminal, so $F(0) = \varnothing$, the empty set. We also have F(1) = {0}, F(2) = {0, 1}, and for $2 \leq k \leq n$, $F(k) = \{k-3, k-2, k-1\}$.

The Sprague-Grundy Function.
Graph games may be analyzed by considering P-positions and N-positions. It may also be analyzed through the Sprague-Grundy function.

Definition: The Sprague-Grundy function of a progressively bounded graph (X, F) is a function, g, defined on X taking non-negative integer values such that

$$g(x) = \min\{n \geq 0 : n = g(y) \text{ for } y \in F(x)\}.$$

In words, g(x) the smallest non-negative integer not found among the Sprague-Grundy values of the followers of x. If we define the minimal excludant, or mex, of a set of non-negative integers as the smallest non-negative integer not in the set, then we may write simply

$$g(x) = \text{mex}\{g(y) : \ y \in F(x)\}.$$

Note that g(x) is defined recursively. That is, g(x) is defined in terms of g(y) for all followers y of x. Moreover, the recursion is self-starting. For terminal vertices, x, the definition implies that g(x) = 0, since F(x) = ∅ for terminal x. For non-terminal x, all of whose followers are terminal, g(x) = 1, and so forth. In the examples in the next sections, we find g(x) inductively.

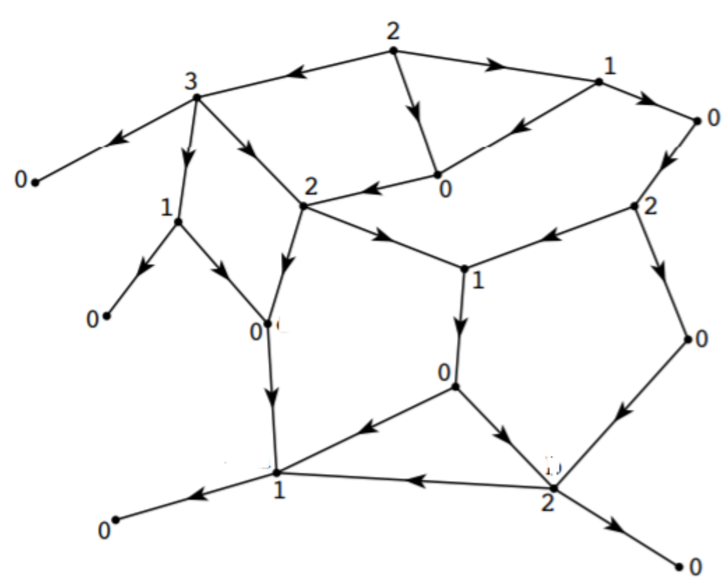
Q) What is the Sprague-Grundy function of the example (aka Subtraction game with subtraction set S={1,2,3}) game given above?
The terminal vertex, 0, has SG-value 0. The vertex 1 can only be moved to 0 and g(0) = 0, so g(1) = 1. Similarly, 2 can move to 0 and 1 with g(0) = 0 and g(1) = 1, so g(2) = 2, and 3 can move to 0, 1 and 2, with g(0) = 0, g(1) = 1 and g(2) = 2, so g(3) = 3. But 4 can only move to 1, 2 and 3 with SG-values 1, 2 and 3, so g(4) = 0. Continuing in this way we see

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
g(x)	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	...

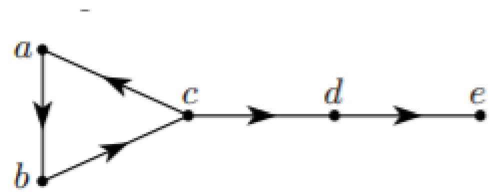
In general g(x) = x (mod 4), i.e. g(x) is the remainder when x is divided by 4.

The Use of the Sprague-Grundy Function.
Given the Sprague-Grundy function g of a graph, it is easy to analyze the corresponding graph game. **Positions x for which g(x) = 0 are P-positions and all other positions are N-positions**.
The winning procedure is to choose at each move to move to a vertex with Sprague-Grundy value zero. This is easily seen by checking the conditions of section on P-positions:
(1) If x is a terminal position, g(x) = 0.
(2) At positions x for which g(x) = 0, every follower y of x is such that g(y) ≠ 0, and
(3) At positions x for which g(x) ≠ 0, there is at least one follower y such that g(y) = 0.

Q). We will use the following figure to inductively find the SG values of the vertices.
Algorithm:
- All terminal positions are assigned SG values of 0.
- Search for a vertex all of whose followers have SG-values assigned. Then apply (1) or (2) to find its SG-value.
- Repeat until all vertices have been assigned values.



Graphs with cycles do not satisfy the Ending Condition. Play may last forever. In such a case we say the game ends in a tie; neither player wins. Here is an example where there are tied positions.



The node *e* is terminal and so has Sprague-Grundy value 0. Since *e* is the only follower of *d*, *d* has Sprague-Grundy value 1.
So a player at *c* will not move to *d* since such a move obviously loses. Therefore the only reasonable move is to *a*. After two more moves, the game moves to node *c* again with the opponent to move. The same analysis shows that the game will go around the cycle *abcabc* . . . forever.

So positions a, b and c are all tied positions. In this example the Sprague-Grundy function does not exist. When the Sprague-Grundy function exists in a graph with cycles, more subtle and non-trivial techniques are often required to find it.