Games

Hello, and welcome to Competitive Programming! Today we are going to talk about games.

Objectives

You will be able to describe how to model a game, use the minimax strategy to find good moves, consider some mathematical strategies to simplify game predictions, and finally know how to solve a classic game called Nim.

Modeling Games

Contest problems involving games typically come in two flavors: the first is to have the computer pick the best move for a game, and the second is to determine if it's always possible for a particular player to win or not.

In order to reason about the game, you have to be able to model the state of the game. This is a combination of whatever the configuration is plus the identity of the current player. A player taking a turn will change the configuration.

Once you have this model, you can simulate the game or study it.

Tic Tac Toe

So for example, let's suppose we want to model tic tac toe. There are nine squares, each of which can be empty, ex, or oh. Each time a player moves one of the empty

squares becomes an ex or an oh. You can then think of the game as a kind of tree; the configuration is a node, and a move takes you to the child nodes. So, the root node of tic tac toe is the empty board, and it has nine children, since there are nine empty squares.

Unless the game is incredibly small, you are not going to actually store this tree. Even tic tac toe has (3^9) possible board positions, just under 20,000. Instead, you will generate the child nodes as you go.

One thing you will need is some way to evaluate how good a board position is. Sometimes this is very simple, especially games in which you can know what a perfect move is. But in larger games this can be tricky since you can't always look at the board and see who will win or lose as a result.

In games like that, there is a technique called minimax. You want to maximize the board evaluation function after a computer move and minimize it after a competitor's move. Picking the move involves a recursive descent several moves into the game to see which move leads to the best results.

Mathematical Tricks

One common trick that happens in competitive programming is that the game has some kind of mathematical property that makes determining the best moves very simple.

The Multiplication Game, UVa problem 847 is a good example of this. The games starts out with a number (p) being equal to one, and the players take turns multiplying it by a number from 2 to 9. There is a target number (n), and the winner is the one who makes (p) greater than (n).

I'm not going to tell you the pattern, but you can discover it by trying out small instances of the game.

NIM

There is a game called NIM in which there are a certain number of piles of stones. They don't all have to have the same number at the start. When it's a players turn, they pick a non-empty pile and must remove at least one stone from it. They can remove as many stones as they want.

There are two versions of the game: the winner can be the one who takes the last stone, or else the one who forces the other player to take the last stone.

It turns out there is a simple pattern to determine the winner: if you take the exclusive or of all the pile sizes, and it is equal to zero, then the first player can always win. Perfect play then means making a move that preserves the exclusive or being 0.

You should study this particular game, because a lot of competitive programming game questions are really just Nim in disguise.

Next
Transcript — Graphs 2 (DFS and BFS) →