Combinatorial Games

Jaehyun Park

CS 97SI
Stanford University

June 29, 2015

Combinatorial Games

- Turn-based competitive multi-player games
- Can be a simple win-or-lose game, or can involve points
- Everyone has perfect information
- Each turn, the player changes the current "state" using a valid "move"
- At some states, there are no valid moves
  - The current player immediately loses at these states

Outline

Simple Games

Minimax Algorithm

Nim Game

Grundy Numbers (Nimbers)

Combinatorial Game Example

- Settings: There are n stones in a pile. Two players take turns and remove 1 or 3 stones at a time. The one who takes the last stone wins. Find out the winner if both players play perfectly
- State space: Each state can be represented by the number of remaining stones in the pile
- Valid moves from state x: $x \to (x-1)$ or $x \to (x-3)$, as long as the resulting number is nonnegative
- State 0 is the losing state

Example (continued)

- No cycles in the state transitions
  - Can solve the problem bottom-up (DP)
- A player wins if there is a way to force the opponent to lose
  - Conversely, we lose if there is no such a way
- State x is a winning state (W) if
  - $(x-1)$ is a losing state,
  - OR $(x-3)$ is a losing state
- Otherwise, state x is a losing state (L)

Example (continued)

- DP table for small values of n:

n	0	1	2	3	4	5	6	7
W/L	L	W	L	W	L	W	L	W

- See a pattern?

- Let's prove our conjecture

Example (continued)

- Conjecture: If n is odd, the first player wins. If n is even, the second player wins.

- Holds true for the base case $n = 0$
- In general,
  - If n is odd, we can remove one stone and give the opponent an even number of stones
  - If n is even, no matter what we choose, we have to give an odd number of stones to the opponent

Outline

Simple Games

Minimax Algorithm

Nim Game

Grundy Numbers (Nimbers)

More Complex Games

- Settings: a competitive zero-sum two-player game
- Zero-sum: if the first player's score is x, then the other player gets $-x$
- Each player tries to maximize his/her own score
- Both players play perfectly

- Can be solved using a *minimax* algorithm

Minimax Algorithm

- Recursive algorithm that decides the best move for the current player at a given state
- Define $f(S)$ as the optimal score of the current player who starts at state S
- Let $T_1, T_2, \ldots, T_m$ be states can be reached from S using a single move
- Let T be the state that minimizes $f(T_i)$
- Then, $f(S) = -f(T)$
  - Intuition: minimizing the opponent's score maximizes my score

Memoization

- (Not *memorization* but *memoization*)
- A technique used to avoid repeated calculations in recursive functions
- High-level idea: take a note (memo) of the return value of a function call. When the function is called with the same argument again, return the stored result
- Each subproblem is solved at most once
  - Some may not be solved at all!

Recursive Function without Memoization

```
int fib(int n)
{
    if(n <= 1) return n;
    return fib(n - 1) + fib(n - 2);
}
```

- How many times is `fib(1)` called?

Memoization using `std::map`

```
map<int, int> memo;
int fib(int n)
{
    if(memo.count(n)) return memo[n];
    if(n <= 1) return n;
    return memo[n] = fib(n - 1) + fib(n - 2);
}
```

- How many times is `fib(1)` called?

Minimax Algorithm Pseudocode

- Given state S, want to compute $f(S)$

- If we know $f(S)$ already, return it
- Set return value $x \leftarrow -\infty$
- For each valid next state T:
  - Update return value $x \leftarrow \max\{x, -f(T)\}$
- Write a memo $f(S) = x$ and return x

Possible Extensions

- The game is not zero-sum
  - Each player wants to maximize his own score
  - Each player wants to maximize the difference between his score and the opponent's
- There are more than two players

- All of above can be solved using a similar idea

Outline

Simple Games

Minimax Algorithm

Nim Game

Grundy Numbers (Nimbers)

Nim Game

- Settings: There are n piles of stones. Two players take turns. Each player chooses a pile, and removes any number of stones from the pile. The one who takes the last stone wins. Find out the winner if both players play perfectly

- Can't really use DP if there are many piles, because the state space is huge

Nim Game Example

- Starts with heaps of 3, 4, 5 stones
  - We will call them heap A, heap B, and heap C

- Alice takes 2 stones from A: $(1, 4, 5)$
- Bob takes 4 from C: $(1, 4, 1)$
- Alice takes 4 from B: $(1, 0, 1)$
- Bob takes 1 from A: $(0, 0, 1)$
- Alice takes 1 from C and wins: $(0, 0, 0)$

Solution to Nim

- Given heaps of size $n_1, n_2, \ldots, n_m$
- The first player wins if and only if the *nim-sum* $n_1 \oplus n_2 \oplus \cdots \oplus n_m$ is nonzero ($\oplus$ is bitwise XOR operator)

- Why?
  - If the nim-sum is zero, then whatever the current player does, the nim-sum of the next state is nonzero
  - If the nim-sum is nonzero, it is possible to force it to become zero (not obvious, but true)

Outline

Simple Games

Minimax Algorithm

Nim Game

Grundy Numbers (Nimbers)

Playing Multiple Games at Once

- Suppose that multiple games are played at the same time. At each turn, the player chooses a game and make a move. You lose if there is no possible move. We want to determine the winner

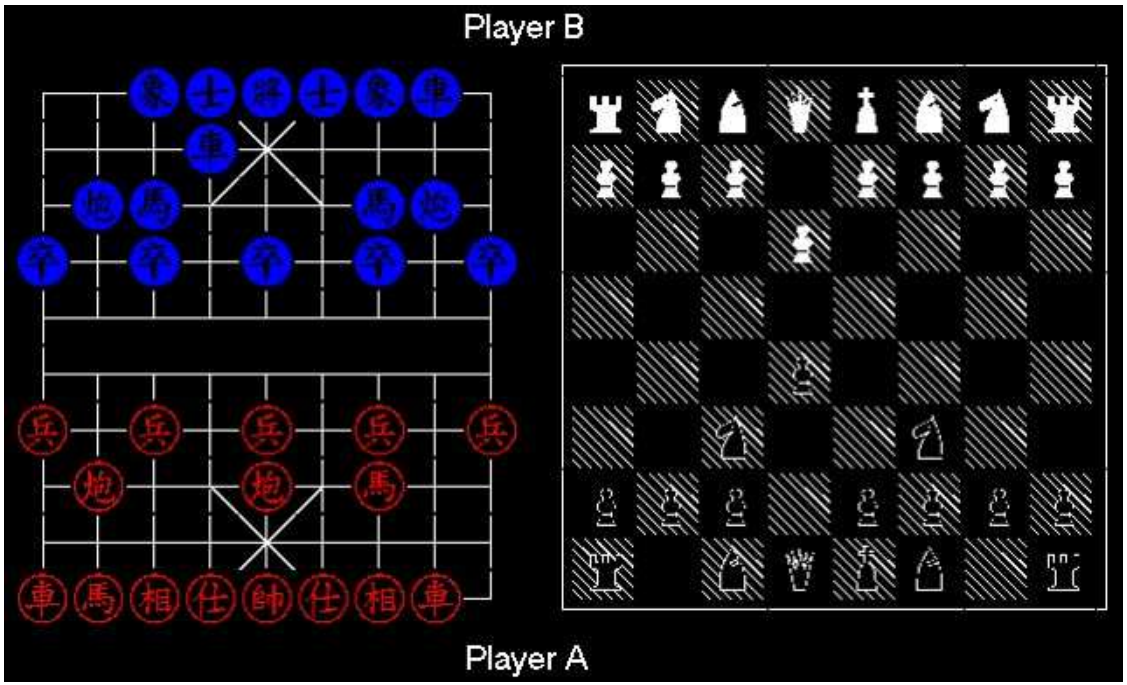

Figure from `http://sps.nus.edu.sg/~limchuwe/cgt/`

Grundy Numbers (Nimbers)

- For each game, we compute its *Grundy number*
- The first player wins if and only if the XOR of all the Grundy numbers is nonzero
  - For example, the Grundy number of a one-pile version of the nim game is equal to the number of stones in the pile (we will see this again later)

- Let's see how to compute the Grundy numbers for general games

Grundy Numbers

- Let S be a state, and $T_1, T_2, \ldots, T_m$ be states can be reached from S using a single move

- The Grundy number $g(S)$ of S is the smallest nonnegative integer that doesn't appear in $\{g(T_1), g(T_2), \ldots, g(T_m)\}$
  - Note: the Grundy number of a losing state is 0
  - Note: I made up the notation $g(\cdot)$. Don't use it in other places

Grundy Numbers Example

- Consider a one-pile nim game
- $g(0) = 0$, because it is a losing state
- State 0 is the only state reachable from state 1, so $g(1)$ is the smallest nonnegative integer not appearing in $\{g(0)\} = \{0\}$. Thus, $g(1) = 1$
- Similarly, $g(2) = 2$, $g(3) = 3$, and so on
- Grundy numbers for this game is then $g(n) = n$
  - That's how we got the nim-sum solution

Another Example

- Let's consider a variant of the game we considered before; only 1 or 2 stones can be removed at each turn
- Now we're going to play many copies of this game at the same time
- Grundy number table:

n	0	1	2	3	4	5	6	7
$g(n)$	0	1	2	0	1	2	0	1

Another Example (continued)

- Grundy number table:

n	0	1	2	3	4	5	6	7
$g(n)$	0	1	2	0	1	2	0	1

- Who wins if there are three piles of stones $(2, 4, 5)$?
- What if we start with $(5, 11, 13, 16)$?
- What if we start with $(10^{100}, 10^{200})$?

Tips for Solving Game Problems

- If the state space is small, use memoization
- If not, print out the result of the game for small test data and look for a pattern
  - This actually works really well!
- Try to convert the game into some nim-variant
- If multiple games are played at once, use Grundy numbers