THINK FAST

Competitive Programming

From Problem 2 Solution in O(1)

Combinatorial Game Theory

Disjunctive and Selective Compound Games

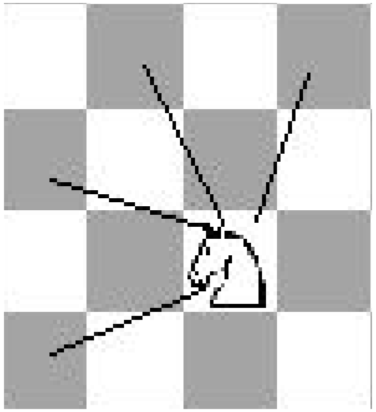
Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Recall Nim game properties

- Impartial game
  - **Same set of moves** at any time allowed for both players
- 2 players play sequentially
- Each pile is independent sub-game
- Perfect information
- Finite game, No draws, No randomization
- Winner = last move
  - Loser = can't make a move
- Such game => Sprague–Grundy theorem

Recall: Knights on chessboard

- Given N x N chessboard with K knights on it
  - Knight Move is to 4 positions only
  - Player turn: Pick a knight and move it
  - Allowed: Multiple knights in 1 position
  - Loser: Player who can’t make a move to knight
- Equivalent game to Nim
  - Knight move is independent nim pile
  - Compute pile size of knights using grundy
  - xor the knights grundy
  - xor ==0 $\Rightarrow$ Loser



Recall: Knights on chessboard

```
int grundy2[120][120];

bool valid(int v) {
  return v >= 0 && v < 8;
}

int calcGrundyChess(int r, int c) {
  int &ret = grundy2[r][c];
  if (ret != -1)
    return ret;

  unordered_set<int> sub_nimbers;

  const int DIR = 4;
  int dr[DIR] = { 1, -1, -2, -2 };
  int dc[DIR] = { -2, -2, 1, -1 };

  for (int d = 0; d < 4; ++d) {
    if (valid(r + dr[d]) && valid(c + dc[d]))
      sub_nimbers.insert(calcGrundyChess(r + dr[d], c + dc[d]));
  }
  return ret = calcMex(sub_nimbers);
}
```

0	0	1	1	0	0	1	1
0	0	2	1	0	0	1	1
1	2	2	2	3	2	2	2
1	1	2	1	4	3	2	3
0	0	3	4	0	0	1	1
0	0	2	3	0	0	2	1
1	1	2	2	1	2	2	2
1	1	2	3	1	1	2	0

Compound games

- Conway proposed **12** compound games
  - **3 movements styles**
    - **Disjunctive**: Move in **one** sub-game
    - **Selective**: Move in at **least one** sub-game
    - **Conjunctive**: Move in **all** sub-games
  - **2 ending rules**: **long rule or short rule**
    - Long rule: Game ends when **all sub-games** are done
    - Short term: Game ends **once a sub-game** is done
    - Short term: also called WTIA (winner takes it all)
  - **2 plays**: **normal or misere**
  - So total: **3 x 2** x 2 = 6 x 2 = 12 games
  - We studied 2 out of 12: See **blue words** above

Disjunctive compound game

- What we studied so far (2 games)
  - We studied **Normal Nim** and its **Misere** using **long** rule
    - Keep playing till last pile is empty (long rule)
  - It is the most popular versus the other 10 games
  - Nim was based on xor of piles.
  - Nim misere was **almost** as Nim except **bottom cases**
  - We used Grundy function for complex games
  - Grundy is not defined for Nim misere
- **Diminished disjunctive compound**
  - Same game, buts uses the **short** rule
  - Once a sub-game (pile, knight) is done, game is over

Other compound game

- Remaining 5 x 2 games
  - They don't offer real new theory
  - They are either
    - Direct make use of **grundy**
    - Little **modification** to **grundy**
    - Compute # steps in **minimax** or **maximin** styles
  - Please after **presenting** every one of the 10 games
    - Consider it a challenge to solve by yourself
    - Think about the terminal cases
    - Think about the possible winning strategy

Diminished Disjunctive compound

- Let’s modify Knights on chessboard game
  - Old game: Loser can’t move any knight
  - New game: Loser have **one knight** in its **final position**
    - E.g. the winner moves a knight to its final position
    - E.g. In Nim, 1 pile is now empty (short rule)
  - **Your turn**: Solve it under **normal** and **misere** plays
  - Hints
    - Think about terminal losing positions
    - Think about positions before the losing positions
    - Modify grundy to consider above 2 notes
    - Misere will be very close to normal play in this game

Diminished Disjunctive compound

- Solution
  - Identify positions where a knight in a **terminal** position
    - Call these **super losing** positions (SL) or TP
  - Identify all knight positions that are 1 step from a SL
    - Then these are **Super winning** position. Call them SW
  - Call any other position a **normal** position
    - These positions follow **normal nim strategy**
    - Compute **grundy** for these positions
    - **Mex** on all sub-moves *except SL and SW*
  - This solution style is close to Misere Nim
    - Normal nim unless you are close to the bottom cases

Diminished Disjunctive compound

Let W-number(x) = w(x): {SL | SW | Grundy*(X)}

SL	SL	SW	SW	0	0	1	1
SL	SL	SW	SW	0	0	2	1
SW	SW	SW	SW	1	2	3	2
SW	SW	SW	1	2	1	2	1
0	0	1	2	0	0	1	2
0	0	2	1	0	0	2	1
1	2	3	2	1	2	1	2
1	1	2	1	2	1	2	0

w-numbers for 8×8 All the King's horses modification

Src: https://www.cs.ox.ac.uk/files/2735/Composite_games.pdf

Diminished Disjunctive compound

- Some logic
  - If we are in SL or SW, it is a special handling
  - If we are in a normal position and play normal nim
    - Next move may be to normal position
    - Or to a SW position (it can’t be to SL or we r in SW)
  - In optimal setting, you won’t move to SW if can
    - As Opponent will just make 1 move to win
    - So always move to **normal nim position if possible**
  - In other words, if all availables moves are to SW => lose
  - Given that optimal strategy is to keep playing like a nim, computing grundies is ok (unless SL, SW positions)
    - Skip SW from the grundy computations

Diminished Disjunctive compound

- X is Losing position
  - w(x) == SL **OR** W(x) = 0
  - Recall: W(x) = Grundy*(x) = 0 => losing in normal nim
- X is Winning position
  - w(x) != SL **AND** (w(x) == SW **OR** w(x) > 0)
- Combining sub-games: g1, g2...gn
  - If **any** sub-game is SL => lose
  - Otherwise, If any sub-game is SW => win
  - Otherwise, xor the grundies: g1 ^ g2 …. ^ gn != 0 => win
  - This helps **computationally**: Once **a sub-game is SL** you don’t need to compute others

Grundies Vs W-numbers

0	0	1	1	0	0	1	1
0	0	2	1	0	0	1	1
1	2	2	2	3	2	2	2
1	1	2	1	4	3	2	3
0	0	3	4	0	0	1	1
0	0	2	3	0	0	2	1
1	1	2	2	1	2	2	2
1	1	2	3	1	1	2	0

SL	SL	SW	SW	0	0	1	1
SL	SL	SW	SW	0	0	2	1
SW	SW	SW	SW	1	2	3	2
SW	SW	SW	1	2	1	2	1
0	0	1	2	0	0	1	2
0	0	2	1	0	0	2	1
1	2	3	2	1	2	1	2
1	1	2	1	2	1	2	0

Recall: w(x): {SL | SW | grundy*(X)}
grundy* = Don't consider SWs

If w(x) = 0 ⇒ grundy(x) = 0
if grundy(x) = 0 ⇒ w(x) = 0 is NOT true

Different values
3 = mex(0, 1, 2)
1 = mex(0, SW) = mex(1)

Diminished Disjunctive compound

```
const int SL = 100000000;
const int SW = SL + 1;

int calcWNumbersChess(int r, int c) {
  int &ret = wNumbers[r][c];
  if (ret != -1)
    return ret;

  unordered_set<int> sub_nimbers;

  int total_moves = 0;
  for (int d = 0; d < 4; ++d) {
    int nr = r + dr[d], nc = c + dc[d];
    if (valid(nr) && valid(nc)) {
      int val = calcWNumbersChess(nr, nc);
      if (val == SL)
        return ret = SW;  // optimization
      if (val != SW)
        sub_nimbers.insert(val);
      ++total_moves;
    }
  }
  if (total_moves == 0)
    return ret = SL;
  return ret = calcMex(sub_nimbers);
}
```

```
int nimXor = 0, knights;
bool anySuperLose = false, anySuperWin = false;

cin >> knights;
for (int d = 0; d < knights; ++d) {
  int x, y;
  cin >> x >> y;
  int val = calcWNumbersChess(x, y);

  if (val == SL) {
    anySuperLose = true;
    break;  // optmization
  } else if (val == SW)
    anySuperWin = true;
  else
    nimXor ^= val;
}
if (anySuperLose)
  cout << "Second win*";
else if (anySuperWin)
  cout << "First win*";
else if (nimXor != 0)
  cout << "First win";
else
  cout << "Second win";
```

Diminished Disjunctive compound

- Your turn
  - Given N piles, and you can remove only {1, 3, 4} stones from a single pile
  - Once any pile is empty, game ends (WTIA)
  - Write code to compute w-numbers and compare it

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13
w(n)	SL	SW	0	SW	SW	1	2	0	2	0	1	3	1	2
g(n)	0	1	0	1	2	3	2	0	1	0	1	2	3	2

Src: https://www.cs.ox.ac.uk/files/2735/Composite_games.pdf

Diminished Disjunctive - Misere

- Under Misere
  - Recall, disjunctive game under Misere has no grundies
  - The terminal positions now are winning positions
    - Let's call them Super Winning SW
  - Again, from a normal position we have 2 choices
    - To normal position
    - To winning position (don't move there unless forced)
  - So we again play normal nim, except bottom cases
    - So again, compute grundies, but ignore SW
    - Values propagate normally from bottom to top
  - Given set of games:
    - If any position is SW (win), otherwise xor != 0 (win)

Diminished Disjunctive - Misere

SW	SW	0	0	0	1	0	0
SW	SW	1	0	2	2	1	0
0	1	1	1	2	1	4	1
0	0	1	0	3	0	1	0
0	2	2	3	0	1	0	1
1	2	1	0	1	0	1	0
0	1	4	1	0	1	0	1
0	0	1	0	1	0	1	0

Try to get intuition why grundy misere failed for Disjunctive Compound, but worked for Diminished Disjunctive Compound case

Diminished Disjunctive - Misere

```
int calcWNumbersChessMisere(int r, int c) {
  int &ret = wNumbers[r][c];
  if (ret != -1)
    return ret;

  unordered_set<int> sub_nimbers;

  int total_moves = 0;
  for (int d = 0; d < 4; ++d) {
    int nr = r + dr[d], nc = c + dc[d];
    if (valid(nr) && valid(nc)) {
      int val = calcWNumbersChessMisere(nr, nc);
      if (val != SW)
        sub_nimbers.insert(val);
      ++total_moves;
    }
  }
  if (total_moves == 0)
    return ret = SW;
  return ret = calcMex(sub_nimbers);
}
```

```
int nimXor = 0, knights;
bool anySuperWin = false;

cin >> knights;
for (int d = 0; d < knights; ++d) {
  int x, y;
  cin >> x >> y;
  int val = calcWNumbersChessMisere(x, y);

  if (val == SW)
    anySuperWin = true;
  else
    nimXor ^= val;
}
if (anySuperWin)
  cout << "First win*";
else if (nimXor != 0)
  cout << "First win";
else
  cout << "Second win";
```

Compound games

- Selective compound (long - short rules)
  - Move **at least 1** knight
  - Move **at least 1** knight but **NOT all**
  - Make use of grundy
- Conjunctive compound (next session)
  - Move **all** knights (short rule)
  - Minimax style game to count the game # steps
  - E.g. introduce **Remoteness function**
- Continued conjunctive compound
  - Move in **all available** knights (long rule)
  - Use **Suspense function** (count the game # steps)

Selective compound

- Game (long rule)
  - Move **at least 1** knight
  - Assume we have piles (2, 3, 4): How to win?
  - Think :)

Selective compound

- Game (long rule)
  - Move **at least 1** knight
  - Assume we have piles (2, 3, 4): How to win?
  - Simply take from all
  - What about input (0, 0, 0)? We directly lose
  - Overall, compute sub-games grundies
  - If ALL grundies = 0 $\Rightarrow$ Losing position
  - Otherwise, just go and win in every one

Selective compound - Misere

- Assume piles
  - (0, 0, 0) => first win (base case)
  - (0, 0, 1) => (0, 0, 0) => Lose (1 non-losing component)
  - (0, 1, 1) => (0, 0, 1) => (0, 0, 0) => Win
  - (0, 3, 4) => (0, 0, 1) => (0, 0, 0) => Win
- So compute grundies
  - All losing => first win
  - Only one losing => first lose
  - Other cases => first win

Shortened Selective compound

- Selective compound (short rule)
  - Solution is same as previous
    - If all grundies = 0 => lose
  - Even Misere case: Same solution exactly
    - Hene 3 scenarios things have same rule (grundies = 0)
  - Please verify

Selective compound - Variant

- Game (long rule)
  - Move **at least 1** knight but **NOT all**
  - Hints:
    - Playing around nim piles game might help in thinking
    - What about input piles (2, 3, 4)?
    - What about input piles (2, 2, 2)?
  - Think :)

Selective compound - Variant

- Game (long rule)
  - Move **at least 1** knight but **NOT all**
  - What about input piles (2, 3, 4)?
    - Convert to equal piles (e.g. find min val) => (2, 2, 2)
    - Second can change to non equal => e.g. (1, 0, 2)
    - Convert to equal => (0, 0, 0)
  - What about input (3, 3, 3)?
    - From above, if all piles = val $\Rightarrow$ losing state
  - So winning strategy
    - Keep converting to **equal length piles**. Last (0,0,0)
  - Overall, compute sub-games grundies
    - If all **grundies are equal** $\Rightarrow$ **Losing** position

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً