P THINK FAST S

Competitive Programming

From Problem 2 Solution in O(1)

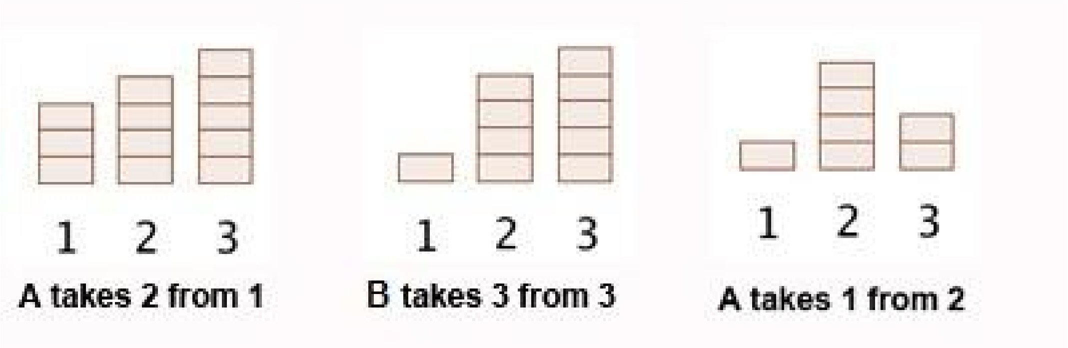
Combinatorial Game Theory

Game of Nim

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

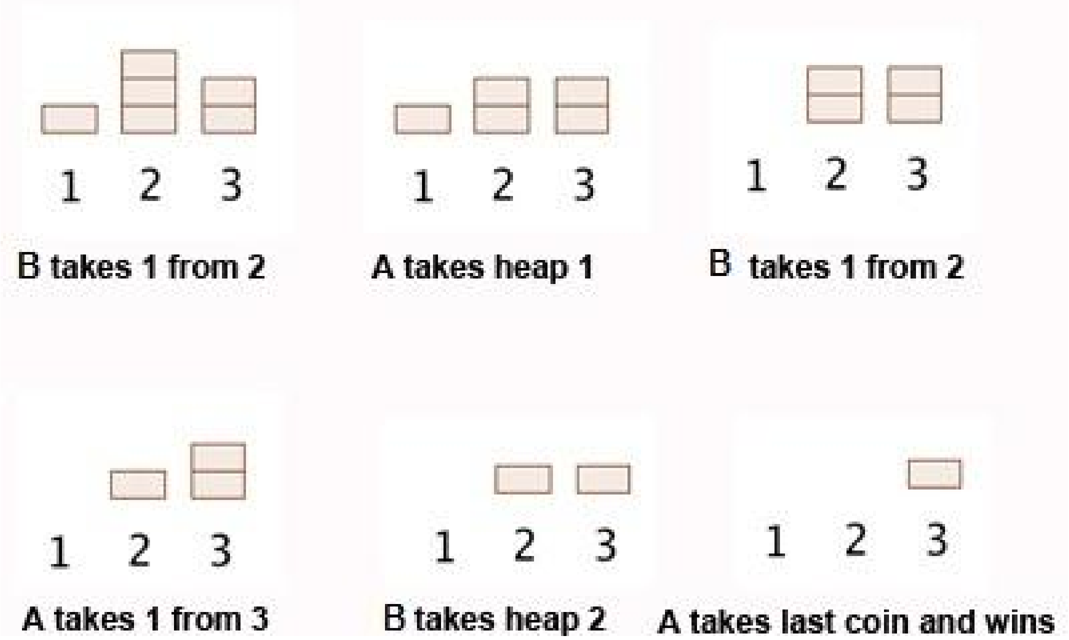
Game of Nim

- K piles (nim heaps), each has some stones
  - Let pile sizes: n1, n2, …nk
  - Player selects 1 pile and take 1 or more items from it
  - Winner = takes last stone
  - E.g. for piles (3, 4, 5)



Src: http://www.geeksforgeeks.org/combinatorial-game-theory-set-2-game-nim/

Game of Nim


1 2 3
B takes 1 from 2
1 2 3
A takes heap 1
1 2 3
B takes 1 from 2
1 2 3
A takes 1 from 3
1 2 3
B takes heap 2
1 2 3
A takes last coin and wins

Game of Nim

- It is pretty hard to write **efficient** recursive solution to this problem!
- Can we have a simple strategy to determine a solution? Yes
  - Compute xor for piles: n1 xor n2 xor …. nk (*Nim-sum*)
  - If result = 0 => First player lose, otherwise win

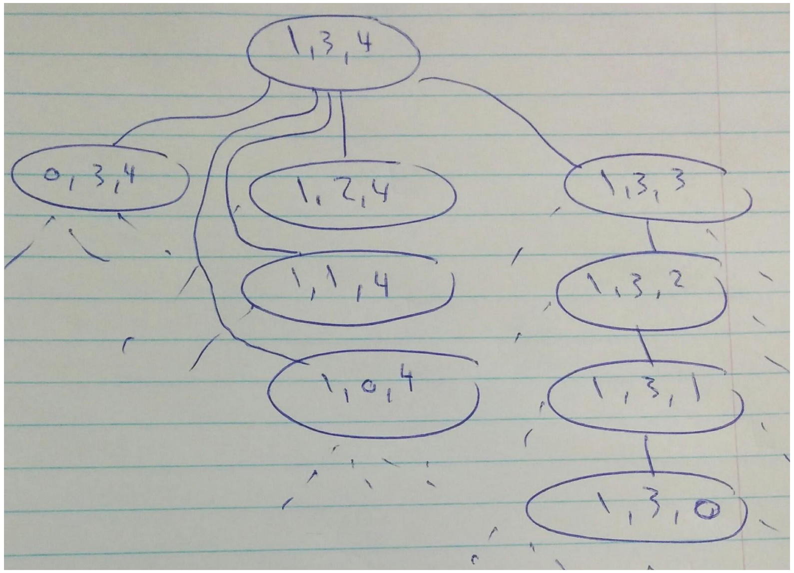
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Game of Nim

- For pile sizes: 6, 9, 3
  - 6 = 0 1 1 0
  - 9 = 1 0 0 1
  - 3 = 0 0 1 1
  - ^ = 1 1 0 0 => column = 1 IFF # of 1s is odd
  - Result != 0 => First player wins
- Piles (1, 2, 3) => xor = 0 ⇒ losing position
- Piles (1, 4, 7) => xor != 0 ⇒ wining position
- **Nim sum = Nim addition = a ^ b**

Nim Game Tree

- Assume we have input piles (1, 3, 4)
  - To compute answer, draw tree of all possible choices
  - If node sum is N (e.g. 8), we have N children for this node
  - Compute Win/lose positions. Losing Base Case (0, 0, 0)



Nim Xor Computations

- To prove correctness, our search results must match xor results (sum = 0 $\Rightarrow$ lose)
  - We need to prove following 3 properties:
  - Same base case result: Trivially correct, as xor (0, 0, 0) = 0 => losing state in both ways
  - If we are in a **losing position**, then **all** child positions are winning positions (<u>in other words</u>, there is **no child** position with xor = 0)
  - If we are in **winning position**, there is **at least** a child position with losing state

Losing position verification

- Assume given x = y > 0, then x^y = 0
  - Assume we allow only decreasing x
  - Can you find **any** (x-v)^y = 0? **Never**.
    - Any change in x, changes the overall bits.
    - Then, some columns will have xor != 0
  - Then all children positions have xor > 0 ⇒ win positions
  - Let x = 4, y = 4
  - 100 ^ 100 = 0 ⇒ let's try all x children
  - 011 ^ 100 != 0
  - 010 ^ 100 != 0
  - 001 ^ 100 != 0
  - 000 ^ 100 != 0

Winning position verification

- Assume given x > y > 0 with x^y != 0
  - Assume we allow only decreasing x
  - Can you find **any** (x-v)^y = 0? **YES**.
    - As x > y, left most bit in x comes <= one in y
    - One can alter any bit in x starting from most left one
    - So find all columns with xor = 1 and flip bit in X
    - In other words, set x = y
  - Then one of children positions have xor is losing positions
  - 10110010 ^ 01000001 = 11110011
  - Alter 10110010 to 01000001
  - 01000001 ^ 01000001 = 0

Position verification

- What if we have array A of numbers?
  - Find a number in A > the xor of remaining (**always** exist)
  - Let X = A[i] > xorsum^A[i]
    - Notice: a ^ (a^**b^c^d**) = **b^c^d**
  - Let Y = xor of remaining numbers = xorsum^A[i]
  - Actually, this also help us simulate the game!
  - For other ways for the proof: Link 1, Link 2, Link 3
- Overall: xor tree matches recursive tree
  - Same base case computation
  - xor rule for losing position has always all children win
  - xor rule for win position have always 1+ lose child
  - Two equivalent trees

Nim Winning Strategy

- If current xor !=0 => I can win
- So if you are in a game state, make the move that makes the next state XOR = 0
- Hence user in losing state
- Whatever he did, he will return me to xor != 0
- Again and again till all piles = 0

Solving Nim and Finding a move

```
void solve_nim(vector<int> heap) {
  int xorsum = 0;
  for (int i = 0; i < (int)heap.size(); ++i)
    xorsum ^= heap[i];

  if (xorsum != 0) {
    cout << "First win: ";

    for (int i = 0; i < (int)heap.size(); ++i) {
      if (heap[i] > (heap[i] ^ xorsum)) {
        cout<< "First Move " << (heap[i] - (heap[i] ^ xorsum))
            << " stones from " << i << " heap\n";
        break;
      }
    }
  } else
    cout << "Second win\n";
}
```

Misère Nim

- Same game as Nim, but the last one to make move loses (e.g. base case reversed)
  - Generally misère rule is more difficult to analyze
  - Even sometimes, you can't find a strategy
- Same rule too, with special case handling **if all sizes are 1s or 0s**
  - First player win if # of 1s is even (their xor = 0)
  - Otherwise, we use the **normal nim rule** (xor != 0 => win)

Misère Nim

```
bool misereNim(vector<int> &piles) {
  int isSpecial = 1, xorVal = 0;

  for(int i = 0; i < piles.size(); ++i) {
    xorVal ^= piles[i];
    isSpecial &= (piles[i] <= 1);
  }
  if (isSpecial)
    return xorVal == 0;
  return xorVal != 0; // normal nim handling
}
```

Misère Winning Strategy

- So we are almost Nim strategy!
  - Keep pushing the other player for piles xor = 0
  - Exception: Never create a state of even non-empty piles of size = 1
- Normal nim strategy is same everywhere, so nothing special about the game tree leaf nodes.
- However, Misère is dependent on the leaves
- We call normal nim ‘normal’, as it is logical who can’t move is loser

Misère: Little analysis

- So wired that Nim is **almost** as Misère Nim!
- Let S1 be starting state (with some piles > 1)
- Let S3 be state that has odd piles of size 1
- Let S2 be state directly before S3 (e.g. 1 pile only of size > 1 and others = 1)
- So according to Misère
  - If S1 xor != 0, you will win: then
  - Move S1 through normal nim steps to S2
  - Hanle S2 carefully to move to S3
  - The question: Does S2 **always** all that handling?

Misère: Little analysis

- Let S1 be some initial state
- S3 = 1 1 1 [odd ones = 1st lose - base case]
- What might be possible previous state S2?
- S2 = [5 1 1]: Take 4 from 5 [1 1 1]: I win
- S2 = [5 1 1 1]: Take 5 from 5 [1 1 1]: I win
- S2 can force winning whatever its content
- Note also, xor S2 always != 0
- Hence, if xor S1 != 0, move from S1 to S2 using normal nim, and then force winning

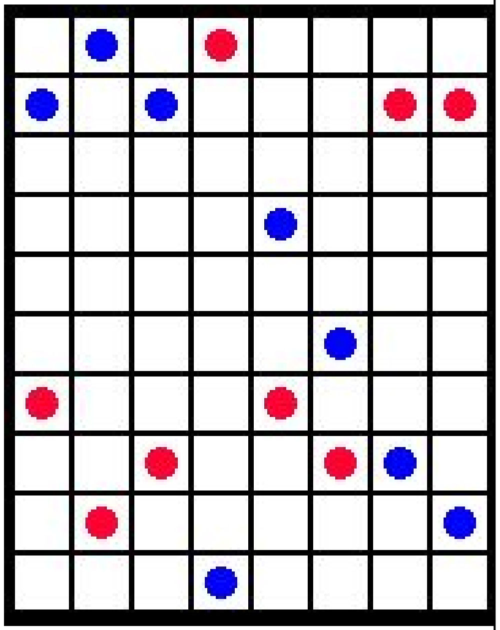
Moore's nim-k

- Generalization of nim, given N piles and K
  - Remove stones from any k piles
  - Expand every pile size in base 2
  - For **every** column, check if 1s sum divisible by **k+1**
    - If this is true **for all** => first player LOSEs
  - Example: piles (1, 2, 3, 3), k = 2
    - 01
    - 10
    - 11
    - 11 [please **prove** - see links above]
  - Column 1 sum 3%3 = 0, Column 2 sum 3%3 = 0
  - All divible by k+1 (3) ⇒ First player loses

Northcott game

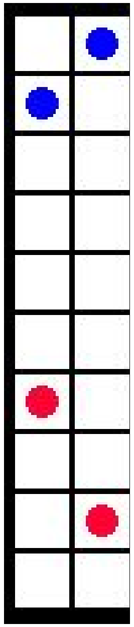
- Find win strategy. Play it here.
  - Board with red and blue opponents
  - Each column has 1 red and 1 blue ball
  - For a column, move only your color

    toward the other color (any # of steps)
  - BUT can't jump over opponent
  - Winner = Last to move his color



Northcott game

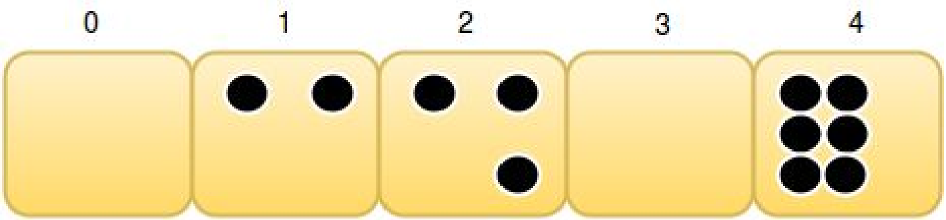
- Thinking
  - **Observation**: Satisfies impartial game conditions
  - **Observation**: Every column is independent from others
  - **Intuition**: Each column is a nim pile
  - **Intuition**: If distance between opponents decrease, nim decrease in same amount
  - **Solution**: Convert each column to a pile size (# of cells between 2 colors). Then do xor for the pile sizes!



- What if moving can be backward too?
  - Same solution
  - Use **move cancellation strategy** (Bogus/Poker Nim)

Nimble game

- Play here (and many other games)
- Solve at judge
  - Array of N cells, that has coins
  - Move: Pick a coin and move to **any left square**, even if it has 1 or more coin
  - End condition: All coins in first cell in array
  - Loser: Can't do a move
  - Sol: every coin is **independent** than others (e.g. a pile)
  - Xor values arr[i]-1



Your turn: 21 game

- 21 game
  - 2 players, first one says 1
  - Next player can say **previous number + (1 or 2 or 3)**
  - The first to exceed 20 lose
  - E.g. 1, 4, 6, 8, 11, 12, 13, 16, 19, 20, 21 (increasing game)
  - Question: is this normal play or **misère**?
    - Loser is **last one** to do action = This is **misère**
  - Find a simple winning strategy for it / variants game
  - See also
  - Solve also the 100 game

Your turn: Nim with skip move

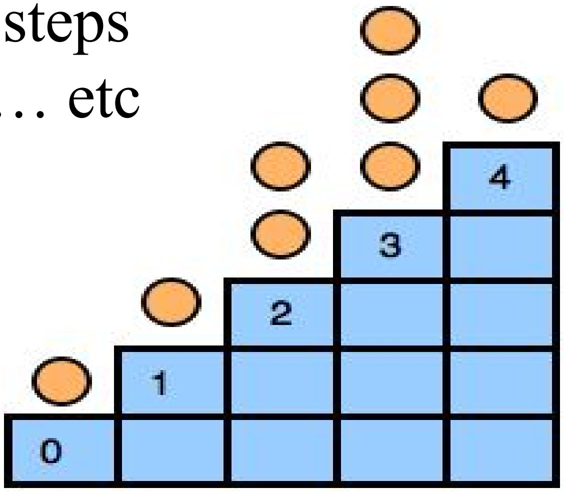
- Given the original nim game with extra rule
  - Skip turn: A player is allowed to say I will skip turn
  - Player 1 is allowed up to A skippings
  - Player 2 is allowed up to B skippings
  - Given the N piles, A, B who is the winner?
  - Hints:
    - What if A = B? Think in **Move Cancellation Strategy**
    - What if A > B and in normal N piles you lose?

Your turn: Dividing a number

- Given an integer N
  - Move: divide N by a prime power > 1
    - e.g. 3, 3^2, 3^3…
    - primes: 2, 3, 5, 7, 11….
  - Loser: N = 1
  - Next time the solution

Your turn: Staircase Nim

- Staircase Nim
  - Staircase with n steps, each step has some coins
  - Move: move some coins to the left step (except first step)
  - Loser: Can’t make a move (e.g. all coins at arr[0])
  - **Intuition**: Every step is a pile?
  - **Issue**: Notice Dependency between steps
  - **Hint**: Analyze positions: 0, 1, 2, 3, … etc
  - Next time the solution



Src: http://codeforces.com/blog/entry/44651

Summary

- Nim
  - N piles, remove from 1 pile, winner = last move
  - Just Xor them => xor == 0 ⇒ Lose
  - For **every** column, check if 1s sum divisible by **2**
    - If so, lose
- Misère Nim
  - N piles, remove from 1 pile, loser = last move
  - Same general solution of nim, except if all piles <= 1
- Moore's nim-k
  - Remove stones from any k piles
  - For **every** column, check if 1s sum divisible by **k+1**

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً