Game Theory - GeeksforGeeks

Source: https://www.geeksforgeeks.org/game-theory/

Courses Tutorials Practice Jobs DSA Practice Problems C C++ Java Python JavaScript Data Science Machine Learning Courses Linux DevOps Technical Scripter 2026 Explore Basics DSA Tutorial Maths for DSA Mathematical Algorithms Bit manipulation Bit Manipulation for Competitive Programming Bit Tricks for Competitive Programming Bitwise Hacks for Competitive Programming DP for CP Dynamic Programming (DP) Introduction Dynamic Programming or DP DP on Trees for Competitive Programming Dynamic Programming in Game Theory for Competitive Programming Advanced Graph Algorithms Segment Tree Binary Indexed Tree or Fenwick Tree Array Range Queries Three 90 Challenge 90% Refund Game Theory Last Updated : 22 Oct, 2025 Game Theory is a topic in competitive programming that involves a certain type of problem, where there are some players who play a game based on given rules and the task is often to find the winner or the winning moves. Game Theory is often asked in short contests with a mixture of other topics like range querying or greedy or dynamic programming. Game Theory for Competitive Programming: Here we will focus on two-player games that do not contain random elements. Our goal is to find a strategy we can follow to win the game no matter what the opponent does if such a strategy exists. Game theory or combinatorics game theory in which we have perfect information (that is no randomization like a coin toss) such as game rules, player's turn, minimum and maximum involved in the problem statements, and some conditions and constraints. There will be three possible cases/ state win, loss or tie. A terminal condition is well-defined/ specified clearly. E.g. player who picks the last coin will win the game, or a player who picks the second last time coin will win the game or something like that. It is assumed that the game will end at some point after a fixed number of moves. Unlike chess, where you can have an unlimited number of moves possible especially when you are left with the only king, but if you add an extra constraint that says “game should be ended within ‘n’ numbers of moves”, that will be a terminal condition. This is the kind of assumption a game theory is looking for. It turns out that there is a general strategy for such games, and we can analyze the games using the nim theory. Initially, we will analyze simple games where players remove sticks from heaps, and after this, we will generalize the strategy used in those games to other games. 1. Game states: Let us consider a game where there is initially a heap of n-sticks. Players A and B move alternately, and player A begins. On each move, the player has to remove 1, 2, or 3 sticks from the heap, and the player who removes the last stick wins the game. For example , if n = 10, the game may proceed as follows: A $\rightarrow$ removes 2 sticks (8 sticks left). B $\rightarrow$ removes 3 sticks (5 sticks left). A $\rightarrow$ removes 1 stick (4 sticks left) B $\rightarrow$ removes 2 sticks (2 sticks left). A $\rightarrow$ removes 2 sticks and wins This game consists of states 0, 1, 2,..., n, where the number of the state corresponds to the number of sticks left. A few examples of Game states are: Tic Tac Toe = Tic Tac Toe is a classic two-player game where the players take turns placing either X or O in a 3x3 grid until one player gets three in a row horizontally, vertically, or diagonally, or all spaces on the board are filled. Rock-Paper-Scissors = Rock-Paper-Scissors is a simple two-player game where each player simultaneously chooses one of three options (rock, paper, scissors). The winner is determined by a set of rules, rock beats scissors, scissors beat paper, and paper beats rock. 2. Winning and Losing states: A winning state is a state where the player will win the game if they play optimally, and a Losing state is a state where the player will lose the game if the opponent plays optimally. It turns out that we can classify all states of a game so that each state is either a winning state or a losing state Let's consider the above game: In the above game, state 0 is clearly a losing state because the player cannot make any moves. States 1, 2, and 3 are winning states because we can remove 1, 2, or 3 sticks and win the game. State 4, in turn, is a losing state, because any move leads to a state that is a winning state for the opponent. More generally, if there is a move that leads from the current state to a losing state, the current state is a winning state, and otherwise, the current state is a losing state. Using this observation, we can classify all states of a game starting with losing states where there are no possible moves. The states 0...15 of the above game can be classified as follows (W denotes a winning state and L denotes a losing state): States 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 Result L W W W L W W W L W W W L W W W It is easy to analyze this game: A state k is a losing state if k is divisible by 4, and otherwise, it is a winning state. An optimal way to play the game is to always choose a move after which the number of sticks in the heap is divisible by 4. Finally, there are no sticks left and the opponent

has lost. Of course, this strategy requires that the number of sticks is not divisible by 4 when it is our move. If it is, there is nothing we can do, and the opponent will win the game if they play optimally. Example :- Basketball = In basketball, a winning state is when a team scores more points than their opponent at the end of the game, while a losing state is when a team scores fewer points than their opponent. Chess = In chess, a winning state is when a player checkmates their opponent's king, while a losing state is when a player's king is checkmated. State graph : Let us now consider another stick game, where in each state k, it is allowed to remove any number x of sticks such that x is smaller than k and divides k. For example, in state 8 we may remove 1, 2 or 4 sticks, but in state 7 the only allowed move is to remove 1 stick. The following picture shows the states 1...9 of the game as a state graph, whose nodes are the states and edges are the moves between them: The states 1...9 of the game as a state graph, whose nodes are the states and edges are the moves between them The final state in this game is always state 1, which is a losing state because there are no valid moves. The classification of states 1...9 is as follows: 1 2 3 4 5 6 7 8 9 L W L W L W L W L Surprisingly, in this game, all even-numbered states are winning states, and all odd-numbered states are losing states 3. Nim game: The nim game is a simple game that has an important role in game theory because many other games can be played using the same strategy. First, we focus on nim, and then we generalize the strategy to other games. There are n heaps in nim, and each heap contains some number of sticks. The players move alternately, and on each turn, the player chooses a heap that still contains sticks and removes any number of sticks from it. The winner is the player who removes the last stick. The states in nim are of the form [x 1 , x 2 ,..., x n], where x k denotes the number of sticks in heap k. For example , A[] = [10,12,5] It is a game where there are three heaps with 10, 12 and 5 sticks. The state [0,0,...,0] is a losing state, because it is not possible to remove any sticks, and this is always the final state. Analysis: It turns out that we can easily classify any nim state by calculating the nim sum s = x1 ⊕ x2 ⊕⋯ ⊕ xn, where ⊕ is the xor operation. The states whose nim sum is 0 are losing states, and all other states are winning states. For example, the nim sum of [10,12,5] is 10⊕12⊕5 = 3, so the state is a winning state. Losing states: The final state [0,0,...,0] is a losing state, and its nim sum is 0, as expected. In other losing states, any move leads to a winning state, because when a single value x k changes, the nim sum also changes, so the nim sum is different from 0 after the move. Winning states: We can move to a losing state if there is any heap k for which x k ⊕ s < x k . In this case, we can remove sticks from heap k so that it will contain x k ⊕ s sticks, which will lead to a losing state. There is always such a heap, where x k has a one bit at the position of the leftmost one bit of s. As an example: consider the state [10,12,5]. This state is a winning state because its nim sum is 3. Thus, there has to be a move that leads to a losing state. Next, we will find out such a move. The nim sum of the state is as follows: 10 12 5 1010 1100 0101 3 0011 In this scenario, the heap with 10 sticks is the only heap that has a one bit at the position of the leftmost one bit of the nim sum: 10 12 5 10 1 0 1100 0101 3 00 1 1 The new size of the heap has to be 10⊕ 3 = 9 , so we will remove just one stick. After this, the state will be [9,12,5], which is a losing state: 9 12 5 1001 1100 0101 0 0000 4. Misère game: In a misère game, the goal of the game is the opposite, so the player who removes the last stick loses the game. It turns out that the misère nim game can be optimally played almost like the standard nim game. The idea is to first play the misère game like the standard game, but change the strategy at the end of the game. The new strategy will be introduced in a situation where each heap would contain at most one stick after the next move. In the standard game, we should choose a move after which there is an even number of heaps with one stick. However, in the misère game, we choose a move so that there is an odd number of heaps with one stick. This strategy works because a state where the strategy changes always appear in the game, and this state is a winning state because it contains exactly one heap that has more than one stick so the nim sum is not 0. 5. Sprague–Grundy theorem: The Sprague–Grundy theorem generalizes the strategy used in nim to all games that fulfil the following requirements: Two players move alternately. The game consists of states, and the possible moves in a state do not depend on whose turn it is. The game ends when a player cannot make a move. The game surely ends sooner or later. The players have complete information about the states and allowed moves, and there is no randomness in the game. For more detail you can refer to this article (Combinatorial Game Theory | (Sprague – Grundy Theorem)) The idea is to calculate for each game state a Grundy number that corresponds to the number of sticks in a nim heap. When we know the Grundy numbers of all states, we can play the game like the nim game. 6. Grundy numbers: The Grundy number of a game state is mex({g 1 , g 2 ,..., g n }). where g 1 , g 2 ,..., g n are the Grundy numbers of the states to which we can move, and the mex function gives the smallest non-negative number that is not in the set. For example: mex({0,1,3}) = 2. If there are no possible moves in a state, its Grundy number is 0, because mex(Ø) =

0.\ For example in the state graph: the state graph The Grundy numbers are as follows: the state graph for grundy numbers The Grundy number of a losing state is 0, and the Grundy number of a winning state is a positive number. The Grundy number of a state corresponds to the number of sticks in a nim heap. If the Grundy number is 0, we can only move to states whose Grundy numbers are positive, and if the Grundy number is x > 0 , we can move to states whose Grundy numbers include all numbers 0,1,..., x−1. As an example: Example: consider a game where the players move a figure in a maze. Each square in the maze is either a floor or a wall. On each turn, the player has to move the figure some number of steps left or up. The winner of the game is the player who makes the last move. The following picture shows a possible initial state of the game, where @ denotes the figure and # denotes a square where it can move. the possible initial state of the game The states of the game are all floor squares of the maze. In the above maze, the Grundy numbers are as follows: the Grundy numbers Thus, each state of the maze game corresponds to a heap in the nim game. For example, the Grundy number for the lower-right square is 2, so it is a winning state. We can reach a losing state and win the game by moving either four steps left or two steps up. Note that unlike in the original nim game, it may be possible to move to a state whose Grundy number is larger than the Grundy number of the current state. However, the opponent can always choose a move that cancels such a move, so it is not possible to escape from a losing state. 7. Subgames: Next, we will assume that our game consists of subgames, and on each turn, the player first chooses a subgame and then move into the subgame. The game ends when it is not possible to make any move in any subgame. In this case, the Grundy number of a game is the nim sum of the Grundy numbers of the subgames. The game can be played like a nim game by calculating all Grundy numbers for subgames and then their nim sum. As an example, consider a game that consists of three mazes. In this game, on each turn, the player chooses one of the mazes and then moves the figure in the maze. Assume that the initial state of the game is as follows: the player chooses one of the mazes and then moves the figure in the maze. Assume that the initial state of the game is as follows The Grundy numbers for the mazes are as follows The Grundy numbers for the mazes In the initial state, the nim sum of the Grundy numbers is 2⊕3⊕3 = 2, so the first player can win the game. One optimal move is to move two steps up in the first maze, which produces the nim sum 0⊕3⊕3 = 0. 8. Grundy's game: Sometimes a move in a game divides the game into subgames that are independent of each other. In this case, the Grundy number of the game is mex({g 1 , g 2 ,..., g n }), where n is the number of possible moves and g k = a k,1 ⊕ a k,2 ⊕...⊕ a k,m , where move k generates subgames with Grundy numbers a k,1 ,a k,2 ,...,a k,m . An example of such a game is Grundy's game. Initially, there is a single heap that contains n sticks. On each turn, the player chooses a heap and divides it into two nonempty heaps such that the heaps are of different size. The player who makes the last move wins the game. Let f (n) be the Grundy number of a heap that contains n sticks. The Grundy number can be calculated by going through all ways to divide the heap into two heaps. For example, when n = 8, the possibilities are 1+7, 2+6 and 3+5 , so f(8) = mex({f (1)⊕ f (7), f (2)⊕ f (6), f (3)⊕ f (5)}). In this game, the value of f (n) is based on the values of f (1),..., f (n−1) . The base cases are f (1) = f (2) = 0, because it is not possible to divide the heaps of 1 and 2 sticks. The first Grundy numbers are: f (1) = 0 f (2) = 0 f (3) = 1 f (4) = 0 f (5) = 2 f (6) = 1 f (7) = 0 f (8) = 2 The Grundy number for n = 8 is 2, so it is possible to win the game. The winning move is to create heaps 1+7 because f (1)⊕ f (7) = 0. Practice Problems on Game Theroy : Problem Find the winner in nim-game Game of N stones where each player can remove 1, 3 or 4 Choice of Area Finding optimal move in Tic-Tac-Toe using Minimax Algorithm in Game Theory Optimal Strategy for a Game Find the player who will win by choosing a number in range [1, K] with sum total N Optimal Strategy for the Divisor game using Dynamic Programming Chessboard Pawn-Pawn game Find the winner of the game with N piles of boxes Coin game of two corners Josephus Problem Variation in Nim Game Game of replacing array elements Combinatorial Game Theory Find winner in game of N balls, in which a player can remove any balls in range [A, B] in a single move Find the winner of a game of removing any number of stones from the least indexed non-empty pile from given N piles A Binary String Game Two player game in which a player can remove all occurrences of a number Make a palindromic string from given string Find the player who will win the Coin game Find the winner of the Game Optimal Strategy for a Game Find the winner of the Game to Win by erasing any two consecutive similar alphabets Find the winner of a game of removing at most 3 stones from a pile in each turn Pen Distribution Problem Find the winner of game of repeatedly removing the first character to empty given string Find the player who wins the game by removing the last of given N cards Winner in the Rock-Paper-Scissor game using Bit manipulation Optimal Strategy for a Game | Special Gold Coin Predict the winner of the game on the basis of absolute difference of sum by selecting numbers Find probability that a player wins when

probabilities of hitting the target are given Find winner when players remove multiples of A or B from Array in each turn Predict the winner of a card game of removing K cards in each turn such that Bitwise AND of K and size of pile is 0 Optimal strategy for a Game with modifications Number of ways for playing first move optimally in a NIM game Probability of A winning the match when individual probabilities of hitting the target given Predict the winner of the game | Sprague-Grundy Game of Chocolates | Wythoff's Game Minimum operations to reduce N to a prime number by subtracting with its highest divisor Find the winner of a game of donating i candies in every i-th move Largest odd divisor Game to check which player wins Combinatorial Game Theory | Set 1 (Introduction) Combinatorial Game Theory | Set 2 (Game of Nim) Combinatorial Game Theory | Set 3 (Grundy Numbers/Nimbers and Mex) Combinatorial Game Theory | Set 4 (Sprague – Grundy Theorem) Minimax Algorithm in Game Theory | Set 1 (Introduction) Minimax Algorithm in Game Theory | Set 2 (Introduction to Evaluation Function) Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI – Finding optimal move) Minimax Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning) Minimax Algorithm in Game Theory | Set 5 (Zobrist Hashing) Variation in Nim Game Find the winner in nim-game Game of Nim with removal of one stone allowed Check if the game is valid or not Game of replacing array elements Game of N stones where each player can remove 1, 3 or 4 The prisoner's dilemma in Game theory Choice of Area Implementation of Tic-Tac-Toe game Dynamic Programming | Set 31 (Optimal Strategy for a Game) Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above. Comment Article Tags: Article Tags: Competitive Programming