Chap 5

MinMax

The Minimax algorithm is a decision-making algorithm commonly used in game theory, particularly in two-player turn-based games such as chess, tic-tac-toe, and others. The algorithm's purpose is to determine the best move for a player, assuming that the other player will always make the best move possible.

The algorithm works by building a game tree, where each node represents a possible move and its corresponding state. The tree is then traversed to find the optimal move, which is the one that results in the highest score for the current player and the lowest score for the opponent.

The algorithm starts by evaluating the leaves of the game tree and assigning scores to each move. These scores are based on the current state of the game and the desired outcome for each player. The scores are then propagated up the tree, so that the score of each parent node is the maximum (for the current player) or minimum (for the opponent) of its children's scores.

The algorithm then selects the move with the highest score for the current player as the best move. This continues until the root node is reached, which represents the current state of the game.

In summary, the Minimax algorithm is a backtracking algorithm that evaluates all possible moves in a game, assuming that the other player will always choose the move that results in the worst outcome for the current player. The algorithm then selects the move that results in the best outcome for the current player.

Alpha beta pruning

Alpha-Beta pruning is a technique used in game tree search algorithms, such as Minimax, to optimize the search process by reducing the number of nodes that need to be explored. The basic idea behind Alpha-Beta pruning is to keep track of two values, called "alpha" and "beta", during the search process. These values represent the best possible value that the maximizing player (the player whose turn it is) and the minimizing player (the opponent) can guarantee, respectively.

At each node in the game tree, the algorithm calculates the score for that node and compares it to the current alpha and beta values. If the score is greater than beta, it

means that the minimizing player has found a better strategy and the maximizing player cannot guarantee a better score, so the search in that subtree can be stopped. On the other hand, if the score is less than alpha, it means that the maximizing player has found a better strategy and the minimizing player cannot guarantee a better score, so the search in that subtree can be stopped as well.

Alpha-Beta pruning helps to reduce the number of nodes that need to be explored by avoiding the exploration of subtrees that can be proven to be of no value. This greatly reduces the computational cost of the search process and allows for faster, more efficient algorithms.

Prisoner's Dilemma

The prisoner's dilemma is a classic problem in game theory that models a situation where two individuals must make a decision that affects both of them. The situation goes as follows: two individuals are suspects of a crime, and they are both held in separate cells, unable to communicate with each other. The prosecutor has enough evidence to convict both of them of a minor crime, but not enough evidence to convict them of a more serious crime.

The prosecutor offers each individual the following deal: if one of them confesses to the more serious crime and the other remains silent, the one who confesses will receive a reduced sentence, while the other will receive a harsher sentence. If both individuals confess, both will receive a harsher sentence than if they had both remained silent. If both individuals remain silent, both will receive a reduced sentence.

The prisoner's dilemma raises questions about cooperation, self-interest, and trust, and it is often used as a model for understanding social, political, and economic systems. In a real-world scenario, the dilemma is often repeated multiple times, allowing for the possibility of cooperation or defection to emerge over time.

Look ahead chess

In the context of chess programming, "ply" refers to the number of half-moves that a chess engine looks ahead in its evaluation of the game tree. For example, a 4-ply evaluation would consider the consequences of the current player's move, and then the consequences of the opponent's reply, and so on, to a depth of four half-moves.

The statement "4-ply ≈ human novice" means that a chess engine that evaluates the game tree to a depth of 4 plies is roughly equivalent in playing strength to a novice human player.

The statement "8-ply ≈ typical PC, human master" means that a chess engine that evaluates the game tree to a depth of 8 plies is roughly equivalent in playing strength to a typical personal computer and a human master player.

The statement "12-ply ≈ Deep Blue, Kasparov" refers to the famous match between Grandmaster Garry Kasparov and the supercomputer Deep Blue, which was programmed to evaluate the game tree to a depth of 12 plies. This level of analysis allowed Deep Blue to defeat Kasparov in 1997, becoming the first computer to beat a reigning world champion in a match.

More Info

I wrote this notes to be more in depth, if you find that something is not well organised remember it's just notes :p, or feel free to report it to me.