Data Structure Report

Game Tree

A game tree is a graph that maps out the moves and states of a game. The edges are moves and the nodes are the states. Game trees are used in artificial intelligence to allow the computer to make decisions. Each branch of the tree is a possible route the computer can take and most artificial intelligence is designed to evaluate which branch to use.

The game tree was a direct representation of John von Neumann’s minimax theorem developed in 1928. In 1950, Claude Shannon developed the minimax algorithm to apply von Neumann’s theorem. Later in 1951, Alan Turing designed a program on paper for playing chess that used decision trees. The first complete chess playing program could evaluate 350 positions per minute and was developed by Alex Bernstein in 1957. Ten years later, Richard Greenblatt made the first program that competed in a human chess tournament. It won three games, drew three games, and lost twelve games. Thirty years later in 1997, the supercomputer Deep Blue beat the world chess champion Gary Kasparov.

A minimax algorithm is a recursive algorithm for choosing the next move in an n-player game, usually two players. A value is associated with each node that represents the relative advantage or disadvantage for each player. Depending upon the goals of the artificial intelligence, it will use the branch that matches closest to the value desired.

A game tree is typically implemented in code as a normal node pointer tree structure. Each node will hold data about the game state that the node represents as well as operations showing which pointer relates to which next move. The complexity of searching a game tree is O(n). Since this complexity is not very efficient, pruning algorithms have been developed, such as Alpha-beta pruning, which reduce the number of nodes that are to be evaluated by the minimax algorithm. Sometimes this will still not be sufficient to allow the computer to evaluate all possible moves. Because of this, the programmer will often limit the program to only evaluate a certain depth of the tree or within a certain time. Lastly, instead of choosing the best possible move for itself sometimes a program will choose the move that leaves the worst move for the opponent out of his or her best possible moves. This requires less computational power while still achieving nearly the same result.

Game trees do not necessarily have advantages and disadvantages because there is no other data structure that can be used to represent the decision-making process. Game trees are used practically to represent all decision-making processes. As artificial intelligence becomes stronger, it will increasingly be used in our daily lives. We can safely assume that game trees will be used constantly by future generations.

<https://www.cs.umd.edu/users/nau/game-theory/4%20Game-tree%20search.pdf>



