**RUSHI MODI**

Use of Game Tree

Rutgers Newark University

Applied Math 2, Prof. **Mousumi Chakrabarty**

Due May9th, 2018

Game Tree

According to Wiki-Scratch, “Game tree is a type of recursive search function that examines all possible move of a strategy game and their results, in an attempt to ascertain the optimal move.” This Trees then can be used to analyze certain type of games such as tic-tac-toe, checkers, nim, chess and etc.

Here, we will classify three different groups of game where Game Tree is being used:

1. Single-player Pathfinding game problems:

Rubik’s Cube

Sliding puzzle

Traveling Salesman Problem

2. Multiplayer games:

Chess (my favorite)

Checkers

Othello

Nim

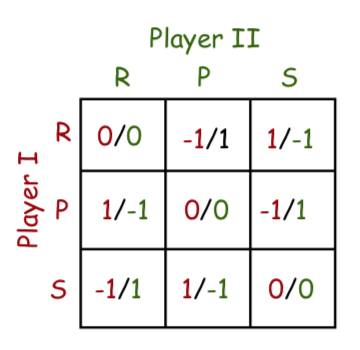
3. Puzzle games:

Sudoku

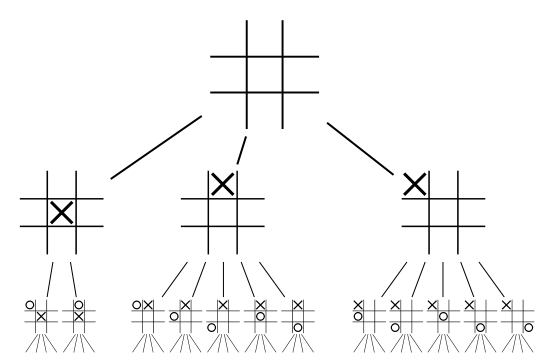
Eight Queens

Game Tree is used in Game theory. According to dictionary.com, Game theory is the branch of mathematics concerned with the analysis of strategies for dealing with competitive situations where the outcome of a participant’s choice of action depends critically on the actions of other participants. Think of Game theory as a science of strategy, and Game tree as a tool which helps to make an optimal decision making move in a strategy based games. Now we will discuss how game trees are used in games.

Game Tree is a directed graph whose vertices represent the positions that a game can be in as it progresses, in other words different positions in a game. Edges represent legal or valid moves between these positions. Depending on the game, Game trees can vary in sizes. The complete game tree for a game requires you to have a starting at an initial position, called root and leaves as representing final stage of either win, lose or draw. For example, Tic-tac-toe only has specific number of moves so therefore you can construct a complete game tree of it, but on the other hand it’s not the same for chess. Chess is played between two opponents on opposite sides of board with board contains 64 squares of alternating colors and each player 16 pieces either black or white colors, and white color pieces moves first. Now, we know the basics of chess, there is one thing that is different in chess then most of board games. In chess, after first move by both players, there are 400 possible board setups, after two moves, there are 197,742 possible board setups and after only three moves, there are over 121 million possible board setups, and chess game can last up to on average of 65 moves by both opponents up to possibly trillions of board setups, so you can imagine that complete game tree of chess is somewhat impossible with our current technology. Therefore, only partial game tree of chess exists. Even though, we seem far from reaching a complete game tree of chess, this might not be done sooner than we think with an invention of AI (Artificial Intelligence), just recently, Google let its AI powered by formidable, possibly the number one in world, play chess, and only in 4 hours it mastered all the chess knowledge and beaten the number one chess software Stock fish 8. Game Trees are very important in Artificial Intelligence, because for AI to decide on which move is the best in a game given in any position, it uses an algorithm minimax. According to dictionary.com, Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case scenario. This algorithm was originally used in zero-sum games (such as checkers, chess, and tic-tac-toe). Zero-sum means the sum of your and your opponent's payoff is zero, in other words, your gain come at your open nonet’s loss and vice versa. Zero-sum games are competitive, meaning if one player wins then other loses. Rock, Paper and Scissors is a good example of a zero-sum game. Rules of the game are, scissors cut paper, paper covers rock and rock smashes scissors. I have represented in below diagram how payoff to each player in each cell works. For representation, I have assigned, 1: win, 0: tie, -1: loss. So for example, when player 1 chooses paper and player 2 chooses rock, then player 1 wins by rules, and as seen in table, player 1 is indeed 1 and player 2 is -1.

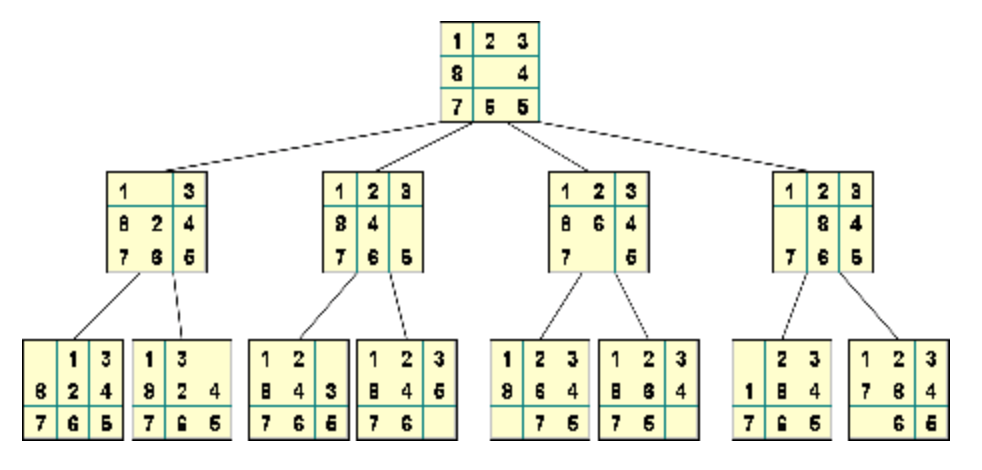


Now, we will discuss on the representation of game tree of different games. One of the most common and easy games that we have all played in our childhood is Tic-Tac-Toe. So, remember that nodes or vertices represent “positions” and edges represent “legal moves”.



This Diagram shows first two moves or levels in the game tree of tic-tac-toe. Even though there are 9 possible squares for first move, you can always rotate the board to get in to one of this three positions after the first move. As you can see in the figure, first player has three choices of moves, 1. In center, 2. On left corner, 3. In middle of row one, if all other moves are played the rotate to get into one of this 3 position as we discussed earlier. Then the second player has two choices if the first player played move 1, and if first player played move 2, then second player has 5 choices and so on. Tic-tac-toe does indeed have a complete game tree with 255,168 leaf nodes. Number of leaf nodes in a complete game tree are associated with the number of different possible ways that the game can be ended in which is what leaf nodes represent.

Now, we will talk about use of game tree in 8-puzzle game. In there, nodes represent different arrangement of tiles, and edges represents “moving the tile up, down, right or left. In bottom, I have attached a pic of the representation of the 8-puzzle game with use of game tree. Each game consists of a problem space (a mathematical representation in form of a tree), an initial state and goal state. Initial state consists of the arrangement of the tiles that you start with or are given with and in goal state you want to achieve the tiles ranking from 1-8 in order. In bottom pic, you can see that your initial state is the top tile arrangement since that is what you are given with and you need to solve it. In the second frame, you can see you are given with 4 permutations of the tiles because you can only have 4 possible moves, and in third frame of the pic, you have 8 permutations of tiles. Often a problem space has two factors: a branching factor and a solution depth.



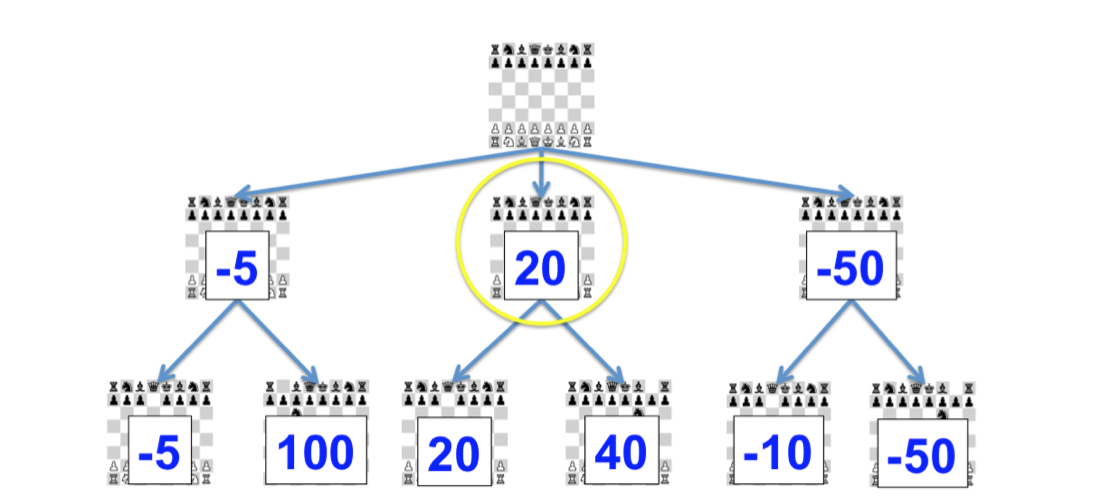
Branching factor is the total average number of children generated by any given node, and solution depth is the total length of the shortest path from a root node to goal node. Just to give you a game perspective, 8 puzzle game has branching factor of 2.13 and Rubik’s cube branching factor is 13.34, and chess has branching factor of 35, remember that earlier I mentioned that chess is partial tree due to the overwhelming amount of possible moves there are in chess. That is the reason why chess has a high branching factor and therefore only partial tree exists for chess. Let’s talk about the size of a solution space of a game. This is different from solution depth. Solution space is set of all possible positions in a game. For a tic tac toe is 9! = 362, 880, 8-puzzle is 9! /2 = 181,440, checkers is 10^ (40), chess s 10^ (120).

You might wonder, how would someone search a particular position or a move in game tree of a game. There are three ways, one can search with in a game tree:

1. Breadth-First Search (BFS)
2. Depth-First Search (DFS)
3. Depth-First Iterative Deepening

This are known as brute force algorithm to search in a game tree. All of this three algorithm tare mostly used to search for nodes in tree or graph. BFS algorithm starts at either at root node and then it explores its neighbor node before moving to next level, so BFS is used to find a connected components of nodes in graph. We also use BFS algorithm to find the shortest path from root node to goal node. Next, Depth - First Search (DFS) is an algorithm that uses backtracking, so it starts at root and then you explore all the branches before you backtrack.

Let’s discuss about minimax. In two players’ games, we use BFS to to find all the game states that is obtained from the root state to certain amount of moves. Also, with some assessment function, you can also see how good is your move is in respect to win/loss. Since, we can figure out all the moves that we can possibly make suppose in a game chess, then our next question rises are how can we tell if that move is good or bad in respect to your position. Also, keep in mind that your opponent is also trying to find a best move. Here, we can use minimax algorithm which follows as in this below figure and steps underneath.



1. Suppose you want to find best moves for x moves in game of chess then use BFS to find all moves up to x moves. Game state tree that reaches in 10 moves will be on 10th level of the tree of root s.

2. start at (x-1)th level and moving up a level each time until you get to S(root).

a. In your turn, you assign a value of each state of a game to be a maximum of their children state, therefore your best move is maximum state value.

b. If it’s your opponent’s turn, assign the value of each game state to be a minimum of your previous or children states, therefore your opponent’s turn is assigned minimize state.

By End of this algorithm, you will have game state values of each of game states of Root S, and it will factor out the best move so you have the edge in the game. This how Minimax algorithm is being used with use of BFS.

In conclusion,

I never knew this is how chess software finds the best move in given position of the game, and this has been a learning experience for me. Now, I know how Algorithms such as Minimax and BFS are used in game like chess to find out the best possible move to play against your opponent. Game tree is a very interesting concept of game theory, and you don’t know about this unless and until you read about use of trees in daily routines of one’s life.

References

*Discrete mathematis and is applications 6th edition page 704*. (n.d.). Game Trees. (n.d.). Retrieved from

<https://www.investopedia.com/terms/g/gametheory.asphttps://en.wikipedia.org/wiki/Game_tree>

Staff, I. (2016, April 16). Game Theory. Retrieved from <https://www.investopedia.com/terms/g/gametheory.asp>

Game Trees. (n.d.). Retrieved from [https://www.cs.cmu.edu/~adamchik/15-121/lectures/Game Trees/Game Trees.html](https://www.cs.cmu.edu/~adamchik/15-121/lectures/Game%20Trees/Game%20Trees.html)

Game Trees. (n.d.). Retrieved from <https://courses.csail.mit.edu/6.006/spring11/rec/rec12.pdf>