Game

Value of the game : The game can take two values +1 and -1. +1 indicates that the MAX player wins and -1 indicates that MIN player wins.

Action: Each player can remove 1 or 2 or 3 matches. The actions takes present state as input to give the list of valid actions(valid_actions)

State: State keeps track of following:

- **Matches**: The number of sticks left.
- **Parent**: To differentiate different states reached through different paths, parents of each state is also kept track of.
- **Utility**: Also we store the utility of each node in its state.
- **Move**: The move that was taken to reach this state
- **Level**: The level of the node in the game tree. This indicates the number of times the players have taken out sticks
- **Children**: All states that can be reached from the present one.

Result: Takes in present state and a action to give a new state.

Terminal State : When there are no more sticks to move the game reaches terminal state. That is, state.matches = 0

Utility: The utility of terminal node is +1 when it terminates with MAX player and -1 when it terminates with MIN player

Minimax:

Minimax algorithm computes recursively to find the next move. Utility value is associated with each state. The MAX player tries to maximize the utility value, whereas a MIN player tries to minimize the utility of the MAX.

Alpha Beta Pruning:

This is same as minimax algorithm except for that it prunes away all the parts of the tree which cannot affect the final decision

Visualization:

Using pygraphviz, which is Python interface to the Graphviz graph layout and visualization package the game trees are drawn. The function to draw game tree is in gameGraph.py. Since the graphs are huge only first few levels are displayed. The graphs are in the folder 'graphs'

Analysis :

Relative Sizes

N	# of nodes using Minimax	# of nodes using Alpha Beta	Ratio
7	95	78	1.22
15	12639	4111	3.07
21	489395	71360	6.86

As we can see from the table, Alpha Beta Search proves to be much more efficient. As the size of n grows, Alpha Beta Search performs much better than Minimax. For n = 21, the number of nodes in Minimax is almost 7 times the Alpha Beta Search.

First Player and Second Player

Value of the Game

n	Value
7	1
15	1
21	-1

Opting to go first and second will make a difference in the outcome of the game. As we can see from the table above, if one chooses to play first for cases n=7 and n=15, it would be beneficial whereas for case n=21 opting to be second player is beneficial. The optimal player choosing to go first for n = 7 and n = 15 will always win and for n = 21 may lose.

From analysis of the graphs for different n, it was observed that value of game is -1 for all 4i+1, where i is a natural number.

Optimal Vs Random Player

For case n = 7 and n = 15: For these two cases, a player who goes first will definitely win.

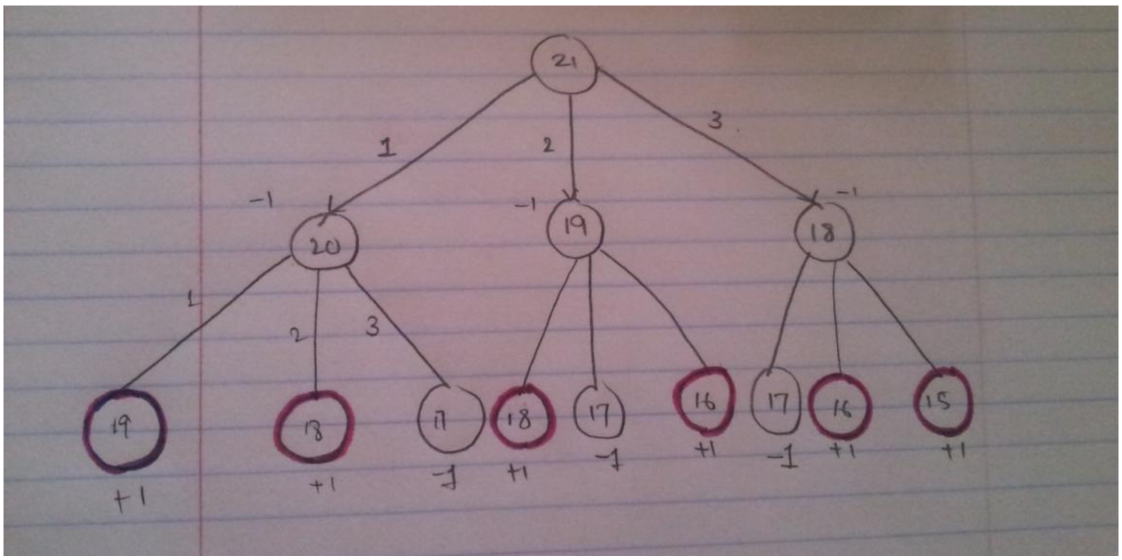
Optimal player goes first: It would win.

Random player goes first: Probability of this player choosing action 1 or 2 or 3 is same.

The utility value of children of the initialState is {-1, +1 and -1} for both n =7 and n =15 . The random player would have taken the right decision if it had chosen to go for the child with utility +1. But the probability of this being chosen is only 1/3 and hence the probability of random player winning is 1/3 and optimal player winning is **2/3**

For case n = 21:

Optimal player goes first: The utility value of all children in this case is {-1, -1, -1}. The favorable nodes for optimal player after random player's turn is colored red in the following image. The probability of random player choosing one of the favorable conditions for optimal player is 2/3. Hence probability of optimal player winning is **2/3**

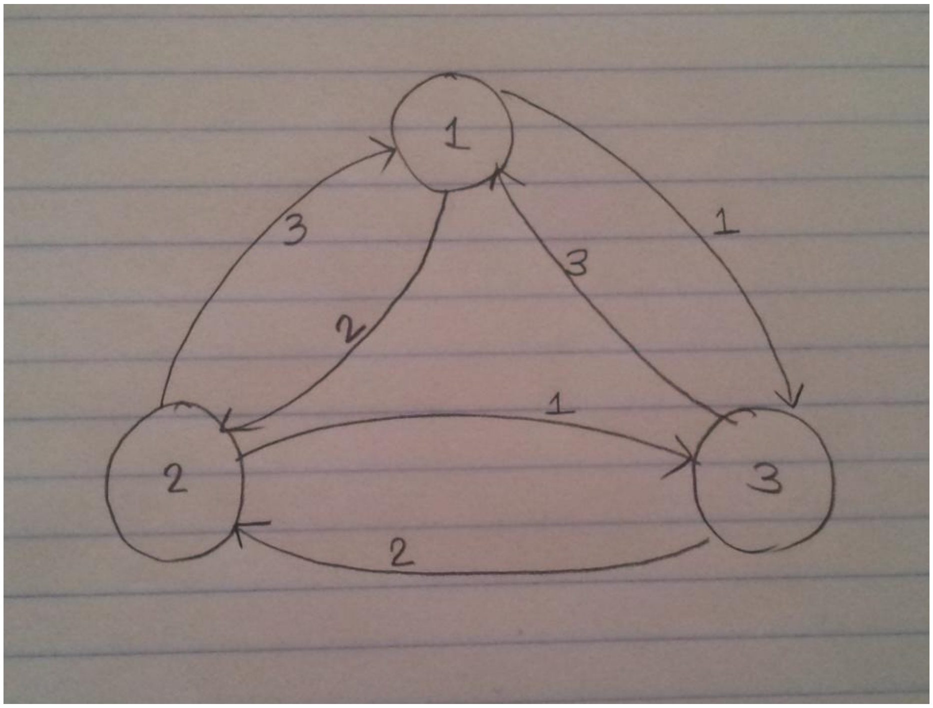


Random player goes first: Optimal player will definitely will, because for n = 21, the second player has an upper hand as already discussed

Moore Machines

The Moore machines for all cases of n is in the folder 'graphs'.

The Moore machines can be compressed. For all cases of n ≠ 4i + 1, where i is a natural number will look like this :


1
2
3
3
2
1
3
1
2