Description of MiniMax Algorithm

The minimax algorithm computes the minimax decision from the current state. It uses a simple recursive computation of the minimax values of each successor state, directly implementing the defining equations. The recursion proceeds all the way down to the leaves of the tree, and then the minimax values are backed up through the tree as the recursion unwinds. For further understanding of minimax algorithm have a look at the definition - makeTreeMinimax, in the main.py in the zip file.

Description of Alpha-Beta Pruning Algorithm

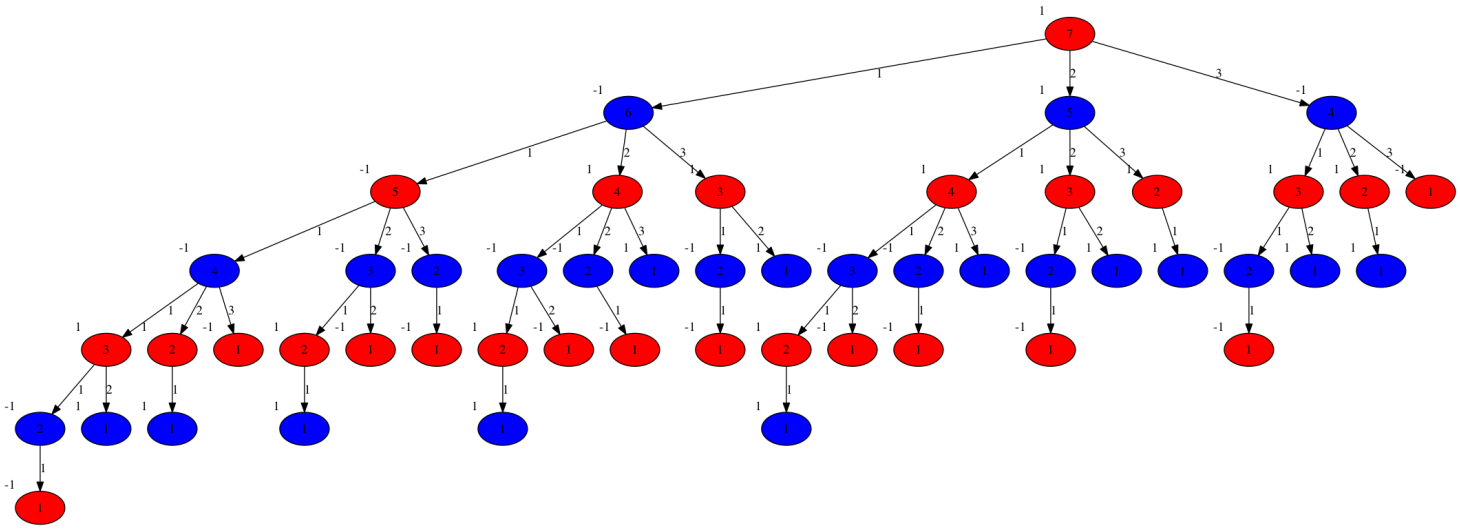
Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision. For further understanding of alpha-beta pruning algorithm have a look at the definition - makeTreeAlphBeta, in the main.py in the zip file.

Visualization of Trees Produced

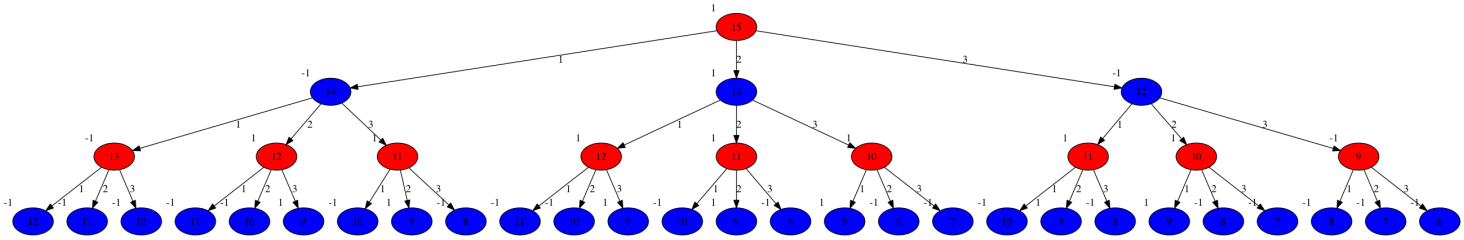
Red Node: Max player's Turn
Blue Node: Min Player's Turn

1) Using MiniMax algorithm

    a) 7 sticks

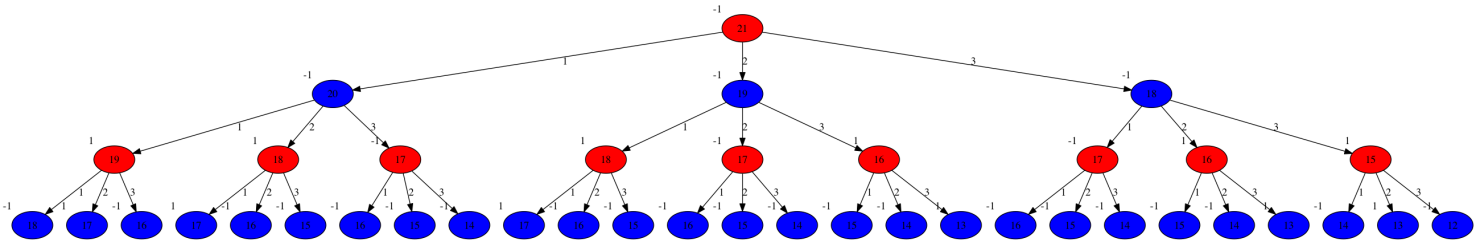


b) 15 sticks



I have also included game15-minimax.png in the zip file. When you open the png file, it will appear as a thin strip but when you zoom in into the picture, clear view of the tree can be viewed. (The view is limited to 200 nodes)

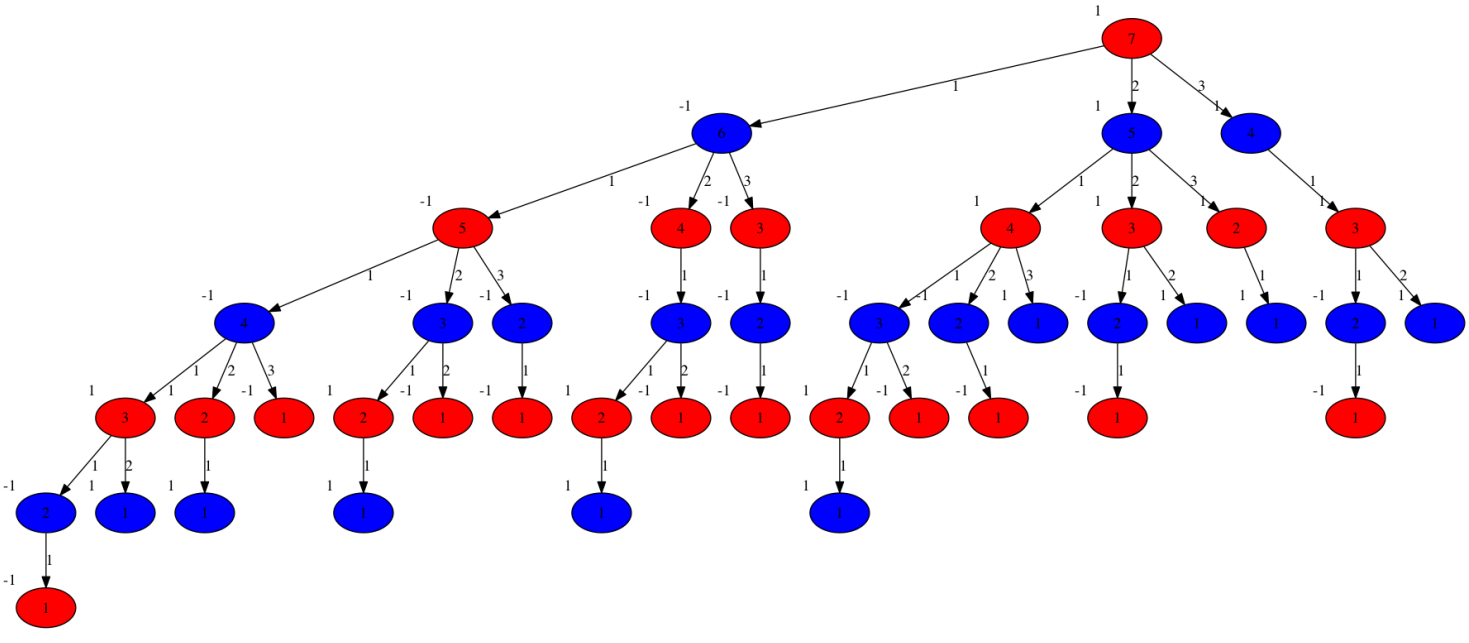
c) 21 sticks



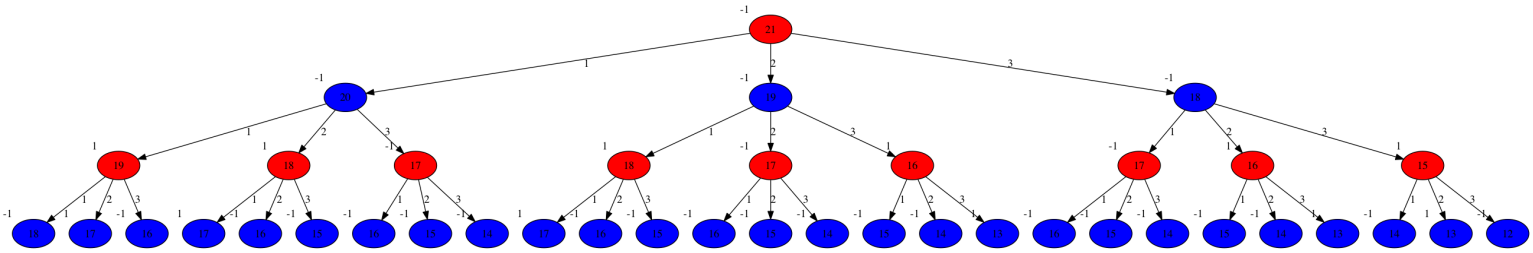
I have also included game21-minimax.png in the zip file. When you open the png file, it will appear as a thin strip but when you zoom in into the picture, clear view of the tree can be viewed. (The view is limited to 200 nodes)

2) Using Alpha-Beta Pruning Algorithm

a) 7 sticks

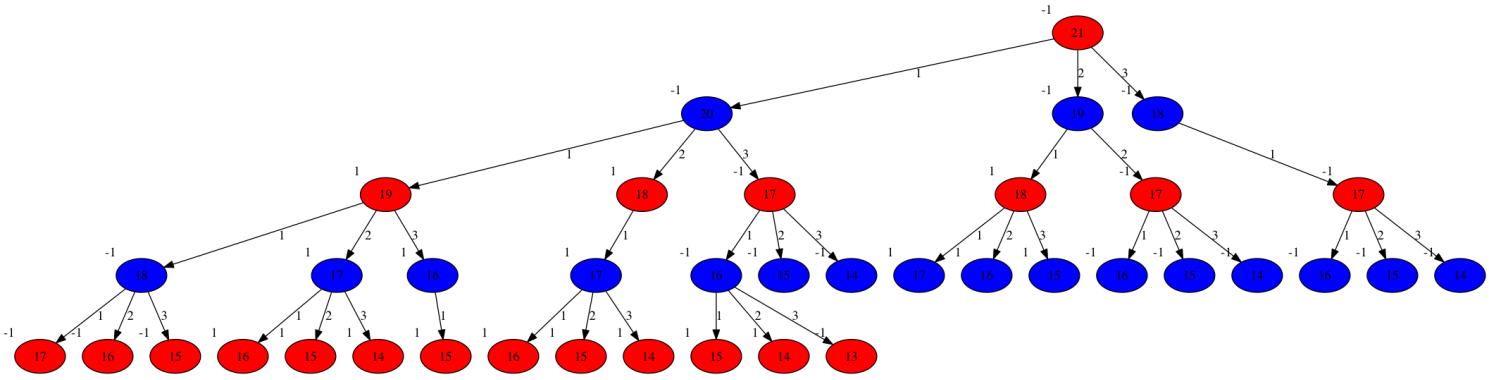


b) 15 sticks



I have also included game15-alpha-beta.png in the zip file. When you open the png file, it will appear as a thin strip but when you zoom in into the picture, clear view of the tree can be viewed. (The view is limited to 200 nodes)

c) 21 sticks



I have also included game21-alpha-beta.png in the zip file. When you open the png file, it will appear as a thin strip but when you zoom in into the picture, clear view of the tree can be viewed. (The view is limited to 200 nodes)

<u>Analysis of games by addressing the questions in 2a-2c</u>

2a) Relative Sizes of Trees

1) 7 sticks

minimax: 52 nodes
alpha-beta: 45 nodes

2) 15 sticks

minimax: 6872 nodes
alpha-beta: 2205 nodes

3) 21 sticks

minimax: 266079 nodes
alpha-beta: 45872 nodes

2b) Does opting to go first or second play a role in the outcome of the game? What is the value of the game?

1) 7 sticks

Yes, machine opting to go first always wins while machine opting to go second has one way where it loses and two ways where it wins.

The value of the game is +1 if M starts and -1 if H starts (M-Max, H-Min).

2) 15 sticks

Yes, machine opting to go first always wins while machine opting to go second has one way where it loses and two ways where it wins.

The value of the game is +1 if M starts and -1 if H starts (M-Max, H-Min).

3) 21 sticks

No, machine opting to go first or second loses.

The value of the game is -1 for whichever player starts.

2c) How well will the optimal player fair in a play-off against a random player?

Game was played a 1000 times and analysis is made on that.

1) 7 sticks
- Optimal Player playing first : 100% wins (1000/1000 games won by optimal player)
- Optimal Player playing second : 100% wins (1000/1000 games won by optimal player)

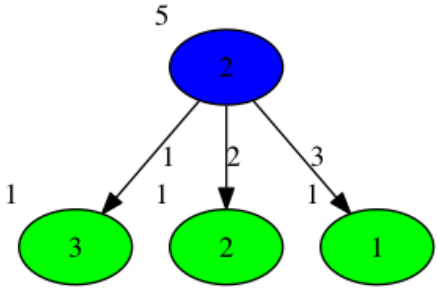
2) 15 sticks
- Optimal Player playing first : 100% wins (1000/1000 games won by optimal player)
- Optimal Player playing second : 100% wins (110/1000 games won by optimal player)

3) 21 sticks
- Optimal Player playing first : 0% wins (0/1000 games won by optimal player)
- Optimal Player playing second : 0% wins (110/1000 games won by optimal player)

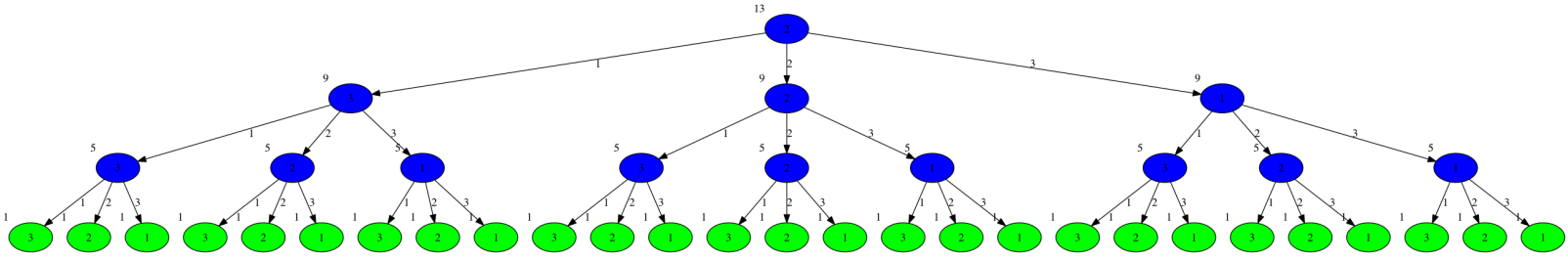
Moore Machine

Green Nodes represent final state
Blue are Max player states
Edges means Min player's decision
The value inside the node means Max player's decision
The value on top of nodes is the number of sticks left on desk
Max player plays first

a) 7 sticks



b) 15 sticks



c) 21 sticks

My code is running into errors for making the moore machine for 21 sticks.