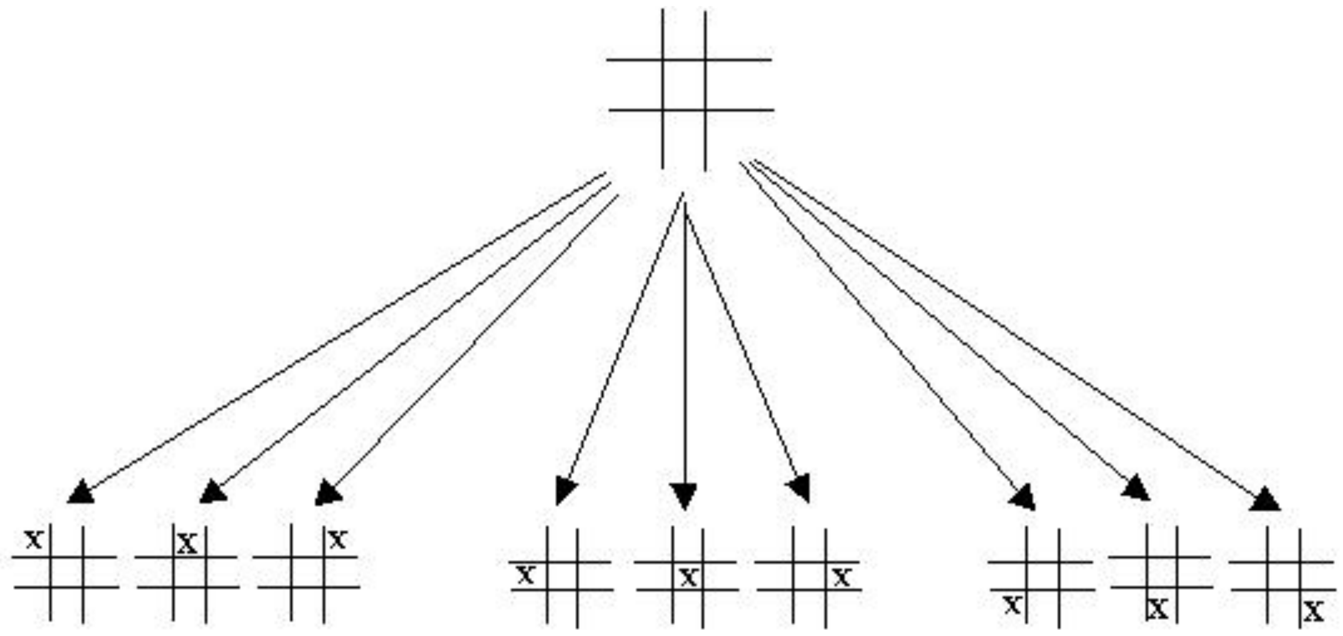
Game Trees

COP4530 – Week 5 Recitation

Game Tree

- A directed graph
  - Nodes are board-states
  - Edges are moves

Tic-tac-toe (3x3)

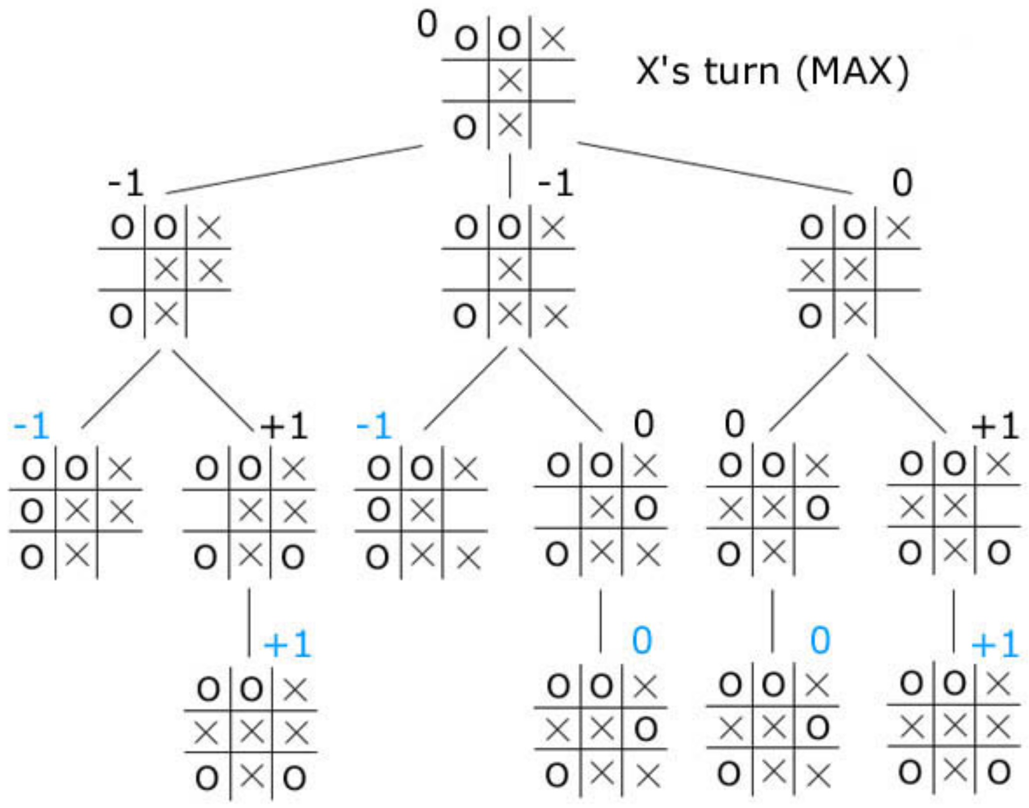


http://www.cs.nott.ac.uk/~ajp/courses/g51iai/003blindsearches/implement.htm

Minimax

- Assumptions
  - The opponent and our program will always make rational moves
  - Zero-sum game, without cooperation
- Assign weighted values to board positions
  - Positive, favorable to the program
  - Negative, favorable to the adversary

Minimax - traversal



http://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html

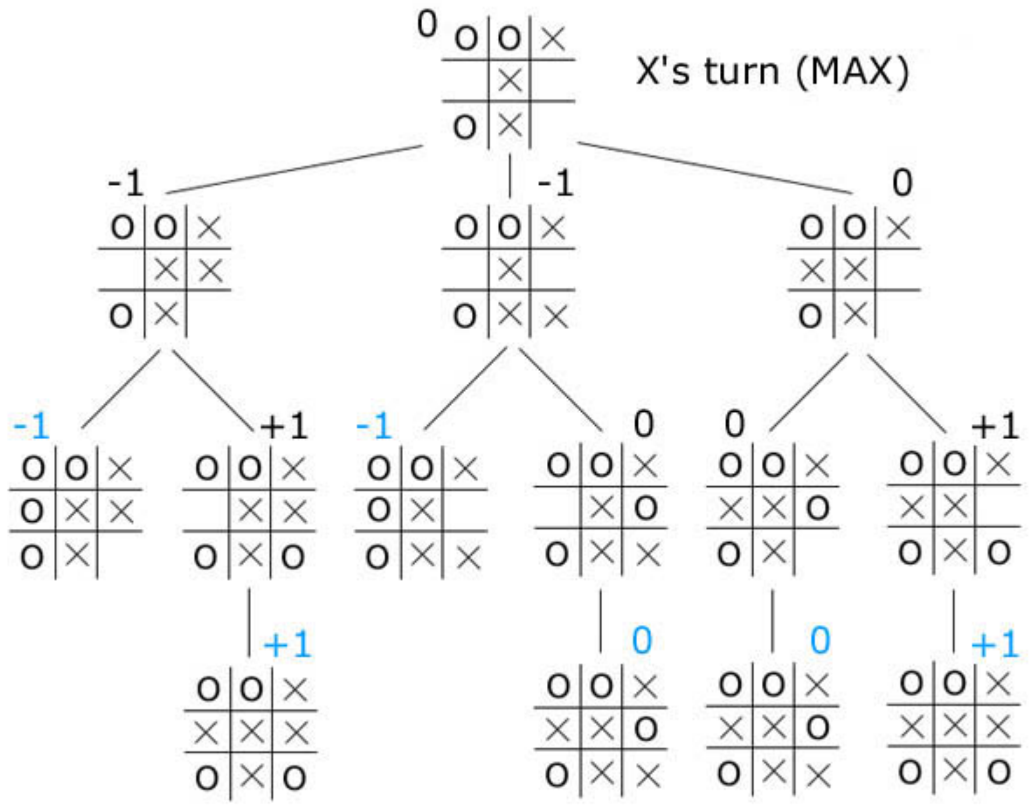
Minimax - pseudocode

```
minimax(player,board)
    if(game over in current board position)
        return winner
    children = all legal moves for player from this board
    if(max's turn)
        return maximal score of calling minimax on all the children
    else (min's turn)
        return minimal score of calling minimax on all the children
```

Negamax

- Variation on Minimax
- Max(a,b) = -Min(-a, -b)

Negamax - traversal



http://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html

Solution Space

- It can take a long time to evaluate an entire game tree
  - Too difficult for games such as Go or Chess
- The solution space for a 3x3 tic-tac-toe board is 9!
  - Larger for board variations

Alpha-Beta Pruning

- Alpha, the program’s best move
- Beta, the adversary’s best move
- If Alpha >= Beta, stop evaluation of current branch and move on
- Alpha = -∞, Beta = ∞

Alpha-Beta Pruning - example

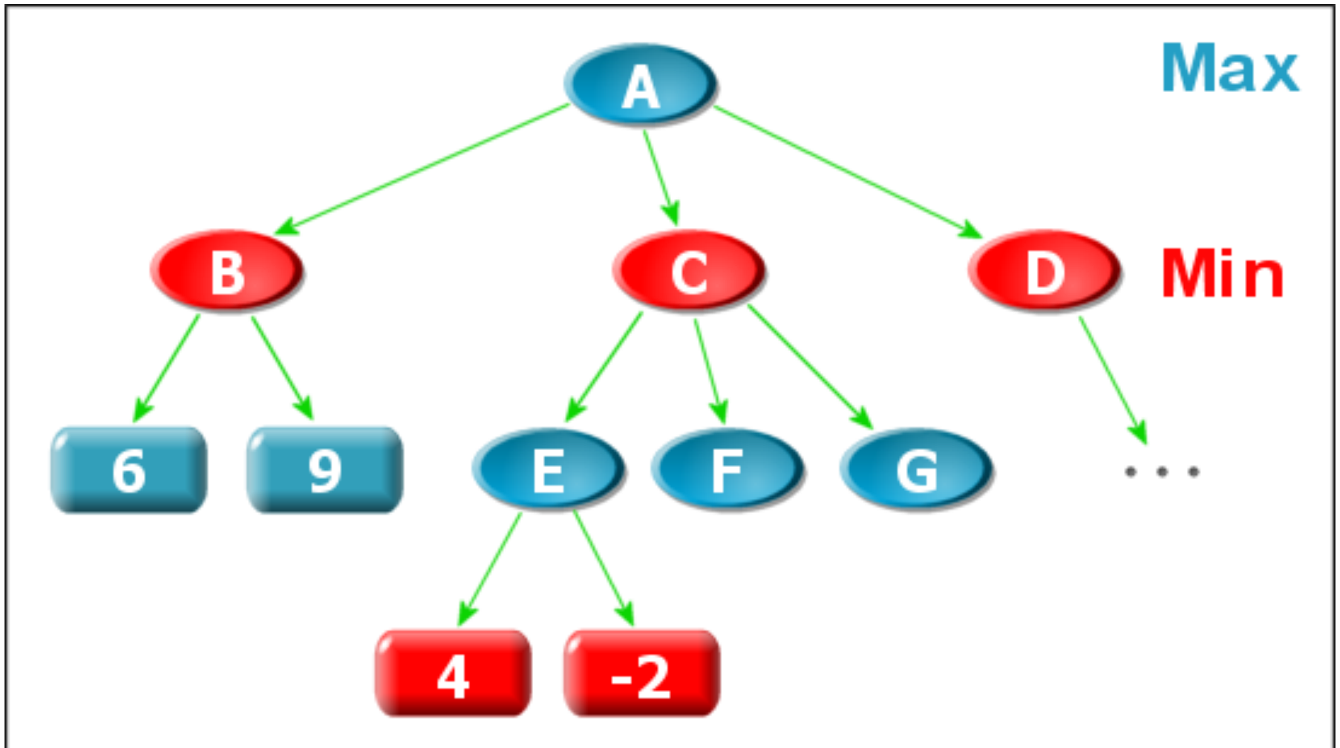


Figure 4 - Alpha-Beta pruning

http://www.hamedahmadi.com/gametree/

Alpha-Beta Pruning - pseudocode

```
alpha-beta(player,board,alpha,beta)
    if(game over in current board position)
        return winner

    children = all legal moves for player from this board
    if(max's turn)
        for each child
            score = alpha-beta(other player,child,alpha,beta)
            if score > alpha then alpha = score (we have found a better best move)
            if alpha >= beta then return alpha (cut off)
        return alpha (this is our best move)
    else (min's turn)
        for each child
            score = alpha-beta(other player,child,alpha,beta)
            if score < beta then beta = score (opponent has found a better worse move)
            if alpha >= beta then return beta (cut off)
        return beta (this is the opponent's best move)
```

More optimizations

- Look at only a few levels of depth
- Eliminate symmetric game boards
- Heuristic search

References

- http://www.cs.nott.ac.uk/~ajp/courses/g51iai/003blindsearches/implement.htm
- http://www.hamedahmadi.com/gametree/
- http://www.cs.cmu.edu/~adamchik/15-121/lectures/Game%20Trees/Game%20Trees.html
- http://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html
- http://web.mit.edu/sp.268/www/gamesearch.pdf