Game Playing

林伯慎 Bor-shen Lin
bslin@cs.ntust.edu.tw
http:www.cs.ntust.edu.tw/~bslin

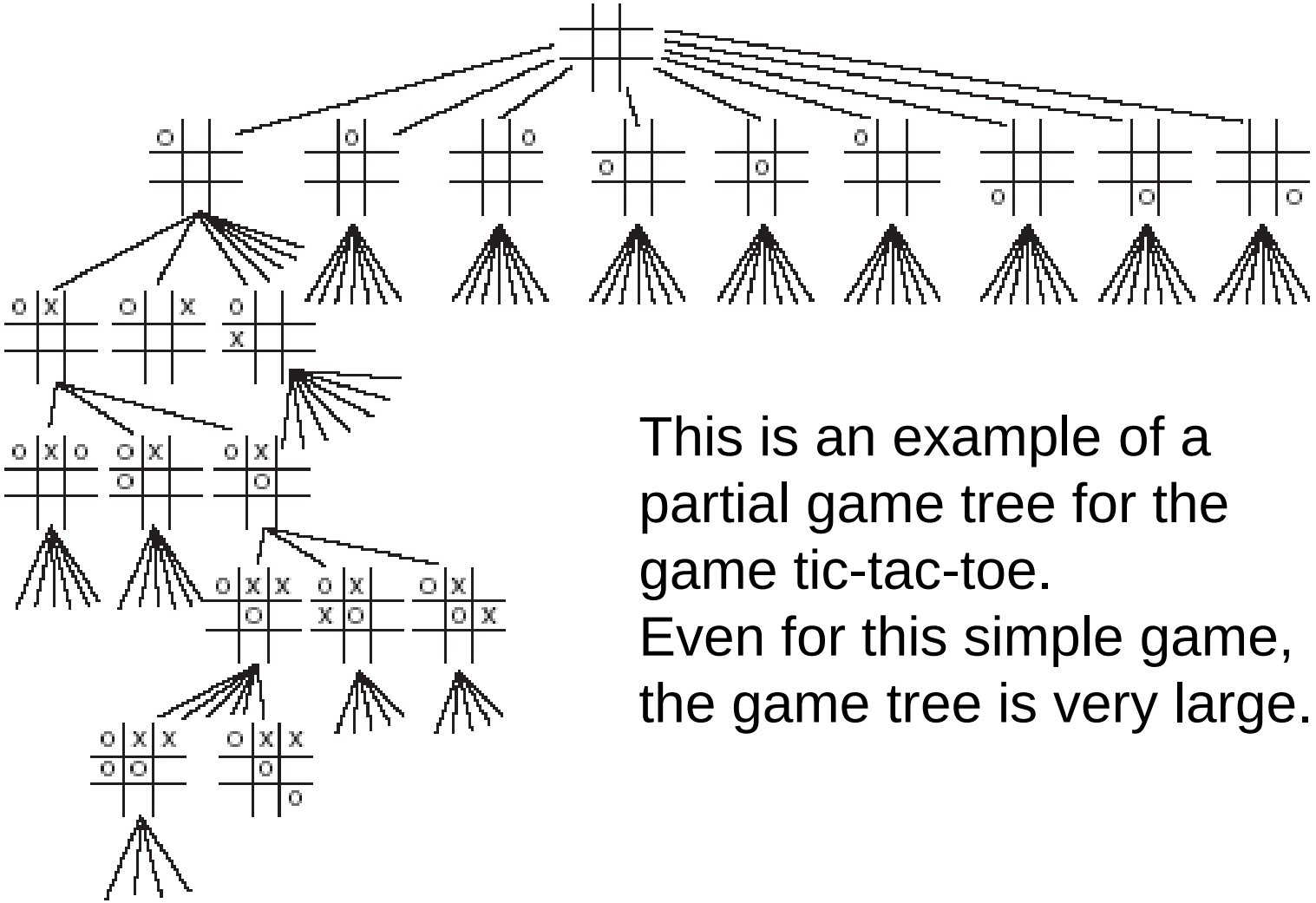
Agenda

- Game Trees
- Assumptions
- Static evaluation functions
- Searching game trees
- Minimax
- Bounded lookahead
- Alpha-beta pruning
- Checkers

Game Trees

- Game trees are used to represent two-player games.
- Alternate moves in the game are represented by alternate levels in the tree (plies).
- Nodes in the tree represent positions.
- Edges between nodes represent moves.
- Leaf nodes represent won, lost or drawn positions.
- For most games, the game tree can be enormous.

Game Trees

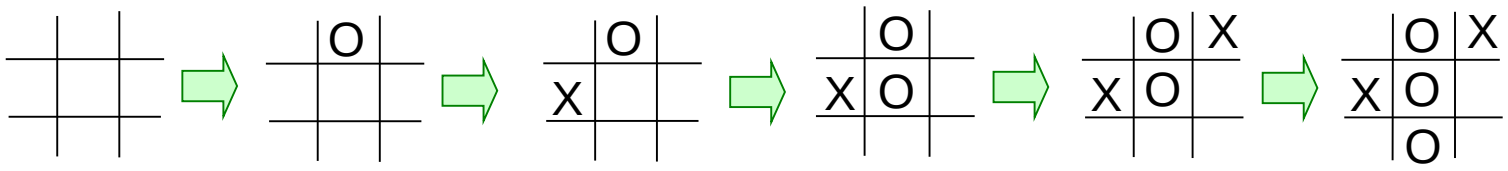


This is an example of a partial game tree for the game tic-tac-toe.
Even for this simple game, the game tree is very large.

Assumptions

- The opponent is rational & adversarial – will play to win.
- The game is zero-sum – if one player wins, the other loses.
- Usually, the two players have complete knowledge of the game. For games such as poker, this is clearly not true due to element of chance.

Non-Adversarial Opponent



- Play golf with the boss or customers
- Play cards or chesses with children

Static Evaluation Functions

- Since game trees are too large to be fully searched, it is important to have a function to statically evaluate a given position in the game.
- Are usually weighted linear function
  - Can be extremely complex
- A static evaluator assigns a score to a position:
  - High positive = computer is winning
  - Zero = even game
  - High negative = opponent is winning

Evaluation Function

- A static evaluator
  - give a better score to a better position – the actual values are not so important.
- How ?
  - could be assigned manually based on expert knowledge
  - could be learned according to the result of the game.
- Chess
  - q = number of queens
    r = number of rooks
    n = number of knights
    b = number of bishops
    p = number of pawns
    score = 9q + 5r + 3b + 3n + p

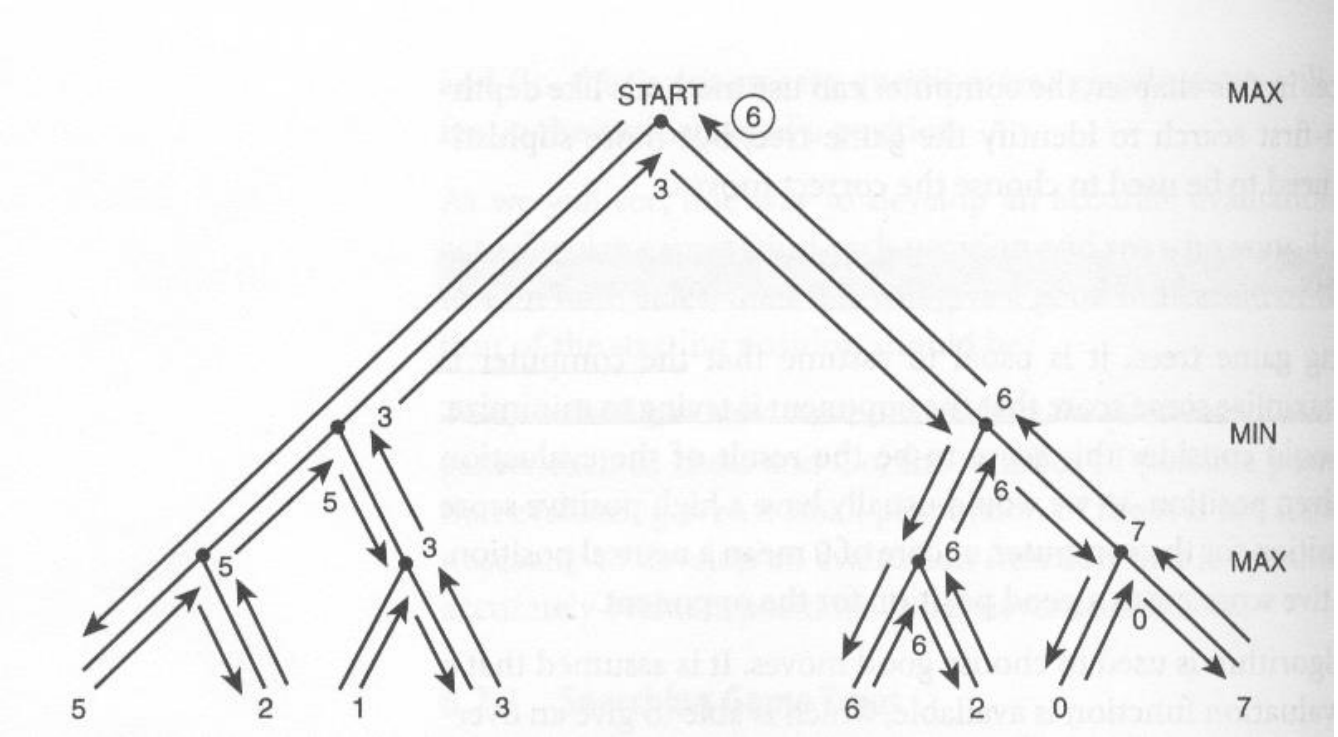
Search of Game Trees

- The concept of ply: the depth of search tree
  - Look ahead more plies → higher depth → higher complexity
- Huge search space
  - Branching factor is usually high
  - Exhaustively searching a game tree is not a good idea.
  - Even for a game as simple as tic-tac-toe there are over 350,000 nodes ($9 \cdot 8 \cdot \ldots \cdot 1$) in the complete game tree .
- Difference of state space search between *game playing* and *problem solving*
  - Computer only gets to choose every other move through the tree – the opponent chooses the others.

Minimax Algorithm

- Minimax is used to evaluate game trees.
  - Already applied in risk management of investment
  - MAX: Computer's turns to maximize the score
  - MIN: Opponent's turns to minimize the score
  - The opponent is not cooperative to make the computer win, but adversarial to defeat the computer
- A static evaluator
  - is applied to leaf nodes, and values are passed back up the tree to determine the best score the computer can obtain against a rational opponent.
- The game tree is searched in depth-first way.
  - With a predetermined depth

Example of Minimax



Minimax Algorithm

```
Function minimax(current_node)
 {
        if is_leaf(current_node)
                then return static_evaluation(current_node);
        if is_min_node(current_node)
                then return min(minimax(children(current_node)));
        if is_max_node(current_node)
                then return max(minimax(children(current_node)));
}
```

Bounded Lookahead

- For trees with high depth or very high branching factor, minimax cannot be applied to the entire tree.
- In such cases, bounded lookahead is applied
  - When search reaches a specified depth, the search is cut off, and the static evaluator applied.

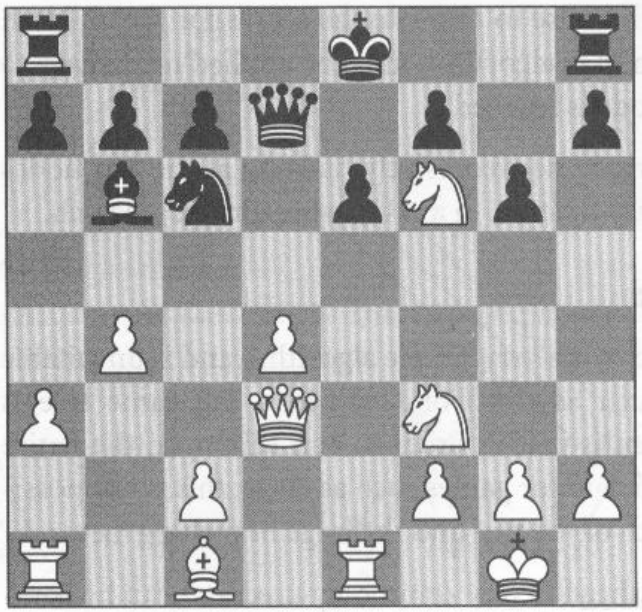
    (Not search to leaf nodes)
- Corresponds to human's way
  - The masters consider only important states instead of checking all possible states exhaustively

Bounded Minimax Algorithm

```
Function minimax(current_node)
 {
        if is_leaf(current_node)
                then return static_evaluation(current_node);
        if depth_of(current_node) == max_depth
                then return static_evaluation(current_node);
        if is_min_node(current_node)
                then return min(minimax(children(current_node)));
        if is_max_node(current_node)
                then return max(minimax(children(current_node)));
}
```

Cut-off At Fixed Depth

- Not good to cut-off sometimes
  - Search deeper states
- Solution
  - Not fixed depth, but till quiescent positions
  - Quiescent position: the next move is unlikely to cause a large change in the relative positions of the two players.



Alpha-beta Pruning

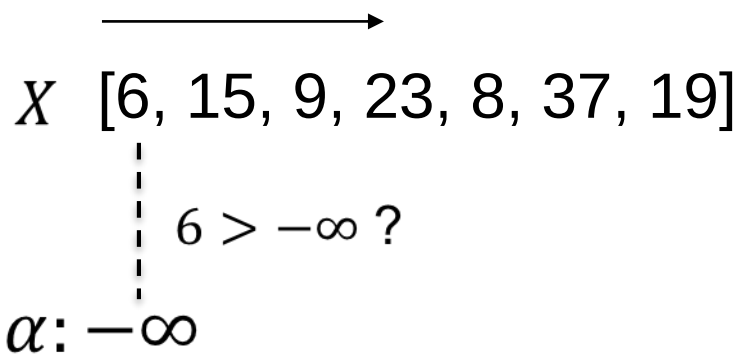
- A method that can often cut off a large part of the game tree.
- Based on the idea that if a move is clearly bad, there is no need to waste time to search the consequences of it completely.

Process of Finding Maximum

```
alpha = -10000000; // very small value
argmax = 0;
for(int i = 0; i < X.length; i++) {
        if(X[i] > alpha) { // replace alpha if larger
                alpha = X[i];  // replaced by X[i]
                argmax = i; // storing its index
        }
}
```

- *alpha* is the *temporary maximum* during the execution.
- *alpha* can *increase only* (could be replaced by a larger entry) during the the maximization process.
- *alpha* is the maximum *finally* after all elements are visited

EXAMPLE



- Value of α is updated when larger values are tested (compared)
- Status of α
  - $-\infty$, 6, 15, 23, 37
  - keep increasing
- α is the temporary maximum

Finding Minimum in An Array

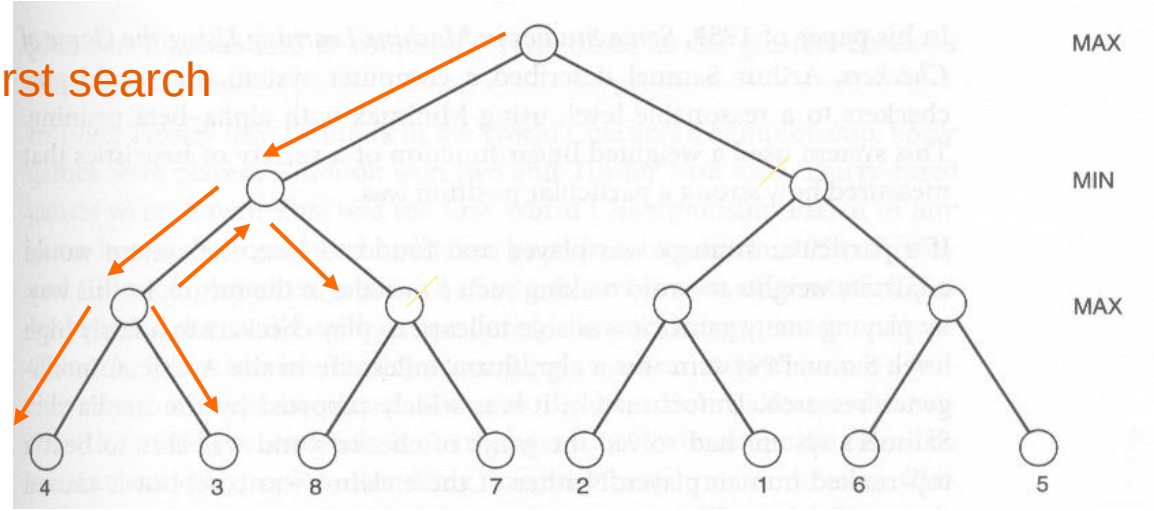
```
beta = 10000000; // very large value
argmin = 0;
for(int i = 0; i < X.length; i++) {
        if(X[i] < beta) { // replace it if smaller than beta
                beta = X[i];  // replaced by X[i]
                argmin = i; // storing its index
        }
}
```

- *beta* is the *temporary minimum* during the execution.
- *beta* can *decrease only*
- *beta* is the maximum *finally* after all elements are visited

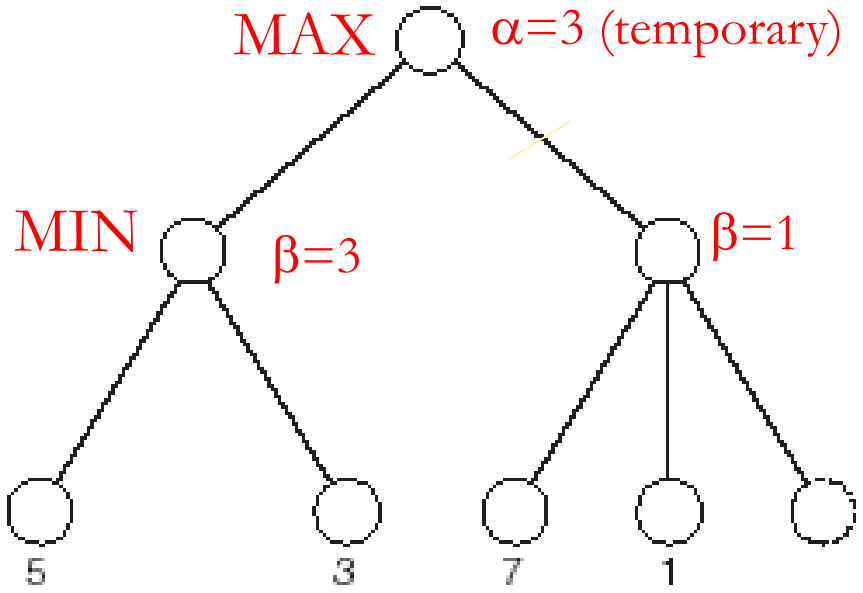
Alpha-Beta Pruning (cont'd)

- Recursive minmax algorithm is depth-first search
- DFS on game tree optimized with alpha-pruning
- Good player does not exhaustively or blindly search the game tree.
  - Pruning & optimized tree search

Depth-first search

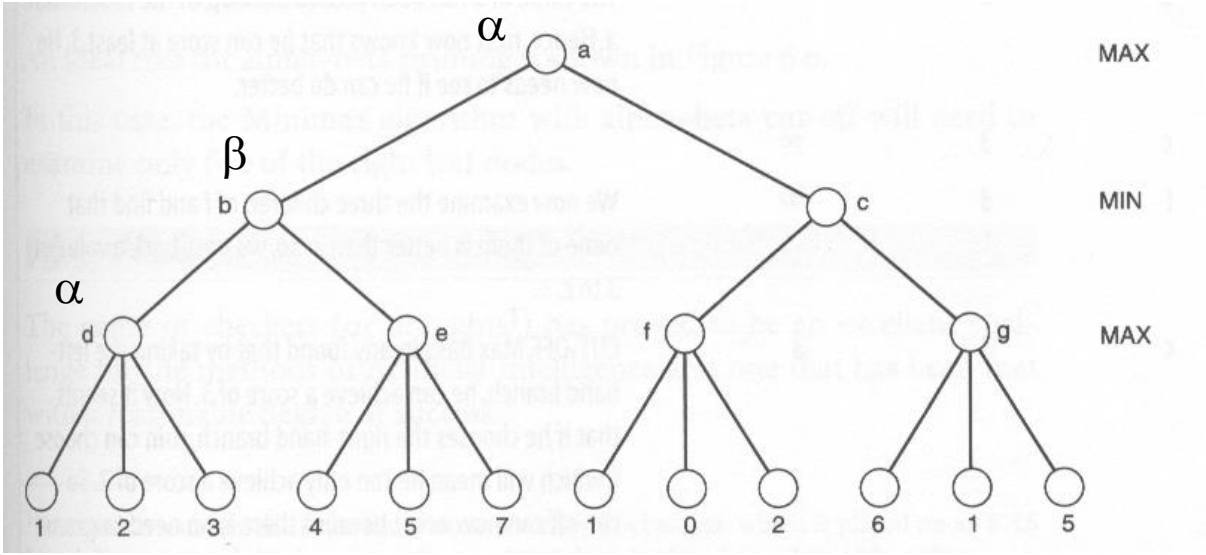


Alpha-beta Pruning



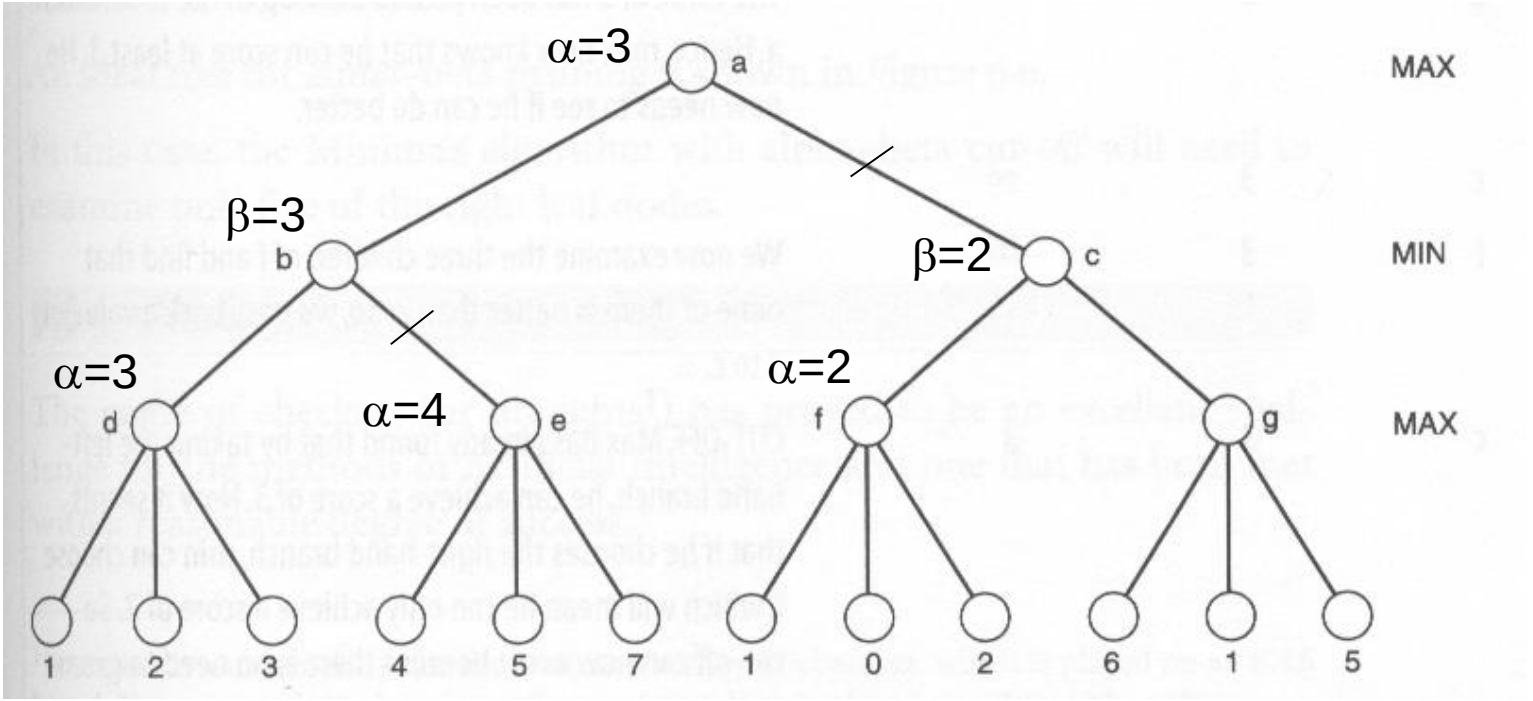
- Having examined the nodes with values 7 and 1, there is no need to examine the final node.
- MIN node
  - $\beta=1 \leq \alpha = 3$

Alpha-Beta Pruning (cont'd)



- MAX nodes → α, MIN nodes → β
- Initial values : α –Infinity, β Infinity
- Pruning check
  - MIN nodes: β < parent's α
  - MAX nodes: α ≧ parent's β

Alpha-Beta Pruning (cont'd)



Effectiveness of Alpha-Beta Pruning

- Ideal case
  - First branch can be pruned
  - Reduce the branching factor from b to $b^{1/2}$.
  - Double the search depth under the same computational cost
- Deep blue
  - Reduce the branching factor from 38 to 6 by applying alpha-beta pruning

Reducing Search Complexity

- Complexity of search tree
  - Determined by depth and breadth (b^d)
- Reduction of complexity
  - Control of depth: force the search to stop at certain level
  - Control of breadth: limit the number of choices when spanning the tree

Games of Chance

- The methods described so far do not work well with games of chance such as poker or backgammon.
- Expectminimax is a variant of minimax designed to deal with chance.
- Nodes have expected values based on probabilities.

Checkers

- In 1959, Arthur Samuel published a paper in which he described a computer program that could play checkers to a high level using minimax and alpha-beta pruning.
- Chinook, developed in Canada defeated the world champion:
  - Uses alpha-beta pruning.
  - Has a database of millions of end games.
  - Also has a database of openings.
  - Uses heuristics and knowledge about the game.

Checkers

- Blondie 24
  - Use evolutionary techniques to develop neural networks that can learn how to play checkers and how to win
  - Use standard Minimax algorithm
  - The static evaluation function is evaluated by neural networks (machine learning)
  - Is able to beat many human players, but not nearly at the level of Chinook
  - Without any help from humans

Chess

- In 1997, Deep Blue defeated world champion, Garry Kasparov.
- This has not yet been repeated.
- Current systems use parallel search, alpha-beta pruning, databases of openings and heuristics.
- The deeper in a search tree the computer can search, the better it plays.

Go (Wei-Qi)

- Ancient Oriental game
- Complex game played on a 19 x19 board.
- Average branching factor in search tree around 360 (compared to 38 for chess).
- Potentially Infinite depth
  - Ko(打劫): infinite loop occurs
- Methods use pattern matching or selective search to explore the most appropriate parts of the search tree.
- Due to vast complexity, traditional “brute force” methods don’t work on Go
- The Everest of search complexity
  - A number greater than there are atoms in the universe

Go (cont'd)

- Goals are vague and complicated
- Not easy to define *evaluation function*
  - Almost the same number of stones for black & white
  - Difficult to define *threatens*
- Not easy to judge the strengths/weakness (the state of game)
  - Power (~ investments): influence, potential for earning profits
  - Profits (~assets)

Summary

- Game tree can be used to represent two-player games.
- Searching game trees is very hard for all but the simplest games.
- Minimax is an algorithm that identifies the best move to make in a two-player game with perfect knowledge, assuming the entire tree can be examined.
- When the entire tree cannot be examined, a static evaluator is needed that can assign a score to any given position according to how well each player is doing and how likely each player is to win the game from that position.

Summary

- Alpha-beta pruning enables Minimax to run more efficiently because it removes unnecessary branches from the game tree.
- In the best case, alpha-beta pruning enables Minimax to search the game tree to double the depth that it would be able to search without pruning.

Summary

- Games such as Go-moku and ti-tac-toe have been solved, meaning that the result of a game is known from the start.
- Computers are able to beat the best players in the world at games such as Othello and checkers.
- Although Deep Blue beat Garry Kasparov in 1997, computers can win any professional human players.
- *AlphaGo* defeat *Lee Sedol* (9d professional player) on go game on *2016/3/9*.

Monte-Carlo Method

- Repeated *random sampling* to obtain numerical results
- Useful for those problems difficult or impossible to solve using other mathematical methods
- Used in optimization, numerical integration, probability distribution

Monte-Carlo Tree Search

- Converged slowly to *minimax*
- Can *minimize the search space* like alpha-beta pruning
- Do not need explicit evaluation function (game-end condition) as computer game of chess
- Concentrates on *most promising subtrees*
- Can be interrupted at any time and yields most promising move already found
- Successfully applied to computer-go since 2006

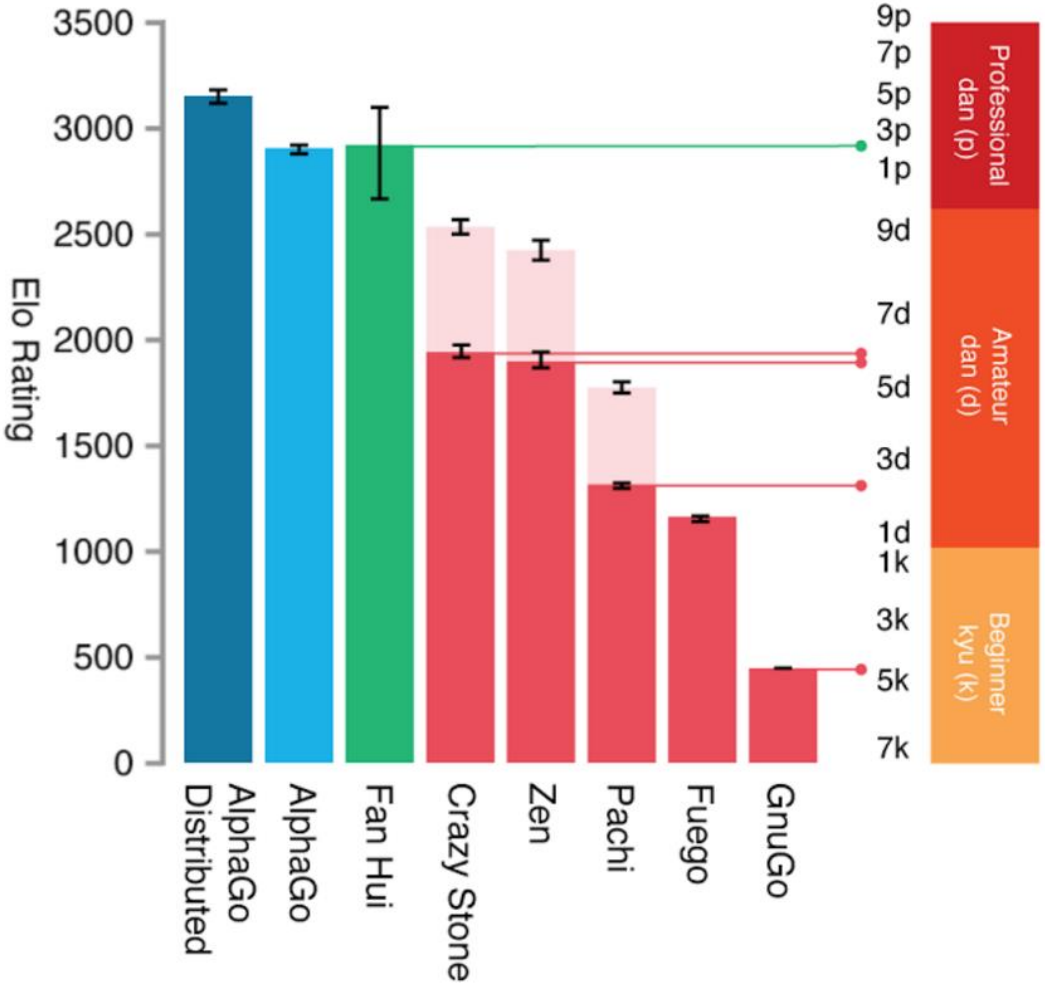
Alpha-Go

- Tree search guided by two deep neural networks
  - Millions of neurons
- Policy network
  - Goal: narrow the search to consider only the moves most likely to lead to a win
  - Trained by 30 millions moves
  - Discover new strategies by self playing games
- Value network
  - Goal : estimate the winner in each position
- Trained by human and computer play
  - Allow it to improve itself by just watching the playing games
  - Reinforcement learning from games of self play

Alpha-Go

- Google Cloud Platform
  - Distributed & parallel processing with multiple CPUs/GPUs
  - Open source library
  - Data flow graphs build with Python (TensorFlow)
  - tensorflow.org

Alpha-Go



Cited from *Google*

http://googleresearch.blogspot.tw/2016/01/alphago-mastering-ancient-game-of-go.html

- In the past, computers can defeat only amateurs
- AlphaGo defeated almost the smartest human expert

Further Thinking

- Applied to business competition or investment?
  - Your opponents hope to win
  - How to represent the state
  - How to evaluate the state
  - Search and make decision automatically
- How is *capability* defined in game?
  - Capabilities of players are measured through games
    - Probability to win → capability moves up/down
  - Another way of *measuring intelligence* !