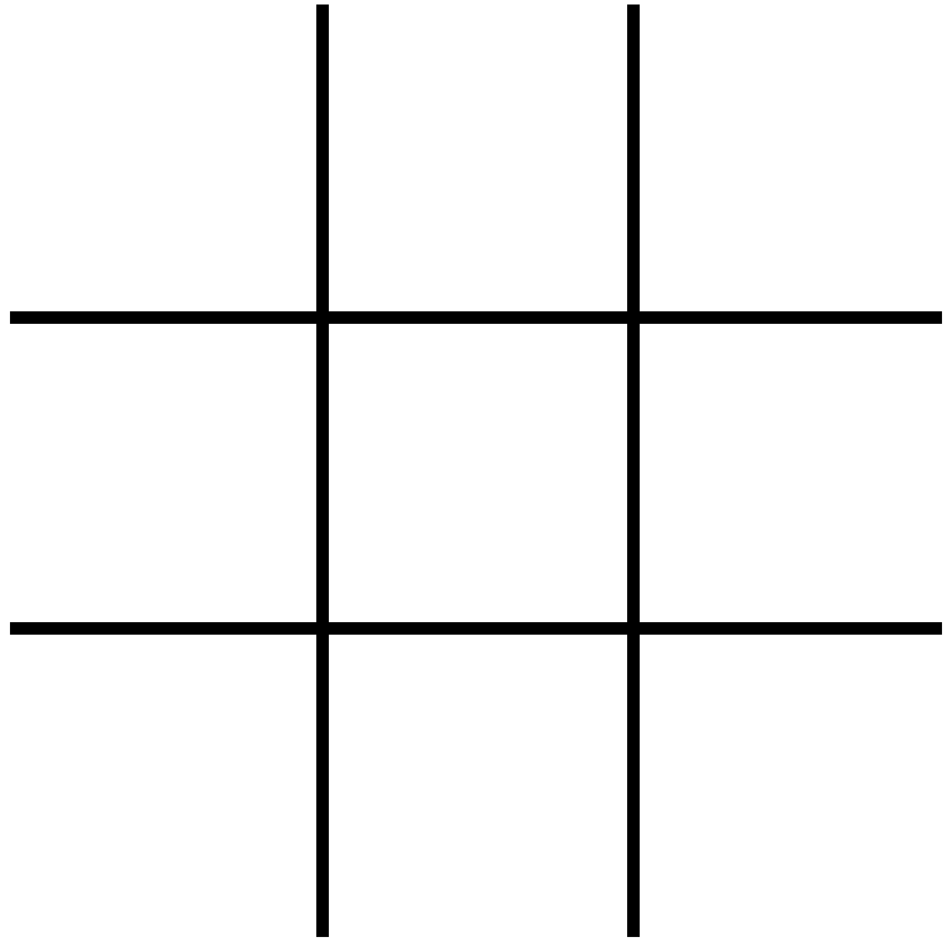
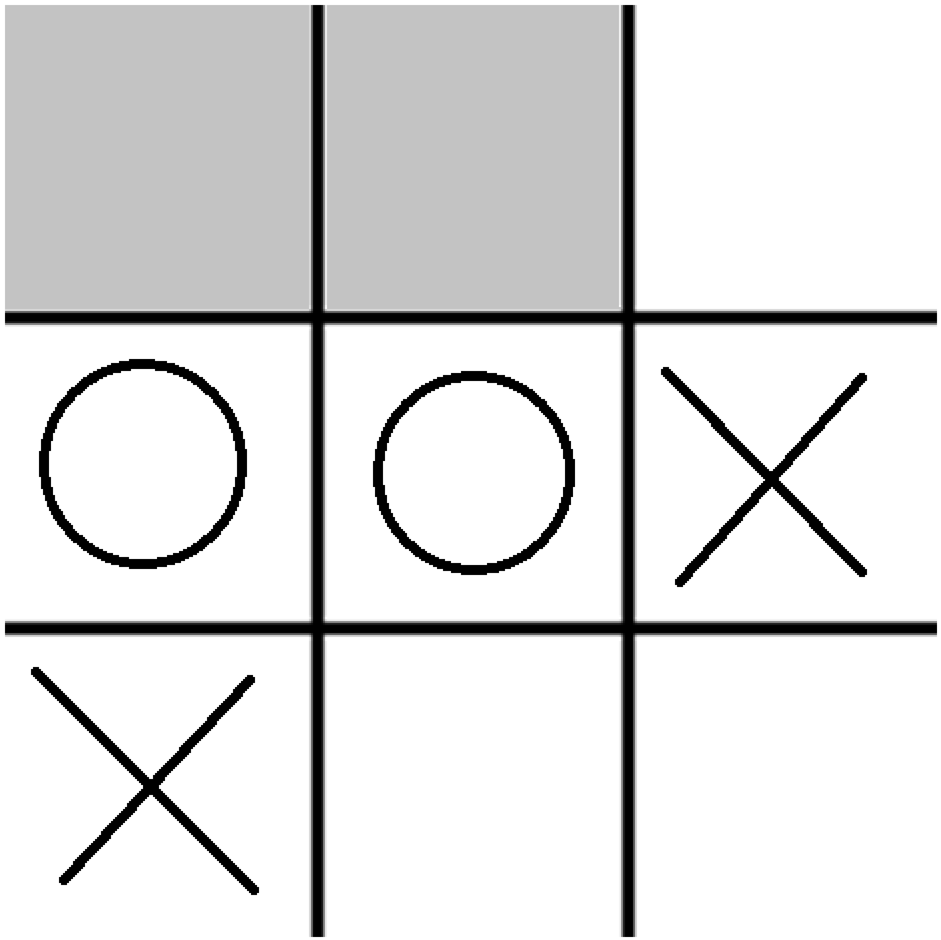
Chess AI's, How do they work?

Math Club 10/03/2011

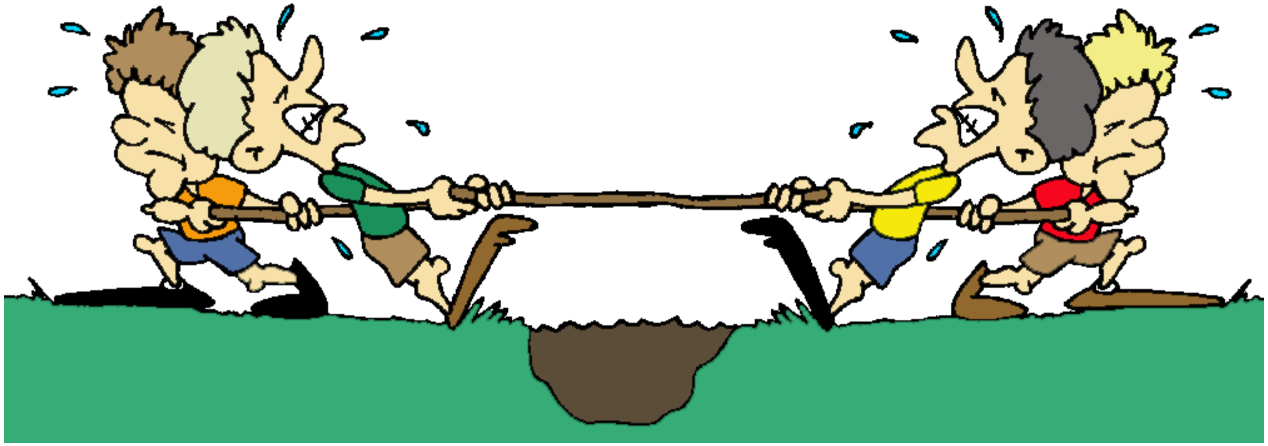
Tic Tac Toe



What is the best move for X?

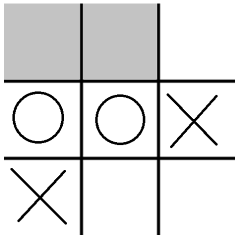


Minimax Algorithm



- Give the game a positive score if White is winning, negative score if Black is winning.
- White does all he can to make the score positive.
- Black does all he can do make the score negative.
- White knows that black is doing all he can to make the score negative
- Etc…

Can we … minimax … it?



But chess is more complicated!

- A simple Fermi problem:
- How many positions will a computer playing chess be able to calculate?
- b = branching factor (how many possible moves in a chess position)
- m = how many moves we need to look ahead
- t = how many positions the computer is able to look at every second
- $b * b * b * b$ ... branching, so $\frac{b^m}{t}$
- If b = 35, m = 8, t = 10,000,000 we have to wait more than 2 days to go through all the moves!

Heuristics!

- Heuristics are simple strategies that the computer can use to "approximate" things.
- Example: if you can take a piece with a pawn, then always do so.
- Caution: simple heuristics like these can lead to very bad moves.

Alpha-Beta Heuristic

- This heuristic usually reduces time and doesn’t do any worse than searching everything.
- Basically, we look at the better moves first.
- If we find a move worse than one we already looked at, we look at something else.

Alpha-Beta

- Suppose we find a really good move, A, so that no matter what they do, we have an advantage by the third move.
- Next we find some other move B, where the is some move they can make that neutralizes the position by the third move.
- Clearly B is inferior to A, so we can stop searching entirely.

How far can we go now?

- Suppose that we can always order the moves so that the best move is searched first.
- On your move you have to search b positions.
- But on their move, you only have to search 1 position and verify that it's inferior.
- Operations is $b * 1 * b * 1$ ... instead of $b * b * b * b$ ...
- So it takes only $\frac{\sqrt{b^m}}{t}$ seconds.
- If b = 35, m = 8, t = 10,000,000 it takes a fraction of a second to search all the moves! (instead of 2 days)
- In reality the best move is not always first searched.

Horizon Effect

- You play QxP, giving yourself a good score believing that you won a pawn.
- But one move after the “horizon”, you don’t see PxQ, which loses you a queen.
- Solution: quiescence search – at the end of the search tree, only consider “quiet” moves.

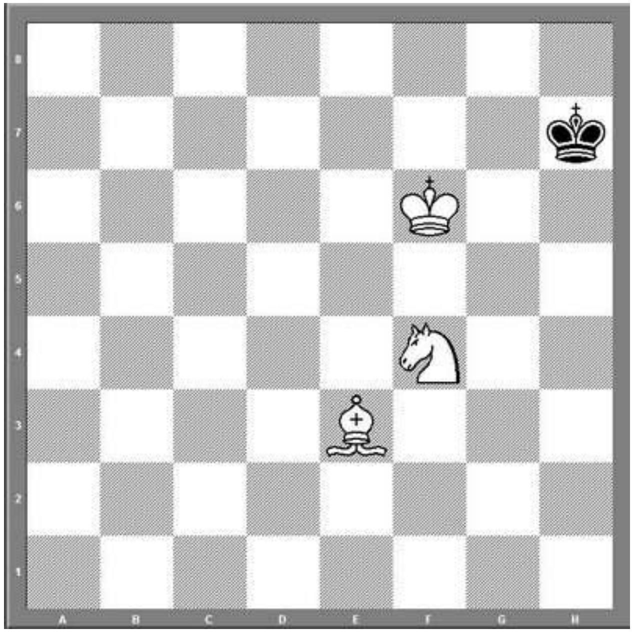
Opening Books

- For the start of the game, the computer already has prepared a set of opening moves – so it doesn't have to search in the opening.

	N	%	Av	Perf	Fact	Prob	[%]
Rybka4.ctg	627950	57.9	2513	2562			
1.e4	419802	59.8	2539	2591	0	48.3	51.4
1.d4	134300	54.6	2460	2504	0	22.6	24.1
1.Nf3	35048	54.7	2481	2528	0	12.7	13.5
1.c4	21228	51.5	2422	2461	0	10.4	11.0
1.Nc3	3233	49.4	2442	2496	0	0.9	0
1.h3	3105	63.5	2605	2622	0	1.4	0
1.b3	2568	51.2	2488	2516	0	0.7	0
1.g3	2185	53.1	2494	2515	0	0.7	0
1.f4	1804	45.4	2320	2325	0	0.5	0
1.d3	1250	51.3	2566	2582	0	0.5	0
1.b4	1076	43.0	2221	2212	0	0.2	0
1.a3	945	56.9	2560	2589	0	0.5	0
1.e3	611	51.7	2545	2565	0	0.2	0
1.c3	429	49.8	2511	2550	0	0.2	0
1.g4	114	23.7	2278	2228	0	0	0
1.a4	71	50.0	2508	2550	0	0	0
1.h4	71	23.9	2381	2340	0	0	0
1.f3	44	30.7	2371	2388	0	0	0
1.Nh3	34	32.4	2302	2322	0	0	0
1.Na3	32	14.1	2372	2327	0	0	0

Endgame Tablebases

- Use brute force to prepare a database of endgame positions and their optimal responses, so you can play perfectly if there are few enough pieces left.



Challenge

- Survive for as long as possible against Chessmaster.