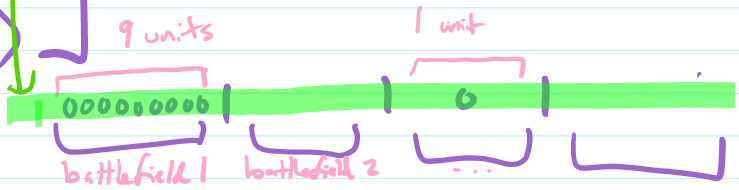
Blotto

10 units
4 battlefields

(10, 0, 0, 0)
(9, 1, 0, 0)
(9, 0, 1, 0)
(9, 0, 0, 1)
⋮
(0, 0, 0, 10)

# of pure strategies = $\binom{13}{10}$ = 286



string of 10 0's
3 1's
choose 10 locations of 0's out of 13 locations

538: 100 units
10 battlefields
$\binom{109}{100}$ ≈ 4 trillion

Chess as a simultaneous game

chess strategy = function from chess positions to moves

1) P1, P2 simultaneously choose strat
2) play game accordingly

assume ≈ 30 legal moves per pos

assume ~30 legal moves per pos

$\sim 30^{(10^{40})}$ different strategies

Minimax(pos)

b = branching factor = avg moves possible

d = depth = avg length

pos ≈ b^d

If pos is terminal (game over), return value determined by rules ↪ check who won: +1 : P1, 0 : draw, −1 : P2

Else if pos is P1's turn then return $\max_{pos \to pos'} MM(pos')$

Else return $\min_{pos \to pos'} MM(pos')$

	very rough est of # pos
Tic-Tac-Toe	$\leq 3^9 \approx 20000$
Mancala	$\leq \binom{49}{36} \times 262$ billion
2-player Yahtzee	$\leq 4 \cdot 10^{18}$
Checkers	$\leq 10^{20}$
Chess	$\leq 10^{43}$
Go	$\leq 10^{172}$

What to do with games of high complexity?

heuristics – estimate of position value

Ex: checkers — % of remaining pieces that are black (scaled to (−1, 1)) — 8 / 4 — heuristic = 1/3

chess — assign value to each type of piece: pawn = 1, knight, bishop = 3, rook = 5, queen = 9 — % of value of remaining pieces belonging to white (scaled)

Minimax(pos, h, depth) → depth bound; higher bounds yield results closer to unbounded MM

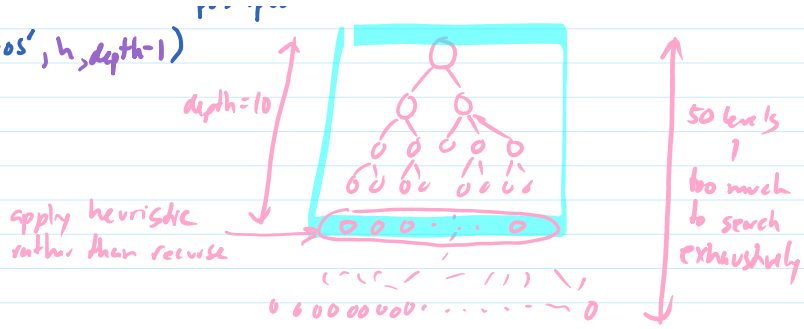
If pos is terminal, return value(pos)

If depth == 0, return h(pos)

Else if pos is P1's turn then return $\max_{pos \to pos'} MM(pos', h, depth-1)$

Else return $\min_{pos \to pos'} MM(pos', h, depth-1)$

Else return $\min_{pos \to pos'} MM(pos', h, depth-1)$



Negamax(pos, h, depth, sign)

If pos is terminal, return value(pos) · sign

If depth == 0, return h(pos) · sign

Else return $\max_{pos \to pos'} -MM(pos', h, depth-1, -sign)$

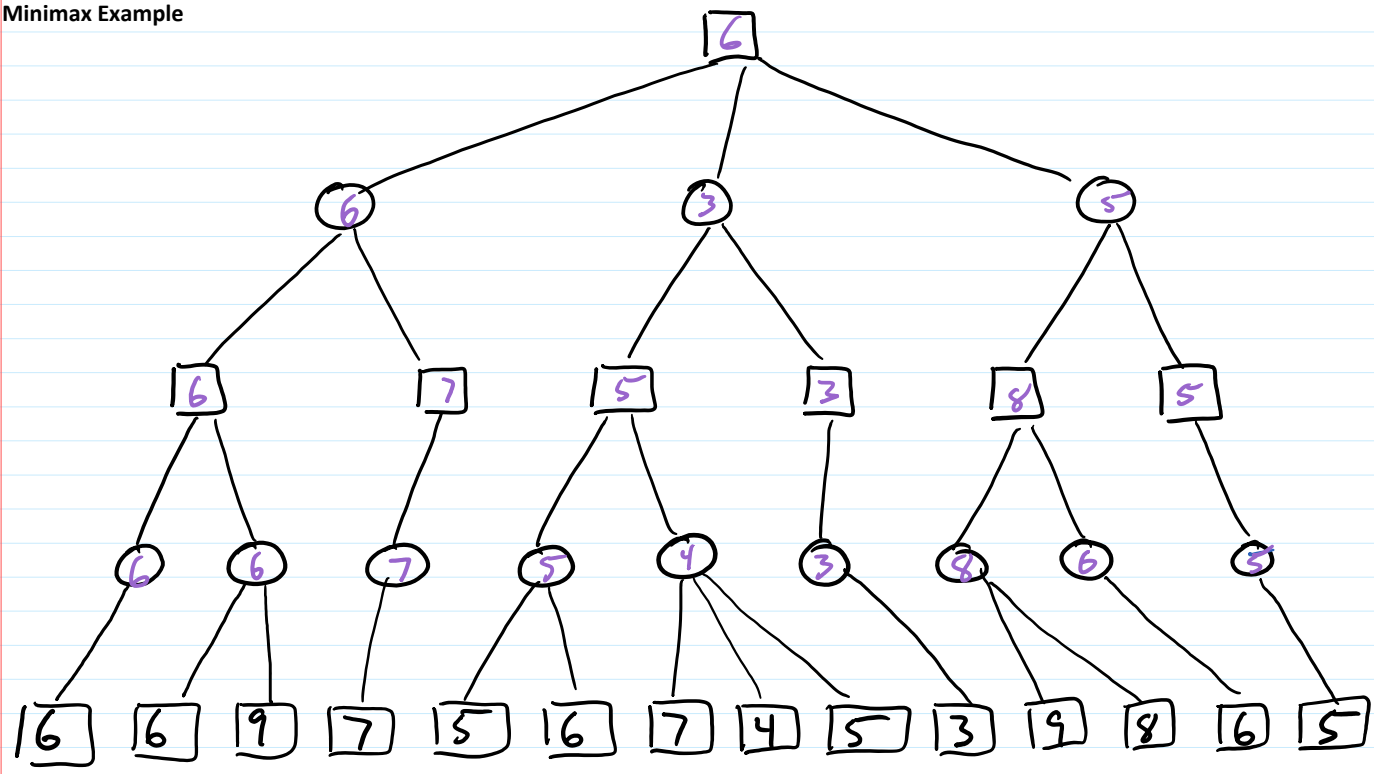
Iterative Deepening - to get value from MM in set time w/ depth high

depth ← 2
while not out of time
do MM(pos, h, depth)
depth ← depth + 1
return value, best move returned by last call to MM that finished

other ideas - search deeper in promising branches (those that seem like good moves)

shallower on bad branches

Minimax Example



Modified example from http://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning