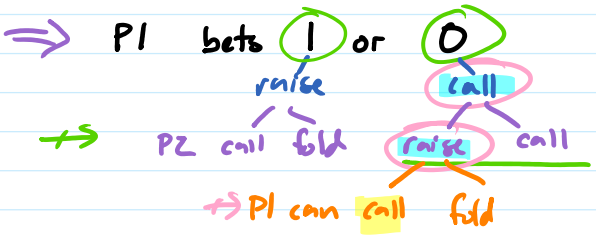
One-Card Poker

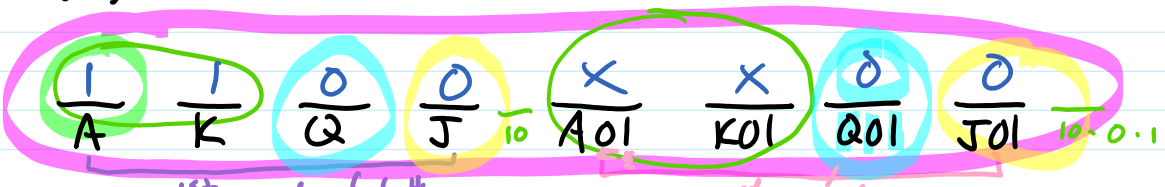
Each player antes 1 unit

Deal one card to each player

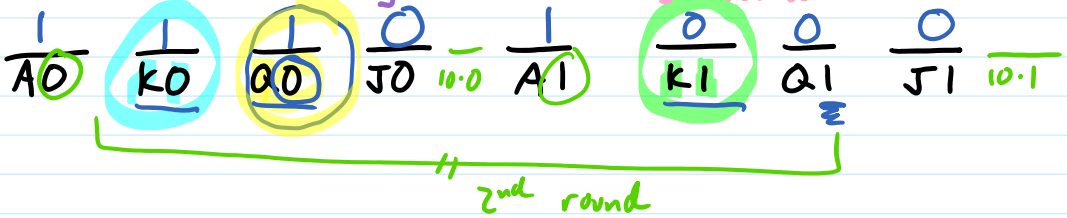


4 card deck A, K, Q, J of clubs

P1 strategy :



P2 strategy :



prob	P1	P2	P1 payoff
1/12	A	K	+1
	A	Q	+1
	A	J	+1
	K	A	-2
	K	Q	+1
	K	J	+1
	Q	A	-1
1/12	Q	K	-1
	Q	J	+1
	J	A	-1
	J	K	-1
1/12	J	Q	-1

≤ 256 rows

256 columns

$-\frac{1}{12}$

P1 expected payoff : $\frac{1}{12}\cdot 1 + \cdots + \frac{1}{12}\cdot -1 + \cdots + \frac{1}{12}\cdot -1 = -\frac{1}{12}$

Sizes of Games

Minimax(pos)

If pos is terminal, return value determined by rules
↳ +1 if P1 wins
0 if draw
-1 if P2 wins

b = branching factor
≈ avg moves available

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

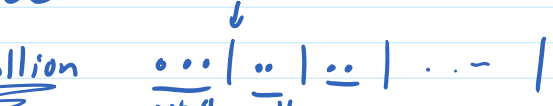
d = depth
= avg length

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

very rough est of # pos
$\leq 3^9 \approx 20000$

Tic-Tac-Toe

d≈30, b≈6 → Mancala (12 home pits, 3 seeds each) $\leq \binom{49}{36} \approx 262$ billion



2-player Yahtzee $\leq 4 \cdot 10^{18}$ (4 quintillions)

Checkers $\leq 10^{20}$

Chess $\leq 10^{43}$

b≈100, d≈100 → Go $\leq 10^{172}$

http://en.wikipedia.org/wiki/Game_complexity

What to do with games of high complexity?

heuristics - estimate of pos value

Ex: checkers — % remaining pieces that are black (scaled to (-1, 1))

0 8 h = 1/3
0 4

chess — assign value to each type of piece

add values of remaining pieces

Q = 9 (10) ×1
R = 5 ×2
K, B = 3 (3.5)
P = 1 ×5
19

×2 ×1
×2
11

Minimax(pos, h, d) — d: depth bound

If pos is terminal, return value(pos) — value determined by rules

If depth == 0, return h(pos)

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

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

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

minimax with P1/P2 cases wrapped together

Negamax (pos, h, depth, sign)

if pos is terminal return sign · value(pos)

if depth == 0, return h(pos) · sign

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

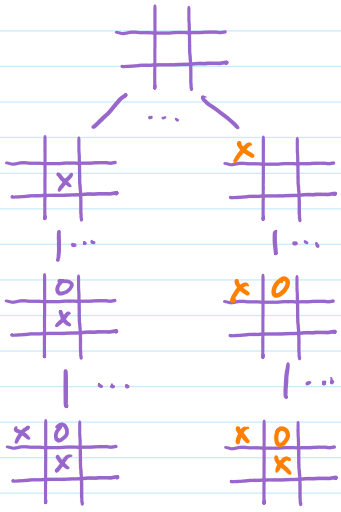
Iterative Deepening - get result from as deep a MM search as possible
have result available at any time

further ideas
search deeper in more

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

depth=2 5 sec ←
=3 12 sec
=4 40 sec
=5 2 min

Transposition Table - (like memo in memoization)



save values for positions in case they reappear at other pos in tree

need replacement policy to avoid t.t. getting too big

stop storing new things?
replace least recently used item?
;

(cache replacement policies)