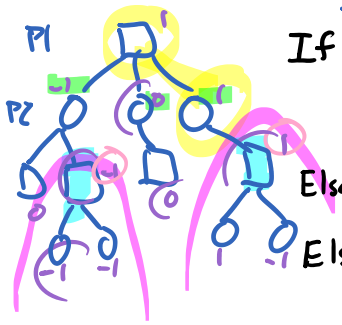
Sizes of Games

Minimax(pos)



If pos is terminal, return value determined by rules of game
+1 if P1 win
0 draw
−1 if P2 win

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

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

00 1 0 1 0

Tic-Tac-Toe $\leq 3^9$

Mancala $2 \cdot \binom{36+13}{13} = \binom{49}{13} \cdot 2 \approx$ 500 billion $b \approx 4$ $d \approx 36$ $b^d = 4^{36} = 2^{72} \approx 10^{38}$ or 10^{18} using better estimate ...

2-player Yahtzee $4 \cdot 10^{18}$

Checkers $\leq 10^{20}$

Chess $\leq 10^{45}$

Go $\leq 10^{172}$

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

What to do with games of high complexity?

heuristics - estimates of positions' values that can be calculated quickly

Ex: checkers K:2 N:1

chess Q:9 (10) R:5 B,k:3 (3.5) P:1

material strength
mobility
attack −
castling rights
pawn position
king safety −

kalah: $\frac{\#\text{seed in P1 store} - \#\text{seeds in P2 store}}{36}$

Minimax(pos, h, d) — d: how deep to search tree

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

If d==0 return h(pos)

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

Else return min mm(pos', h, d−1)

pos → pos'

Negamax(pos, h, depth, sign)

if pos is terminal return value(pos)

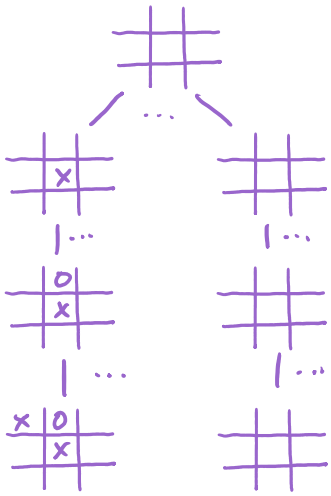
if depth == 0, return h(pos)

else return max_{pos→pos'} MM(pos', h, depth-1,)

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

depth ← 2

Transposition Table



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

Minimax w/ Transposition Table

Minimax (pos, h, d)

If pos is terminal, return value(pos)

If depth == 0, return h(pos)

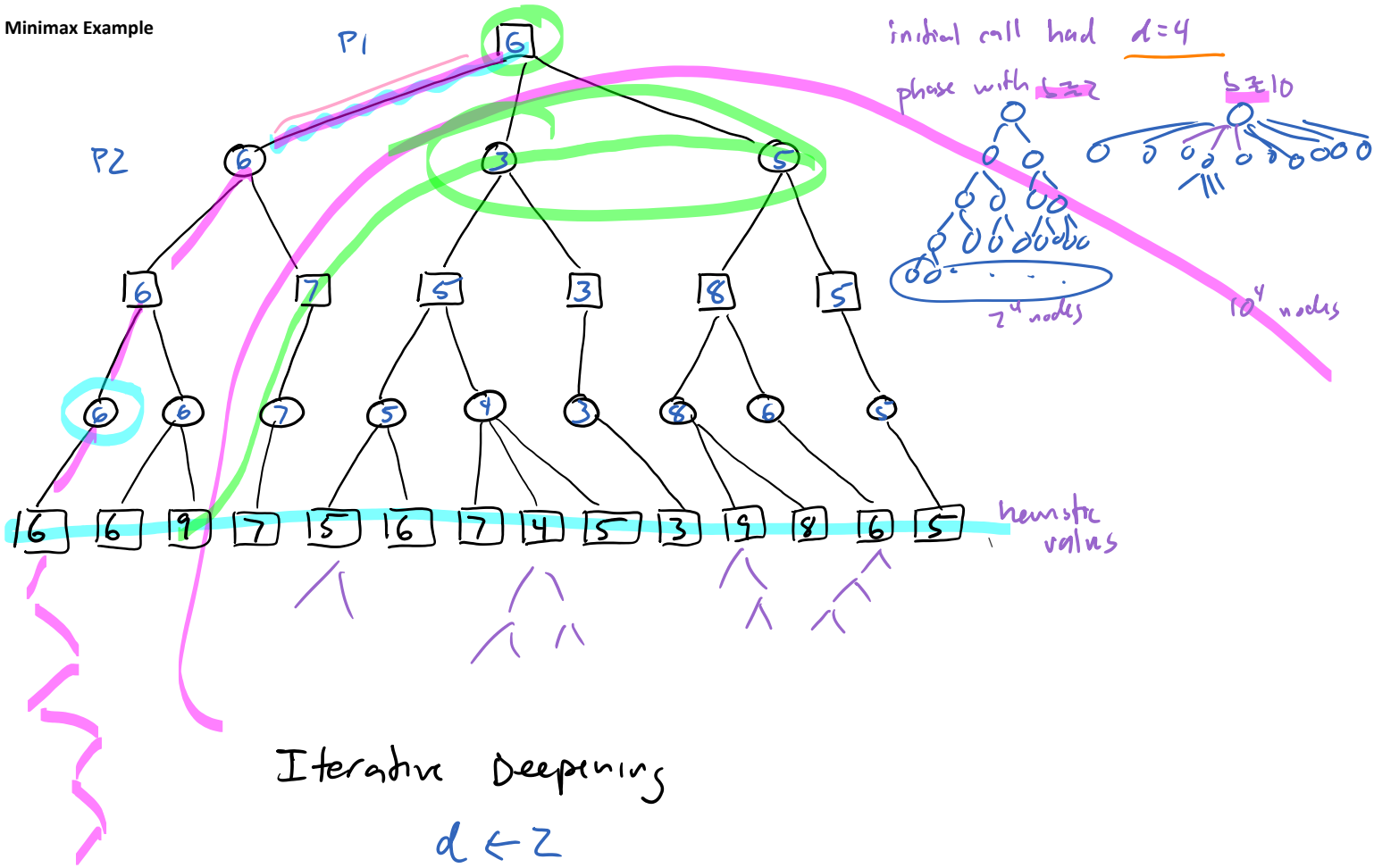
Else if pos is P1's turn then

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

Else

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

Minimax Example



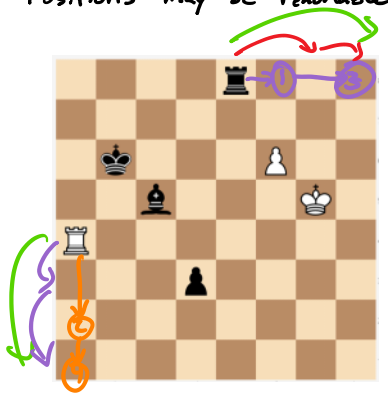
Iterative Deepening

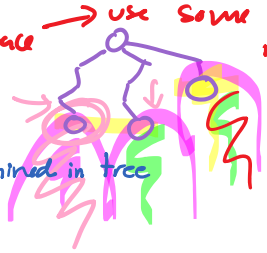
```
d ← 2
while time left
    move, value ← MM(pos, h, d)
    d ← d+1
return result from last completed MM
```

Transposition Table

↪ like memo, but allowed to remove entries when out of space → use some cache replacement policy

Positions may be reachable by multiple sequences of moves





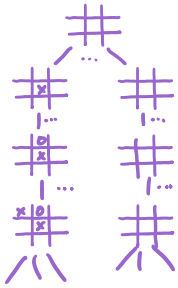
Keep table of values for all positions examined in tree

Keys: positions, depth

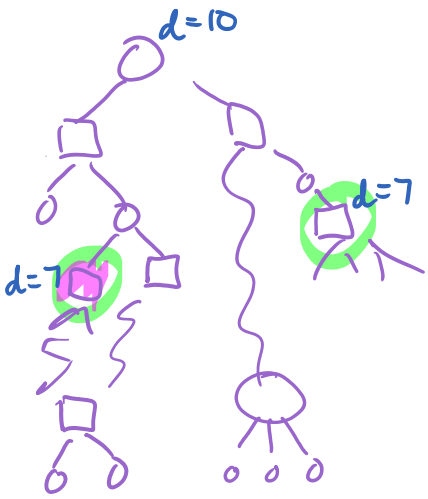
Values: values computed by minimax

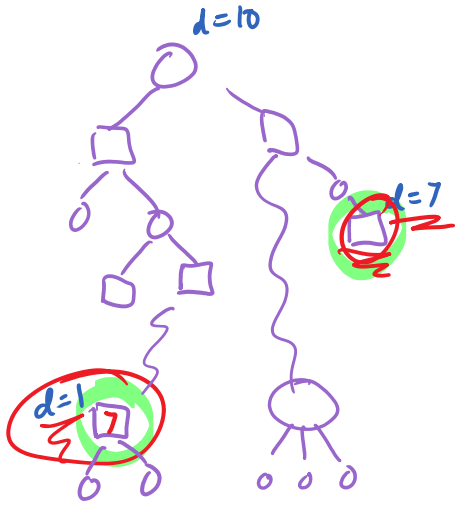
Add check at start of Minimax

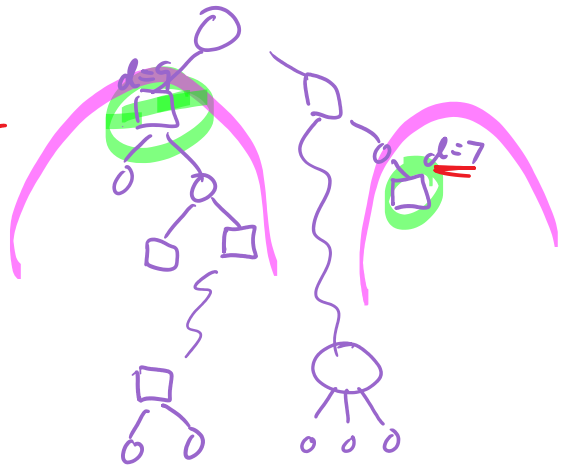
if pos in tt, return value from tt w/ depth ≥ d



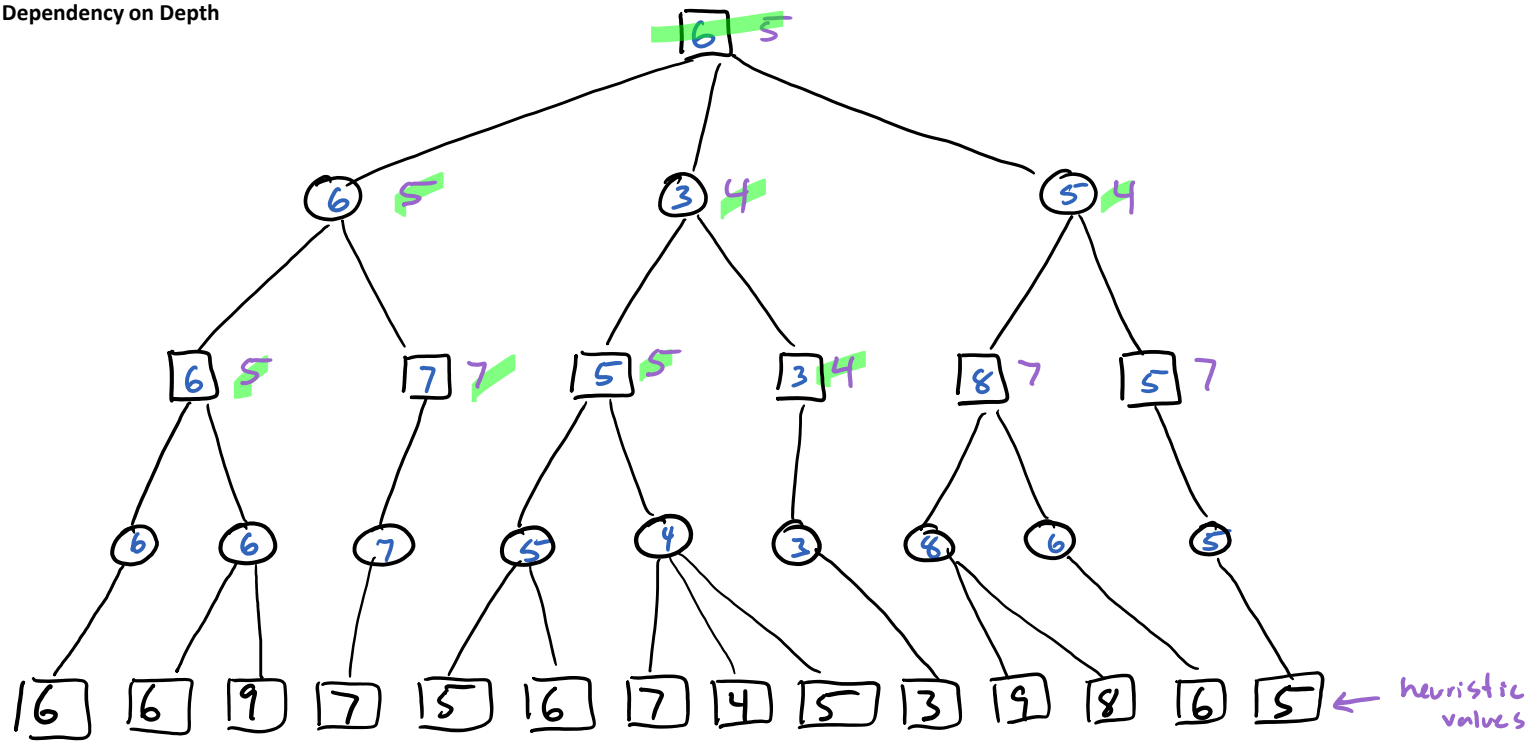
Save returned values in table







Dependency on Depth



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