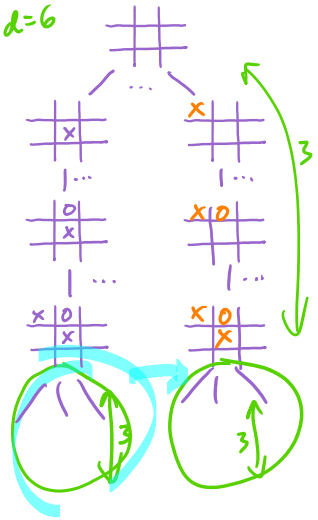
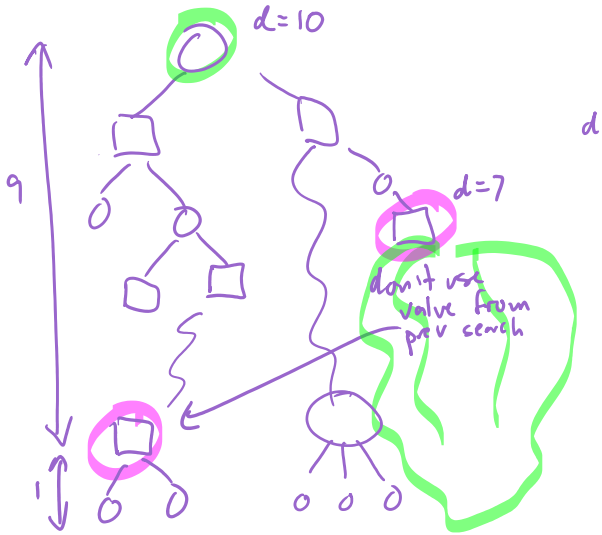
Non-constant sum Blotto

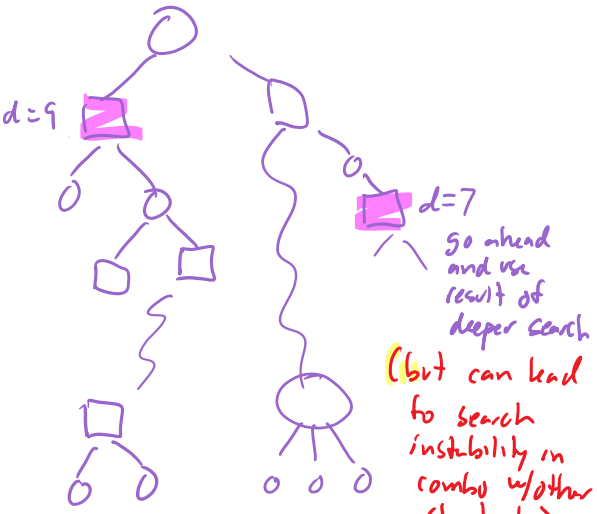
```
Minimax (pos, h, d, tt)
    If pos is terminal, return value(pos)
    If depth == 0, return h(pos)
    if pos in tt with depth ≥ d then return value from tt
    Else if pos is P1's turn then
        add pos, (d, ____) to tt
        return max_{pos→pos'} mm(pos', h, d-1, tt)
    Else
        add pos, (d, ____) to tt
        return min_{pos→pos'} mm(pos', h, d-1, tt)
```

Transposition Table – like memo in memoization

save values for positions and depth to which we searched, in case they reappear at other pos in tree







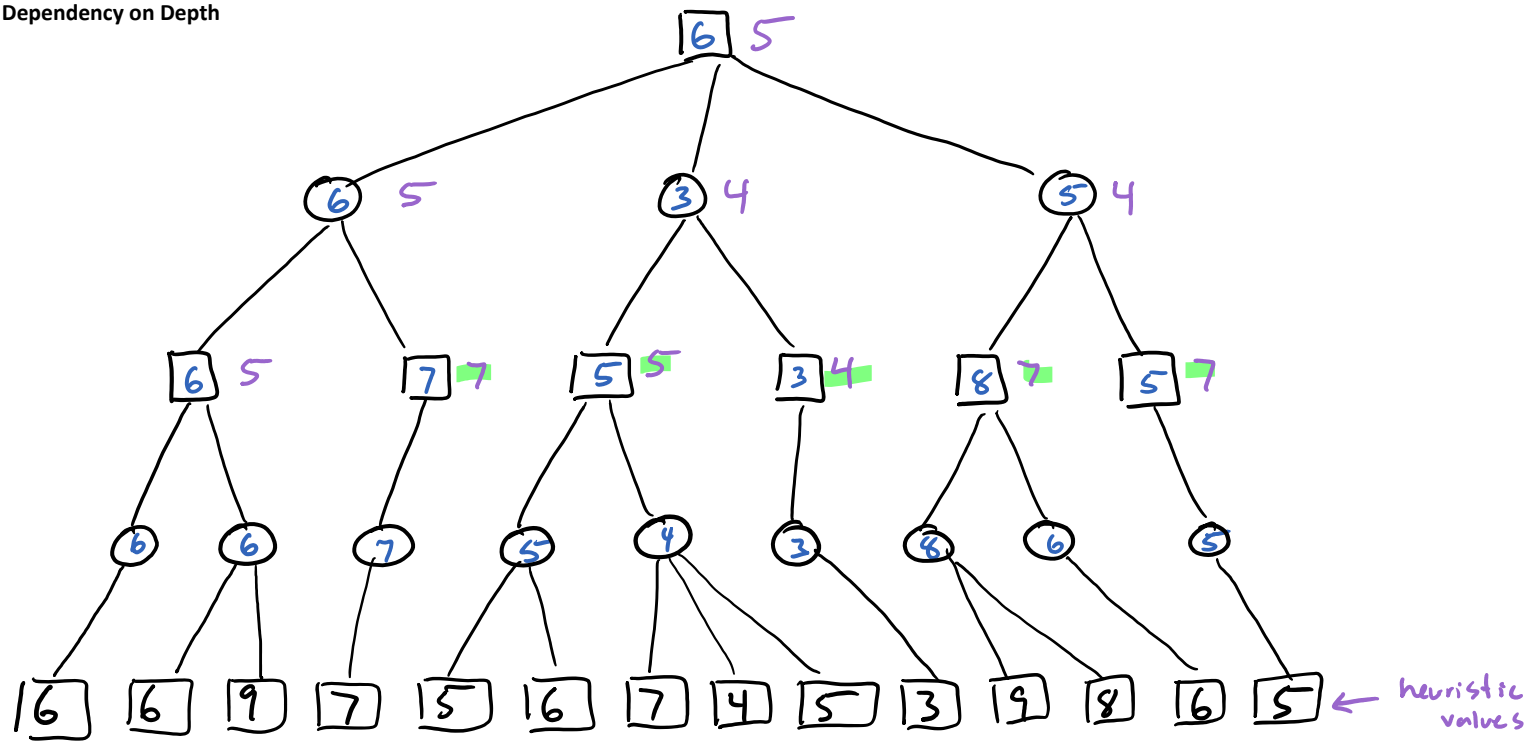
size of tree ≈ b^d

↑ can change during game

Iterative deepening

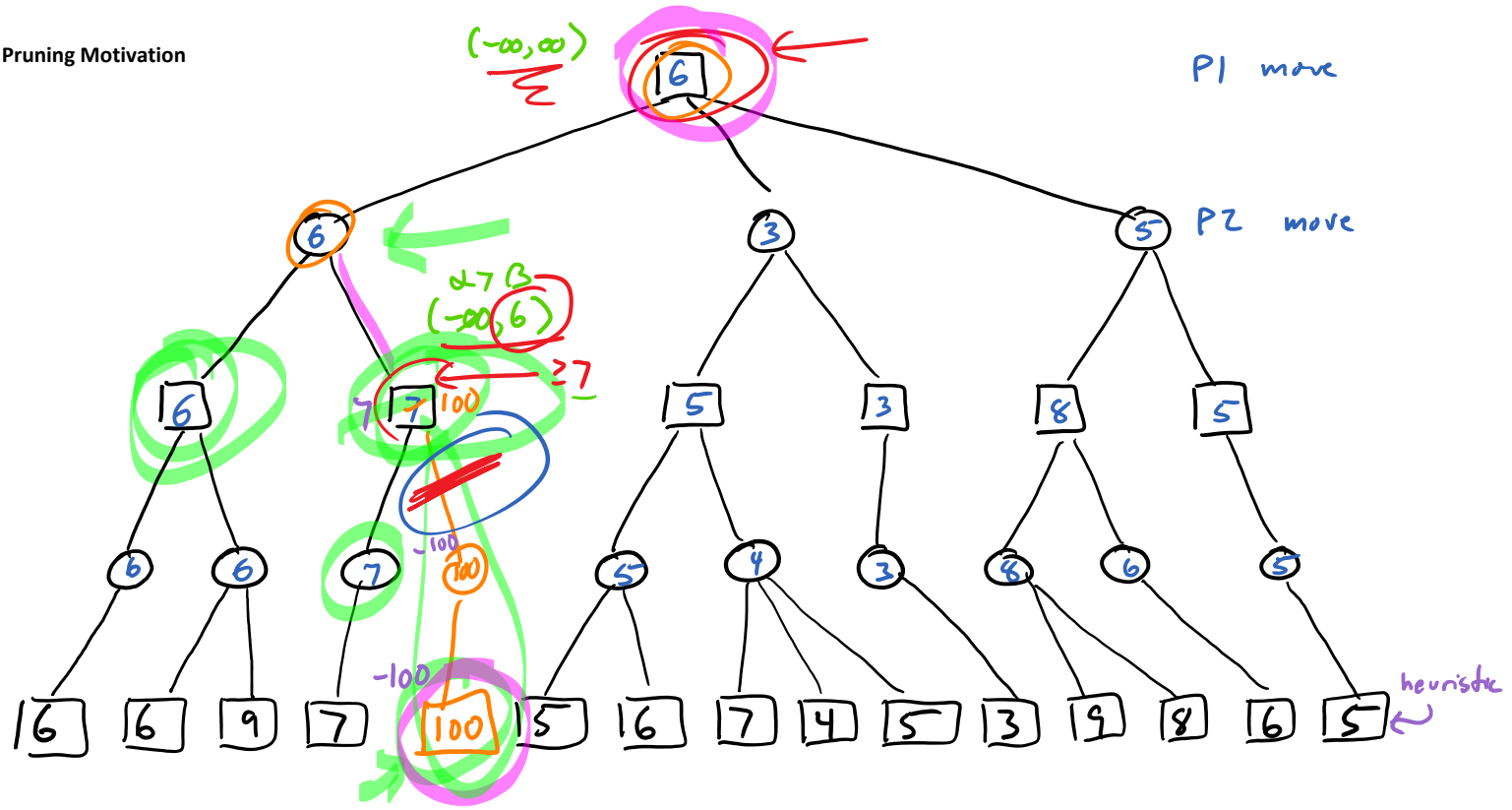
```
d ← 2
while time left
    move, value ← mm(pos, h, d, tt)
    d ← d+1
return result of last finishing mm
```

Dependency on Depth



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

Pruning Motivation



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

Alpha-Beta Pruning

range of values we need to distinguish between

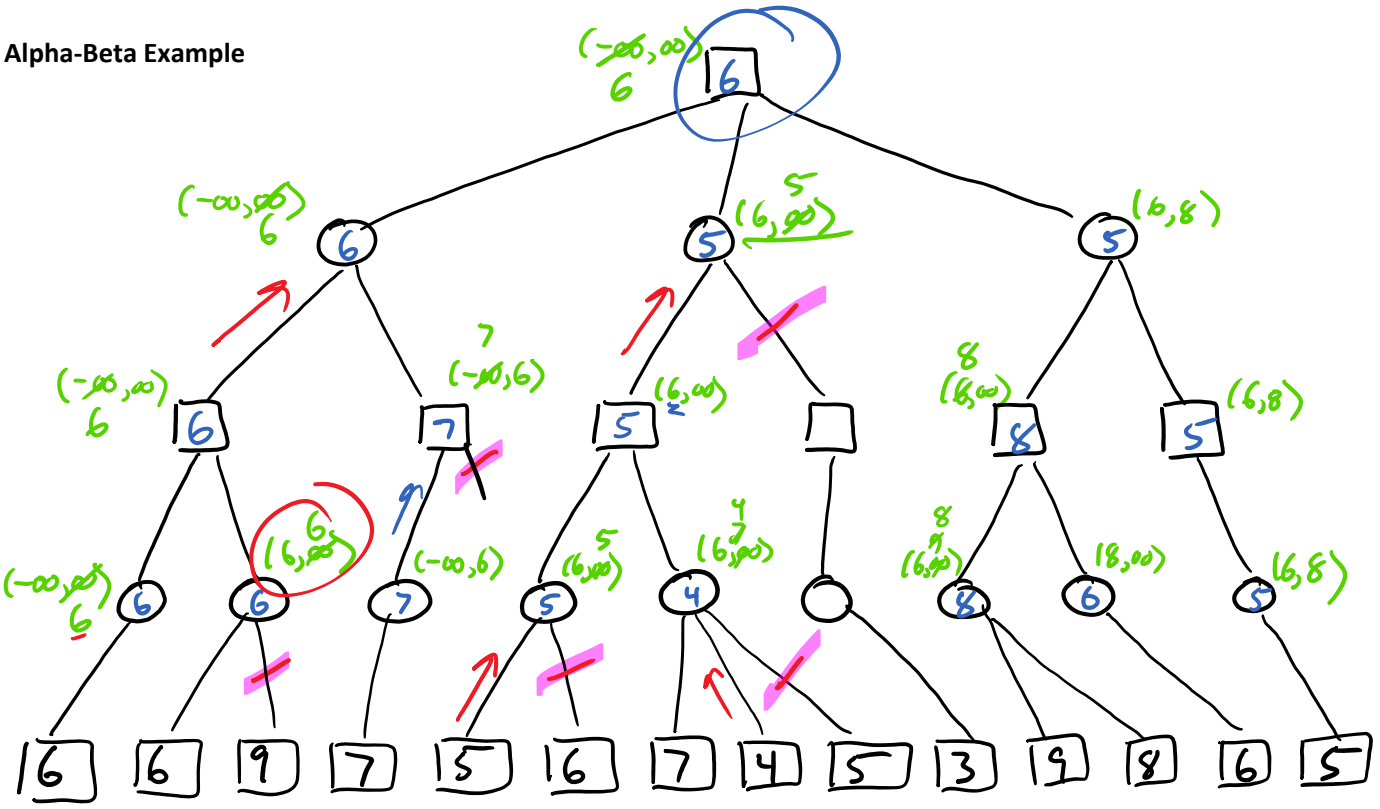
Alpha-Beta (p, α, β, h, depth) returns

post-conditions

- value (p) if p is terminal
- h (p) if depth = 0
- MM (p, h, d) if $\alpha \le MM(p,h,d) \le \beta$
- lower bound $\ge \beta$ on MM(p,h,d) if $MM(p,h,d) \ge \beta$
- upper bound $\le \alpha$ on MM (p,h,d) $MM(p,h,d) \le \alpha$

```
if depth = 0 then return heuristic(p)
if p is terminal then return value(p)
if p is a max position (P1)
    a ← −∞   (value of best move so far)
    for each position p' reachable in one move from p and while α < β
        a ← max(a, AB(p', α, β, h, depth-1))
        α ← max(α, a)
    return a
else (P2)
    b ← ∞
    for each position p' reachable in one move from p while α < β
        b ← min(b, AB(p', α, β, h, depth-1))
        β ← min(β, b)
    return b
```

Alpha-Beta Example



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