Sizes of Games

Minimax(pos)

If pos is terminal,

Else if pos is P1's turn then return

Else return

Tic-Tac-Toe

Mancala

2-player Yahtzee

Checkers

Chess

Go

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

http://xkcd.com/1002/

What to do with games of high complexity?

heuristics -

Ex : checkers

chess

Minimax (pos)

If pos is terminal, return value(pos)

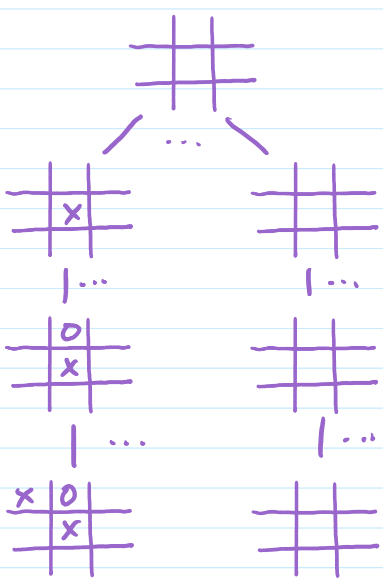
Else if pos is P1's turn then return $\max_{pos \to pos'} MM(pos', \quad)$

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

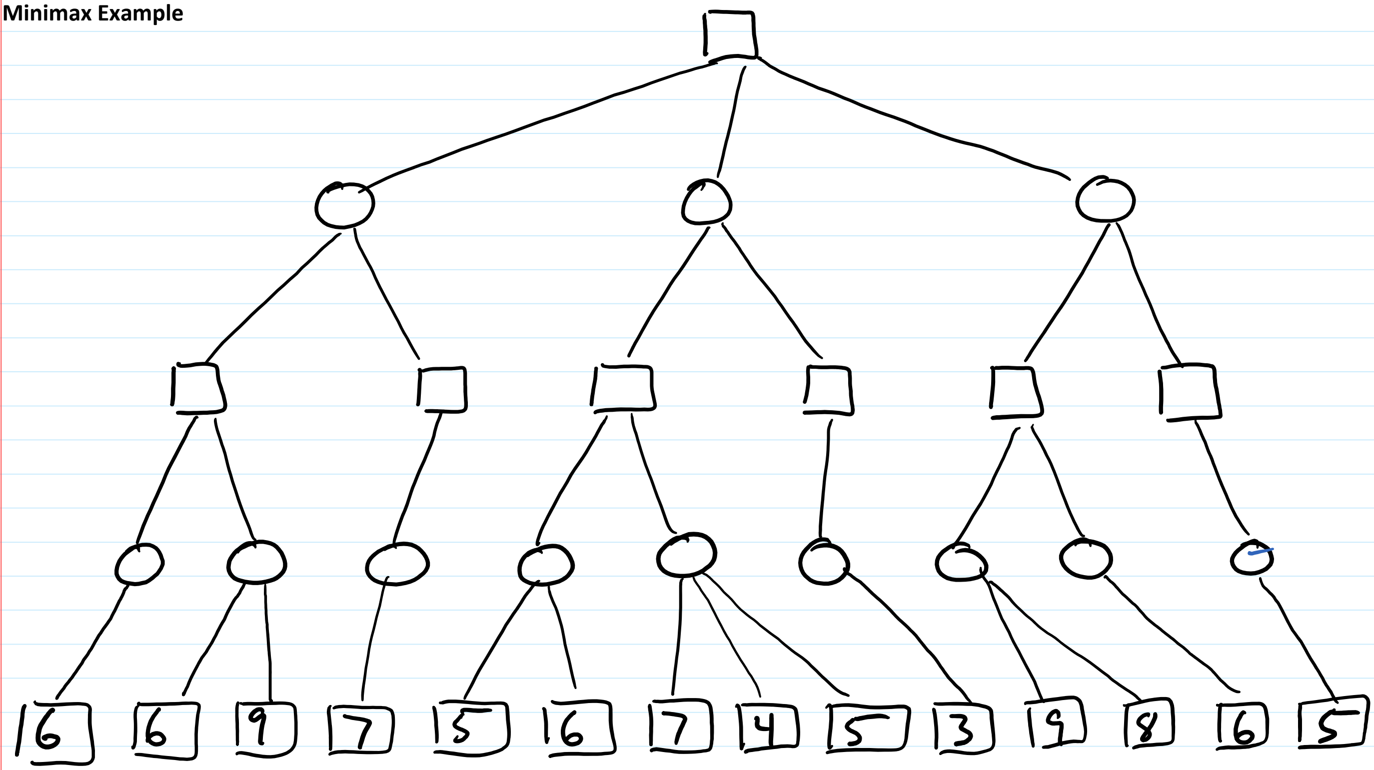
Negamax(pos, h, depth, color)

Iterative Deepening

Transposition Table



Minimax Example



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

Alpha-Beta Pruning

Alpha-Beta(p, α, β, depth) returns

if depth = 0 then return heuristic(p)

if p is terminal then return value(p)

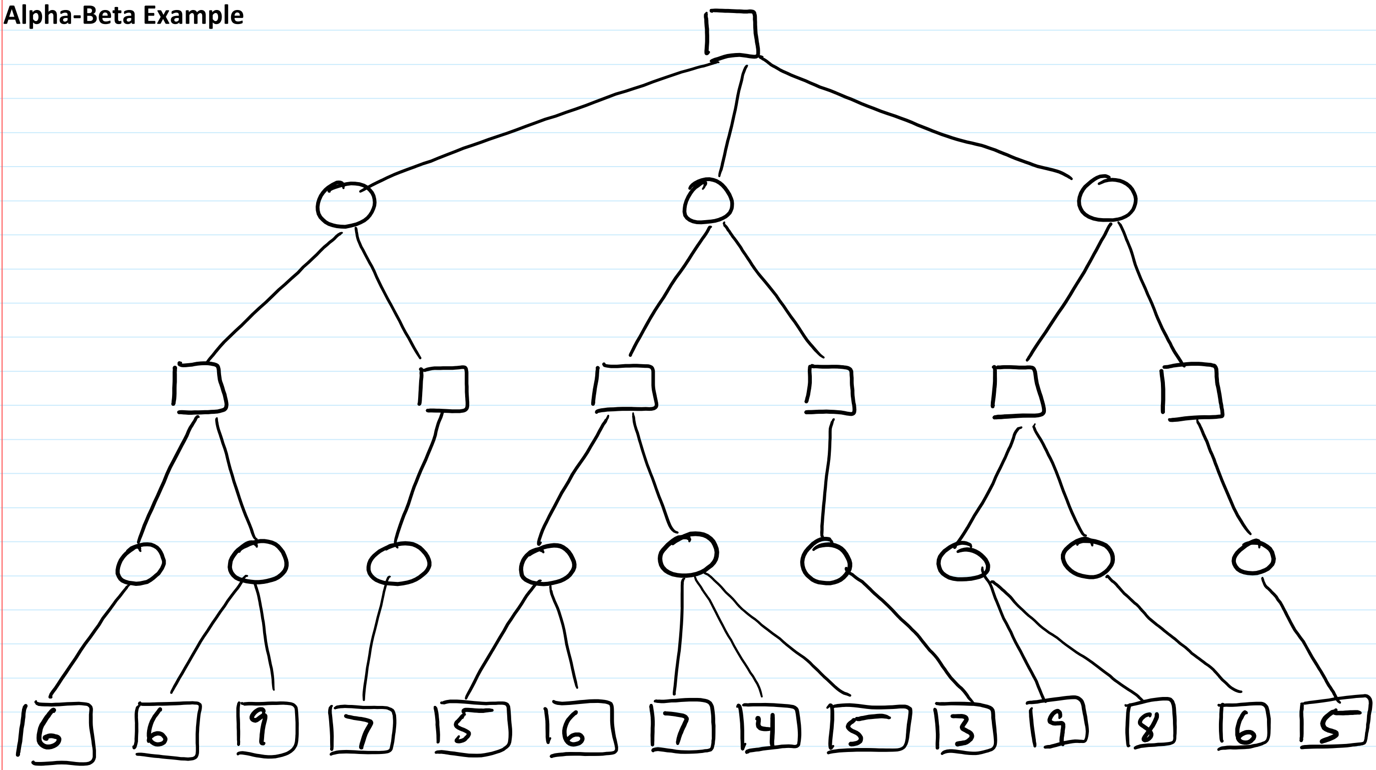
if p is a max position

for each position p' reachable in one move from p

else

for each position p' reachable in one move from p

Alpha-Beta Example



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

Scout (p, α, β, depth)

if depth = 0 then return heuristic(p)

if p is terminal then return value(p)

if p is a max position

for each reachable position p' and while $\alpha < \beta$

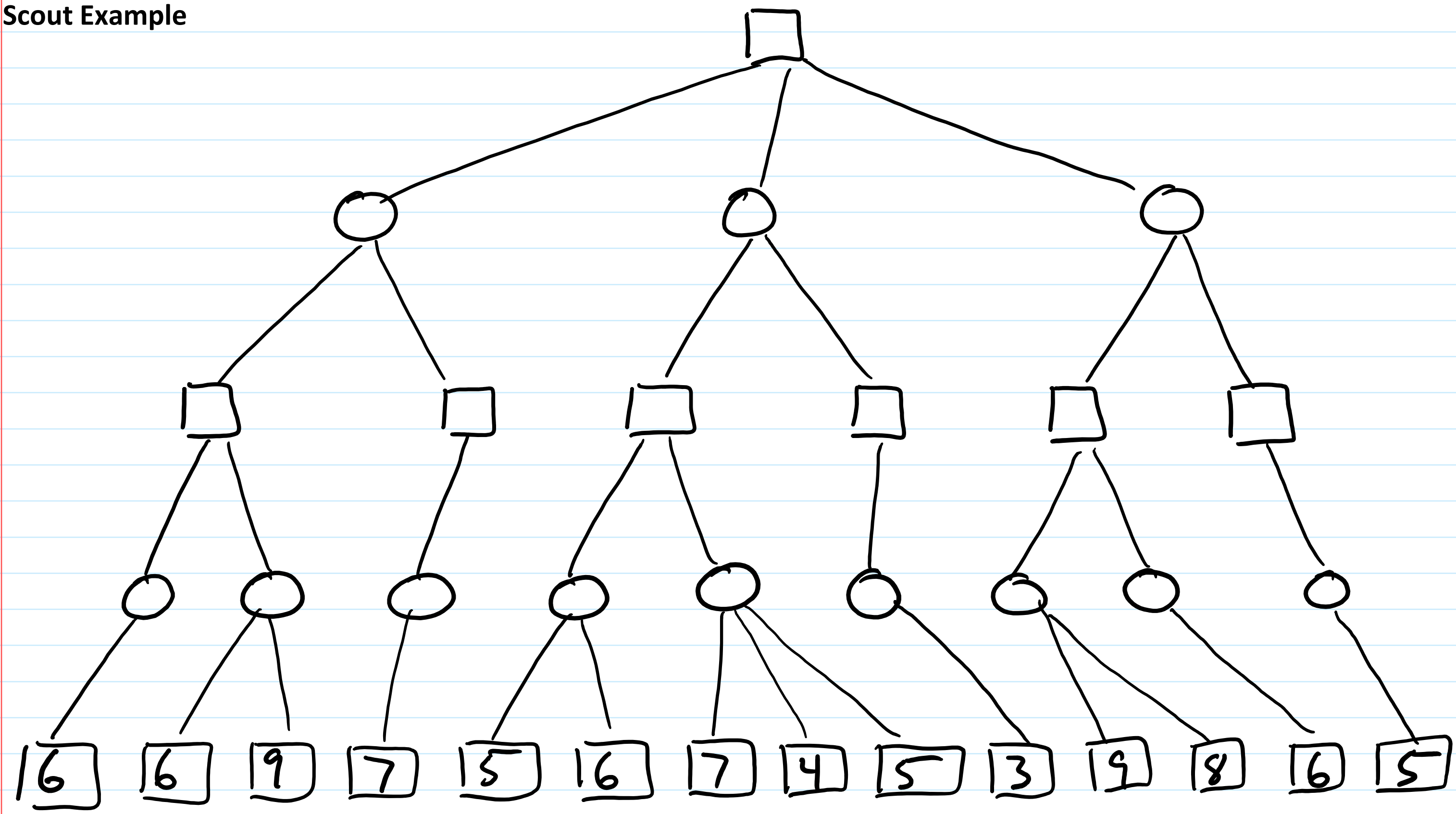
α = max(α, Alpha-Beta(p', α, β, depth − 1))

return α

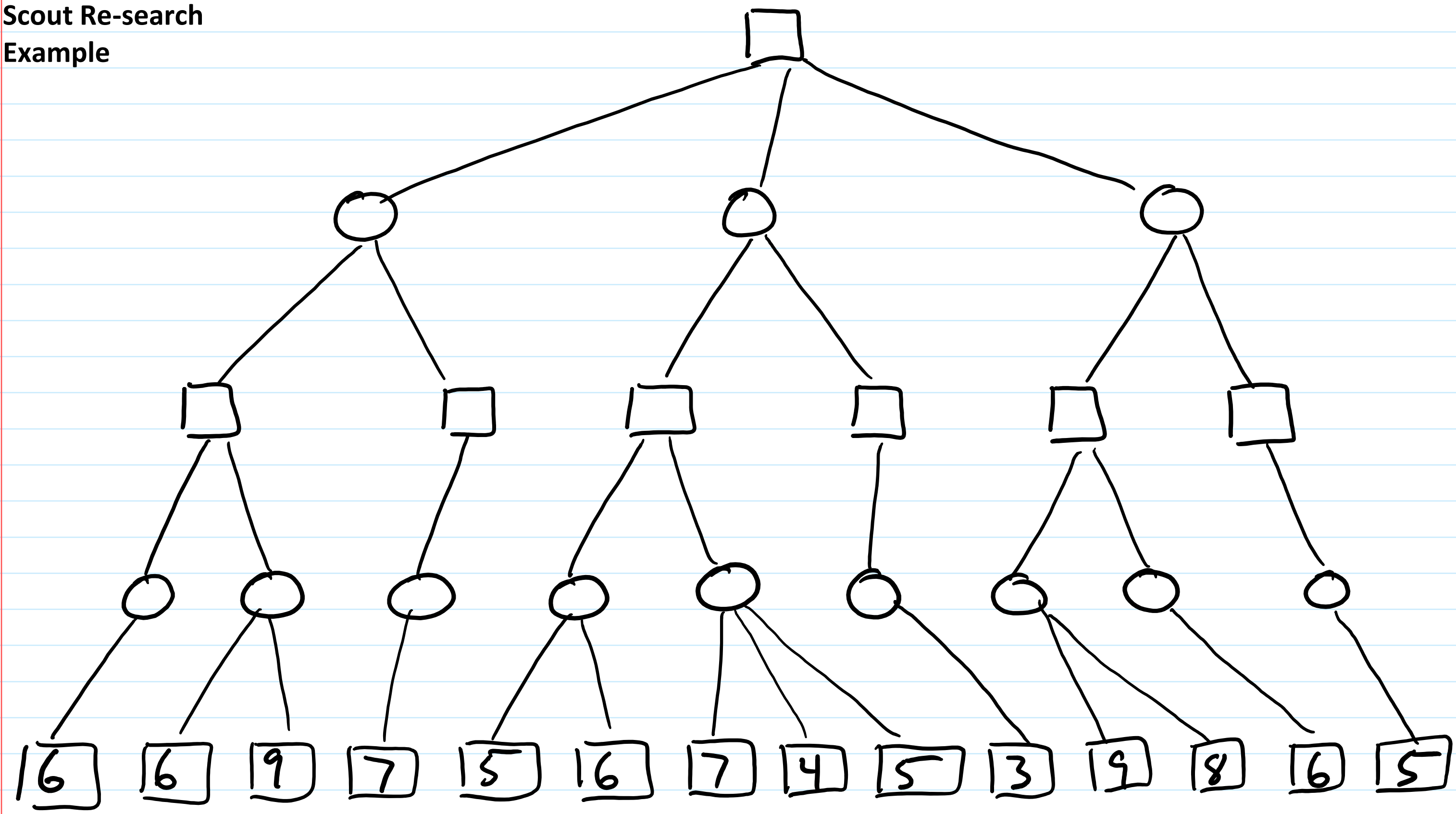
else

⋮

Scout Example

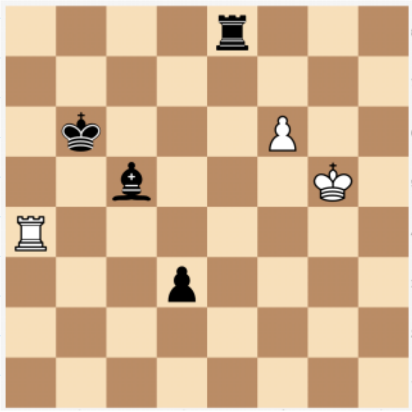


Scout Re-search Example



Transposition Table

Positions may be reachable by multiple sequences of moves



Keep table of values for all positions examined in tree

Keys:

Values:

Add check at start of A-B

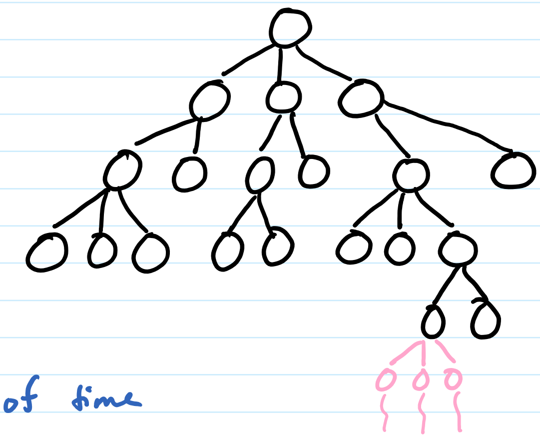
Save returned values in table

MTD-f

```
MTD-f (n, f, d)

    lowerBound ←
    upperBound ←
    g ←

    while lowerBound < upperBound

        β ←

        g ←
        if
        else

    return g
```

#

Monte Carlo Tree Search



search tree: nodes = positions
branches = moves
children = resulting pos

Until out of time

traverse tree   root ⇝ leaf → new node   tree policy

play from new node   default policy   random heuristic (fast)

update statistics   long games hurt

\# times played
total score

Advantages: always have move ready

no domain knowledge except rules

Multi-Armed Bandit

Given unknown probability distributions $R_1, \ldots, R_K$
with means $\mu_1, \ldots, \mu_K$

Choose indices $i_1, i_2, \ldots$ to optimize payout

Regret =

$\rho_T =$

"optimal" means

Ex:

Arm 1 prob	Arm 1 payout	Arm 2 prob	Arm 2 payout	Arm 3 prob	Arm 3 payout
$\frac{1}{3}$	2	$\frac{1}{4}$	3	$\frac{1}{100}$	200
$\frac{2}{3}$	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{99}{100}$	0
		$\frac{1}{2}$	0		

uniform rotation:

$$\lim_{T\to\infty} \frac{\rho_T}{T} =$$

greedy :

ε-greedy : play each once, then play random w/prob ε, arm with best avg reward so far otherwise zero regret

Choose arm j that maximizes $\bar{r}_j + \sqrt{\frac{2 \ln T}{n_j}}$