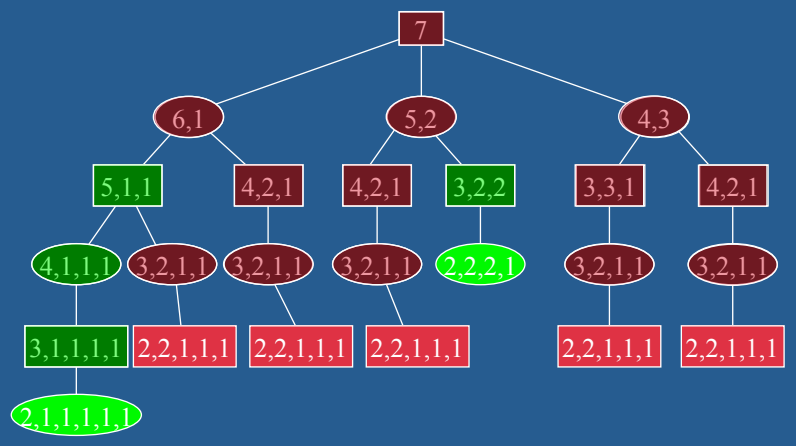


ADVERSARIAL SEARCH

Based on RN ch. 5

EECS 492
1 Feb 11

Two-Player Games

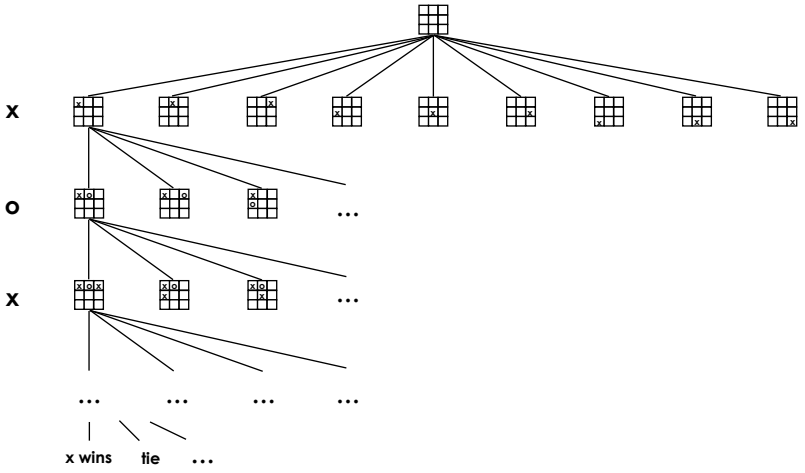
- A special category of multiagent Environment
- One other agent
- Other agent is adversarial
  - objectives are exactly opposed to the primary agent
  - (zero-sum game)
- Two agents move in alternating turns

Game as Search Problem

- States (as usual)
  - E.g., board position
  - Player to move
- Actions: legal moves in state
- Result function:
  - State resulting from choosing given (legal) move in given state
- Terminal-test: determines when game over
- Utility (payoff) function: assigns numeric values to terminal states

These together define the game tree (analogous to state-space graph)

A Game Tree



Winning Strategies

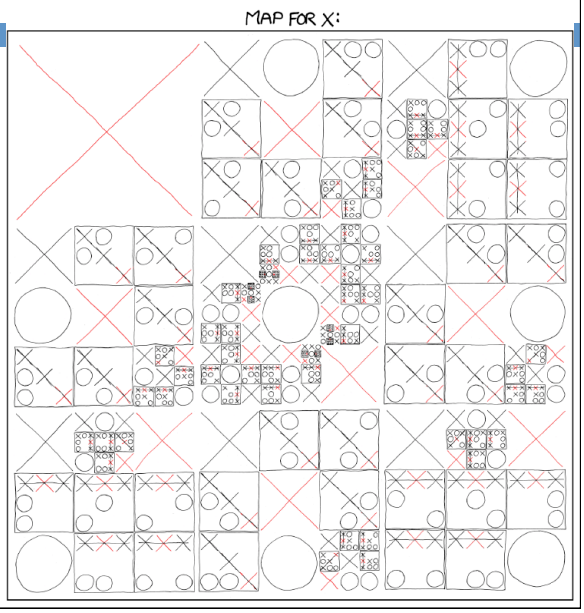
- A winning strategy for tic-tac-toe:
  - A move1 for X, such that
    - for any move1 for O, there is
      - a move2 for X, such that
        - for any move2 for O, there is

        …

        such that X wins.

Game Tree Strategies

- mapping from own nonterminal nodes to actions
- what to do in any (reachable) situation

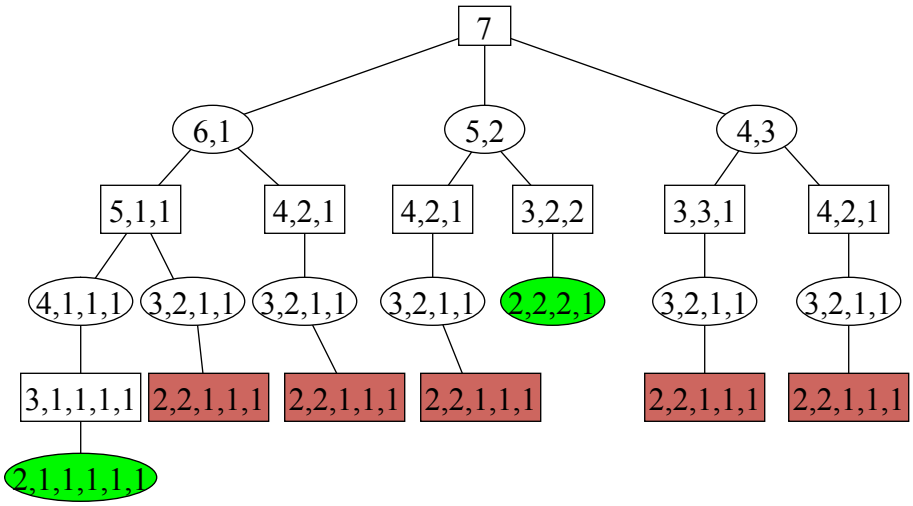


xkcd 832

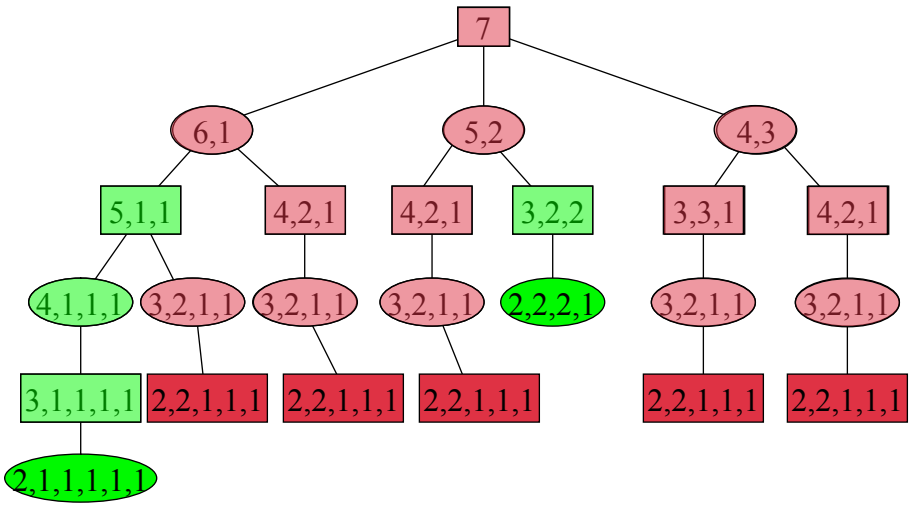
"Grundy's Game"

- Two players, one or more piles of pennies.
- Players take turns splitting one of the piles into two of unequal size.
- Piles of one or two cannot be split.
- Winner is last player to successfully split a pile.

Grundy, n = 7



Propagating Game Values



Minimax

- First player MAX
- Second player MIN
- Label leaves with utility of result
  (e.g., win = 1, lose = –1)
- Minimax algorithm
  - Traverse depth-first to label nodes with minimax value
  - MAX or MIN nodes, depending on whose turn

Minimax Values

- Leaf node

  Utility of result of game (e.g., 1 for win)
- MAX node

  Max of minimax value of successors
- MIN node

  Min of minimax value of successors

Minimax Algorithm

```
function Minimax-decision(state) returns an action
   inputs: current state in game
   v ← Max-value(state)
   return the action in Actions(state) with value v
-----------------------------------------------------
function Max-Value (state) returns a utility value
    if terminal-test(state) then return payoff(state)
     v ← – infinity
    for a in Actions(state) do
            v ← Max(v, Min-value(Result(s,a)))
    return v
-----------------------------------------------------
function Min-Value …
    analogous
```

Negamax Trick

Rather than MIN, just treat opponent as MAX with negated score.

```
function Minimax-decision(state) returns an action
   v ← Negamax-value(state)
   return the action in Actions(state) with value v
-----------------------------------------------------
Function Negamax-Value (state) returns a utility value
    if terminal-test(state) then return payoff(state)
     v ← – infinity
    for a in Actions(state) do
            v ← Max(v, – Negamax-value(Result(s,a)))
    return v
```

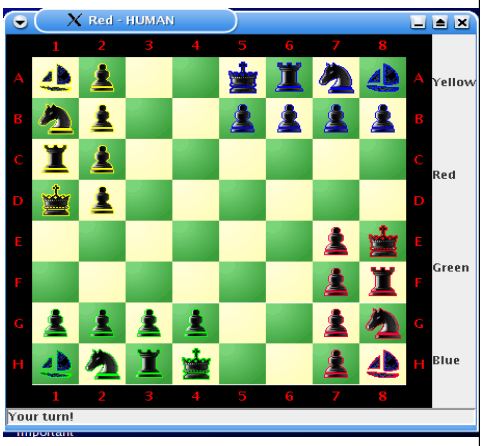
Search Complexity

- Time: $O(b^m)$
  - b branching factor, m max depth
- For chess, $b = 35$ and $m = 100$
- $35^{100} \approx 10^{154}$
- Suppose we had 10^{100} processors, each exploring 10^{18} nodes/second

Multi-Player Games

Max^N algorithm

- conceptually similar to Minimax
- value of terminal node: an *N*-vector
- value of intermediate node: that of best successor for currently-moving player



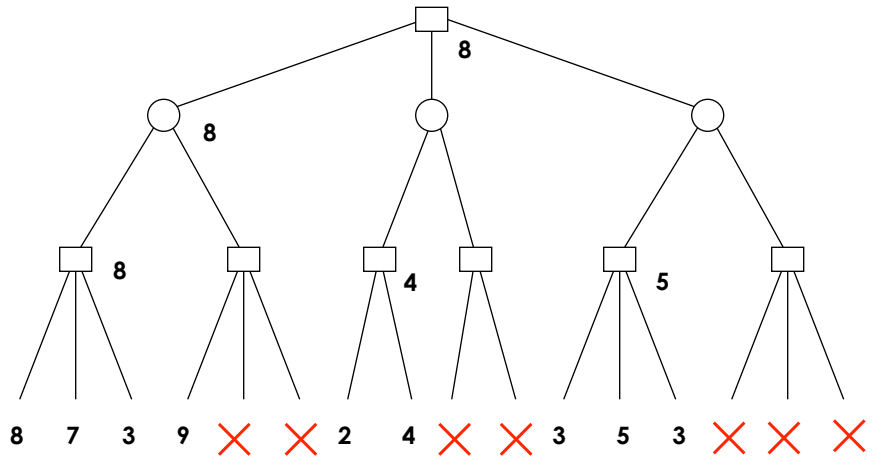
Static Evaluation

- Provide direct evaluation of intermediate game states
- Should represent *expected* utility, e.g., probability of winning
- Typically, searching deeper yields better evaluations
- But horizon and other effects suggest searching to *quiescence*

Quiescence Search

- Horizon effect
  - Searching to fixed depth misses important moves just past stopping point
  - Paths may look good because they delay inevitable disaster
- Remedy
  - Continue searching until no more "violent" (big impact) moves possible
  - Must be selective to avoid blowup (e.g., just search captures)

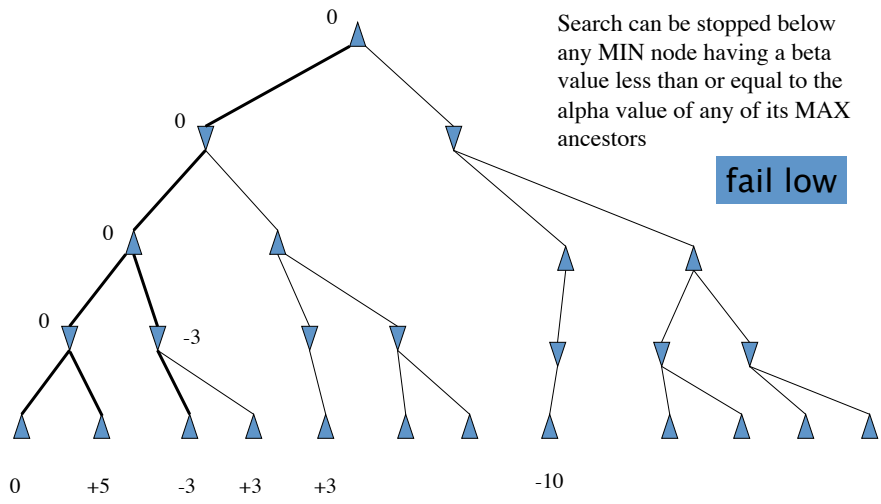
Pruning Game Trees



Alpha-Beta Search

- Maintain at each node *n*:
  - α: value of best choice found so far for any MAX node on path to *n*
  - β: value of best choice found so far for any MIN node on path to *n*
- During minimax search, prune subtree (terminate recursive call) whenever worse than current α or β value.

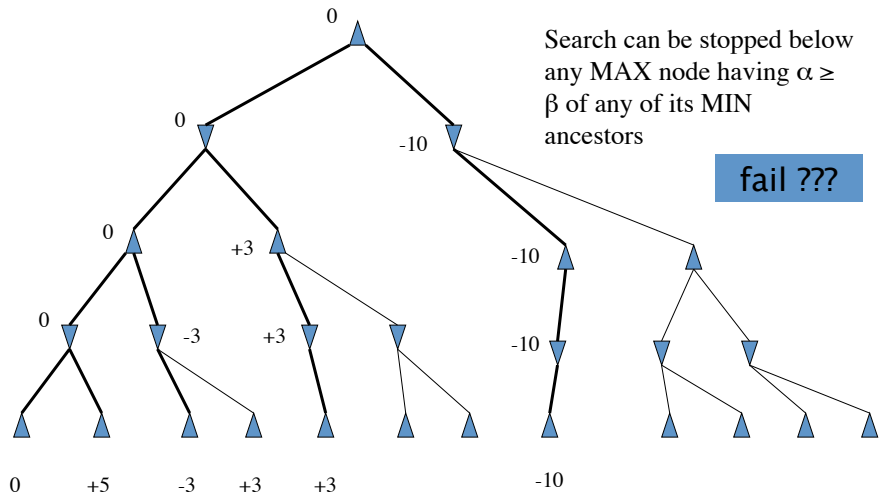
Example



Search can be stopped below any MIN node having a beta value less than or equal to the alpha value of any of its MAX ancestors

fail low

Example



Search can be stopped below any MAX node having $\alpha \geq \beta$ of any of its MIN ancestors

fail ???

α–β Pruning Effectiveness

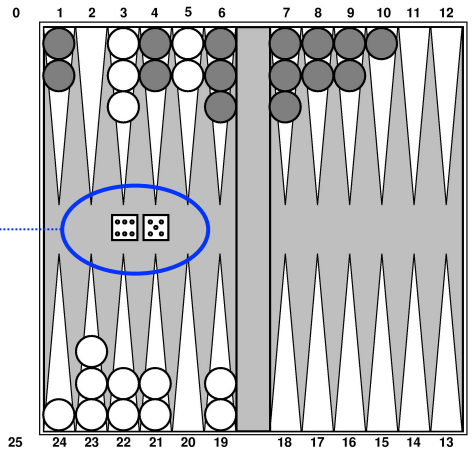
- Order of generating successors matters
- In worst-case, still $O(b^m)$
  - (how?)
- Best case:
  - examine successors in most favorable order
  - $O(b^{m/2})$
- Average case:
  - examine successors in random order
  - $O(b^{3m/4})$

Transposition Table

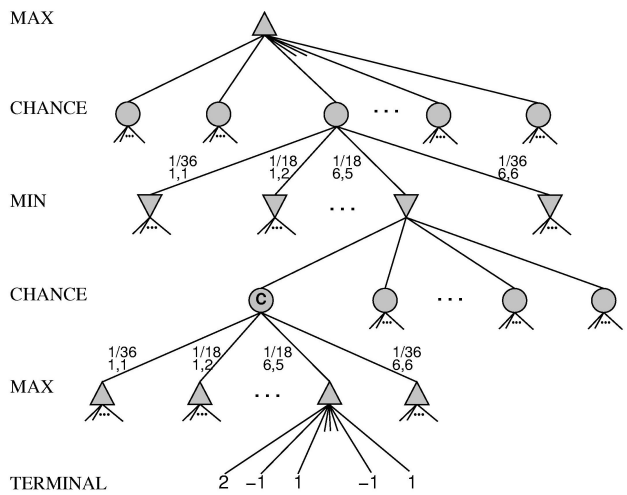
- Hash table of previously seen positions, with cached values (or bounds)
- Avoids repeating search on multiple branches of search tree
- Can dramatically deepen search
- But expensive therefore must be used selectively

Games of Chance

- Uncertainty also relevant for games
- Can extend turn-playing model to include chance elements
- E.g., backgammon

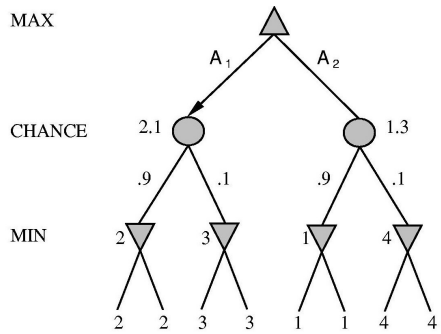


Backgammon Tree Fragment



Can solve using expectiminimax algorithm

Expectiminimax



Expectiminimax(n) =

Payoff(n)	if n is terminal state
$\max_{s \text{ in Successors(n)}}$ Expectiminimax(s)	if n is MAX node
$\min_{s \text{ in Successors(n)}}$ Expectiminimax(s)	if n is a MIN node
$\sum_{s \text{ in Successors(n)}}$ Pr(s) Expectiminimax(s)	if n is CHANCE node

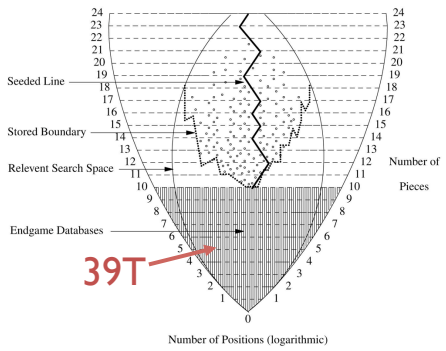
State-of-Art (Chess)

- Deep Blue defeats world champion (Kasparov), May 1997
- Deep Fritz defeats world champion (Kramnik), Nov 2006
  - Multiprocessor version of commercial program Fritz
- Today: PC chess programs beat ~~almost~~ everybody
- Rybka: current World Computer Chess champion
  - Off-the-shelf 8-core 3.2 GHz processor

Checkers is *Solved*

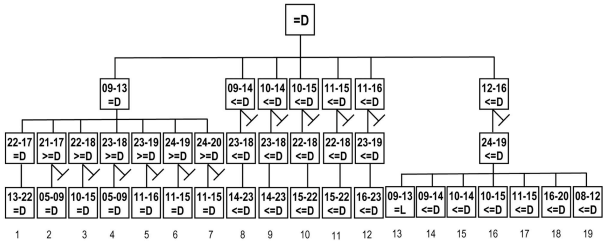
- Announced in *Science*, July 2007
  - Schaeffer et al., U Alberta
- State space > 10^{20}: Largest game solved to date
- Massive parallel search
  - Endgame databases computed 1989–2005
  - Forward search 2005–2007
  - 50+ dedicated processors over much of search time
- 18 years: Longest successful computation?

Search Process



Forward search to endgame databases

- Solve selected 3-move openings, construct overall proof from these
- Proof-number search, using Chinook evaluations, iterating on error tolerances



Next Challenge: JEOPARDY!

- IBM's WATSON to compete this month against human champions

Ferruci et al., *AI Magazine* 2010

Watson

- Demonstration of IBM's DeepQA technology
  - massive parallelism
    - consider multiple interpretations and hypotheses
  - many experts
    - contextual evaluation and integration of loosely coupled probabilistic question and content analytics
  - pervasive confidence estimation
    - every result associated with degrees of confidence
  - integrate shallow and deep knowledge
    - balance strict and shallow semantics, loosely formed ontologies