2 handouts: slides, asst 1 solution
asst 1 due
Adversarial Search
Adversarial Search

- Another Type
- Minimax
- Tic-tac-toe
- Improvements
- Break
- α-β Pruning
- α-β Pseudo-code
- Why α-β?
- Progress
- EOLQs

Adversarial Search
Another Twist on Search

- Shortest-path (M&C, vacuum, tile puzzle)
  - want least-cost path to goal at unknown depth
- Constraint satisfaction (map coloring, \(n\)-queens)
  - any goal that satisfies constraints (fixed depth)
- Combinatorial optimization (TSP, max-CSP)
  - want least-cost goal (fixed depth)
- Decisions with an adversary (chess, tic-tac-toe)
  - adversary might prevent path to best goal
  - want best assured outcome
Each ply corresponds to half a move. Terminal states are labeled with value. Can also bound depth and use a static evaluation function on non-terminal states.
A 3-length is a complete row, column, or diagonal.

value of position = $\infty$ if win for me,

or = $-\infty$ if a win for you,

otherwise = $\# 3$-lengths open for me $-$

$\# 3$-lengths open for you
Tic-tac-toe: two-ply search

Adversarial Search
- Another Type
- Minimax
- Tic-tac-toe
- Improvements
- Break
- α-β Pruning
- α-β Pseudo-code
- Why α-β?
- Progress
- EOLQs

Fig. 3.8 Minimax applied to tic-tac-toe (stage 1).
Tic-tac-toe: second move

Fig. 3.9 Minimax applied to tic-tac-toe (stage 2).
Fig. 3.10 Minimax applied to tic-tac-toe (stage 3).
Improving the Search

- partial expansion, SEF
- symmetry (‘transposition tables’)
- search more ply as we have time (De Groot figure)
- avoid unnecessary evaluations
asst 1 was due
book
asst 2 (theorem prover) going out on Wed. parse simple CFG.

exams are during common exam time
have web access? a clicker?
Adversarial Search
- Another Type
- Minimax
- Tic-tac-toe
- Improvements

Break
- \( \alpha - \beta \) Pruning
- \( \alpha - \beta \) Pseudo-code
- Why \( \alpha - \beta \)?
- Progress
- EOLQs
\( \alpha \) best outcome Max can force at previous decision on this path (init to \(-\infty\))

\( \beta \) best outcome Min can force at previous decision on this path (init to \(\infty\))

\( \alpha \) and \( \beta \) values are copied down the tree (but not up). Minmax values are passed up the tree, as usual.
### Pseudo-code

<table>
<thead>
<tr>
<th>Adversarial Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ Another Type</td>
</tr>
<tr>
<td>■ Minimax</td>
</tr>
<tr>
<td>■ Tic-tac-toe</td>
</tr>
<tr>
<td>■ Improvements</td>
</tr>
<tr>
<td>■ Break</td>
</tr>
<tr>
<td>■ (\alpha-\beta) Pruning</td>
</tr>
</tbody>
</table>

#### \(\alpha-\beta\) Pseudo-code

**Max-value (state, \(\alpha, \beta\)):**
- when depth-cutoff (state), return \(\text{SEF}(\text{state})\)
- for each child of state
  - \(\alpha \leftarrow \max(\alpha, \text{Min-value (child, } \alpha, \beta))\)
  - when \(\alpha \geq \beta\), return \(\alpha\)
  - return \(\alpha\)

**Min-value (state, \(\alpha, \beta\)):**
- when depth-cutoff (state), return \(\text{SEF}(\text{state})\)
- for each child of state
  - \(\beta \leftarrow \min(\beta, \text{Max-value (child, } \alpha, \beta))\)
  - when \(\beta \leq \alpha\), return \(\beta\)
  - return \(\beta\)
Fig. 3.12 An example illustrating the alpha-beta search procedure.
## Why \( \alpha - \beta \)?

<table>
<thead>
<tr>
<th>Adversarial Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Another Type</td>
</tr>
<tr>
<td>Minimax</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
</tr>
<tr>
<td>Improvements</td>
</tr>
<tr>
<td>Break</td>
</tr>
<tr>
<td>( \alpha - \beta ) Pruning</td>
</tr>
<tr>
<td>( \alpha - \beta ) Pseudo-code</td>
</tr>
<tr>
<td>Why ( \alpha - \beta )?</td>
</tr>
</tbody>
</table>

Time complexity of \( \alpha - \beta \) is about \( O(b^{d/2}) \)
**Computers best:** chess, checkers, Othello, backgammon, Scrabble

**Computers competitive:** bridge, crosswords, poker, small Go

**Computers amateur:** full Go
Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

Thanks!