14-0: Overview

- Example games (board splitting, chess, Othello)
- Min/Max trees
- Alpha-Beta Pruning
- Evaluation Functions
- Stopping the Search
- Playing with chance

14-1: Two player games

- Board-Splitting Game
  - Two players, V & H
  - V splits the board vertically, selects one half
  - H splits the board horizontally, selects one half
  - V tries to maximize the final value, H tries to minimize the final value

<table>
<thead>
<tr>
<th>14</th>
<th>5</th>
<th>11</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

14-2: Two player games

- Board-Splitting Game
  - We assume that both players are rational (make the best possible move)
  - How can we determine who will win the game?

14-3: Two player games

- Board-Splitting Game
  - We assume that both players are rational (make the best possible move)
  - How can we determine who will win the game?
    - Examine all possible games!

14-4: Two player games
14-5: Two player games

14-6: Two player games

14-7: Two player games

- Game playing agent can do this to figure out which move to make
  - Examine all possible moves
  - Examine all possible responses to each move
  - ... all the way to the last move
  - Calculate the value of each move (assuming opponent plays perfectly)

14-8: Two-Player Games
• Initial state
• Successor Function
  • Just like other Searches
• Terminal Test
  • When is the game over?
• Utility Function
  • Only applies to terminal states
  • Chess: +1, 0, -1
  • Backgammon: 192 . . . -192

14-9: Minimax Algorithm

Max(node)
if terminal(node)
    return utility(node)
maxVal = MIN_VALUE
children = successors(node)
for child in children
    maxVal = max(maxVal, Min(child))
return maxVal

Min(node)
if terminal(node)
    return utility(node)
minVal = MAX_VALUE
children = successors(node)
for child in children
    minVal = min(minVal, Max(child))
return minVal

14-10: Minimax Algorithm

• Branching factor of \( b \), game length of \( d \) moves, what are the time and space requirements for Minimax?

14-11: Minimax Algorithm

• Branching factor of \( b \), game length of \( d \) moves, what are the time and space requirements for Minimax?
  • Time: \( O(b^d) \)
  • Space: \( O(d) \)
  • Not manageable for any real games – chess has an average \( b \) of 35, can’t search the entire tree
  • Need to make this more manageable

14-12: > 2 Player Games

• What if there are > 2 players?
  • We can use the same search tree:
    • Alternate between several players
    • Need a different evaluation function

14-13: > 2 Player Games

• Functions return a vector of utilities
  • One value for each player
• Each player tries to maximize their utility
• May or may not be zero-sum

14-14: > 2 Player Games