Game Playing
REVIEW - MiniMax

• Searching with minimax
  – We use depth first search to reach down to the leaves first
  – When we reach a leaf, we evaluate the current state by a utility function and then returns the result to the parent node.
  – If we are not in a leaf and have evaluated all children.
    • If it, from this state, is max move, then return the highest utility
    • If it is a min move return the lowest utility.
MiniMax

function MiniMax(State s, Event e, boolean isMax) {
  State s1 = updateState(s, e)
  if (isLeaf(s1))
    return eval(s1)
  if (isMax)
    highest = -∞
    foreach (Event e1 in maxmoves(s1))
      tmp = MiniMax(s1, e1, !isMax)
      if (tmp > highest)
        highest = tmp
        move = e1
  return highest, move
  else
    lowest = ∞
    foreach (Event e1 in minmoves(s1))
      tmp = MiniMax(s1, e1, !isMax)
      if (tmp < lowest)
        lowest = tmp
        move = e1
  return lowest, move
MiniMax

• Example of MiniMax – 1st version of NIM
  – The rules of NIM are as follows
    • We start with a number of sticks in a group
    • On each move, a player must divide a group into two smaller groups
    • Groups of one or two sticks can’t be divided
    • The last player to make a legal move wins
MiniMax

- NIM Search tree

```
6
/ \
1-5 2-4 3-3
/ \ / \ / \
1-1-4 1-2-3 2-2-2
/ \ / \ / \
1-1-1-3 1-1-2-2
/ \ / \
1-1-1-1-2
```

Max moves

Min moves

Max moves

Min moves
MiniMax

- NIM Search tree
MiniMax

• NIM Search tree

Max moves
Min moves
Max moves
Min moves
MiniMax

- NIM Search tree

```
    6
   / | \  
  1-5 2-4 3-3
 /    /    /  
1-1-4 1-2-3 2-2-2
 /    /    /  
1-1-1-3 1-1-2-2
```

Max moves

Min moves
MiniMax

• NIM Search tree

1-5  1-2-3  2-2-2
  |     |     |
1-1-4 1-1-2-2 6
  |     |     |
1-1-1-3  1-1-1-1-2

Max moves
Min moves
Max moves
Min moves
MiniMax

- NIM Search tree

Max moves

Min moves

Max moves

Min moves
MiniMax

- NIM Search tree

Max moves

Min moves

Max moves

Min moves
MiniMax

- NIM Search tree

Max moves

Min moves

Max moves

Min moves
MiniMax

- Example of MiniMax – 2nd version of NIM
  – The rules of NIM are as follows
    - We start with a number of sticks in a group (7)
    - On each move, a player may pick up 1, 2, or 3 sticks
    - The last player to make a legal move wins
What does the search tree look like?
Properties of Minimax

• Complete?
Properties of Minimax

• Complete?
  – … yes, provided following are finite:
    • Number of possible legal moves (generative breadth of tree)
    • “Length of game” (depth of tree) – more specifically?
Properties of Minimax

• Complete?
  – … yes, provided the tree is finite:
    • Number of possible legal moves (generative breadth of tree)
    • “Length of game” (depth of tree) – more specifically?

• Optimal?
Properties of Minimax

• Complete?
  – … yes, provided the tree is finite

• Optimal?
  – … yes, provided perfect info (evaluation function) and opponent is optimal!

• Time Complexity?
Properties of Minimax

• Complete?
  – ... yes, provided the tree is finite:

• Optimal?
  – ... yes, provided perfect info (evaluation function) and opponent is optimal!

• Time Complexity?
  – Depth of tree: $m$
  – Legal moves at each point: $b$
  – $O(b^m)$ – NB, $m \approx 100$, $b \approx 35$ for chess!

• Space Complexity?
Properties of Minimax

- Complete?
  - ... yes, provided the tree is finite:

- Optimal?
  - ... yes, provided **perfect info** (evaluation function) and opponent is optimal!

- Time Complexity?
  - Depth of tree: \( m \)
  - Legal moves at each point: \( b \)
  - \( O(b^m) \) – NB, \( m \approx 100, b \approx 35 \) for chess!

- Space Complexity? \( O(bm) \) – why?