Minimax 3.0

1. Set L to be a list containing the initial state.

2. Forever:

   a. Let x = first item on L. If it is the initial state and has a value, stop.

   b. If x has a value, then
      Retrieve x's parent, p.
      If p is a max node, set p's value
         to max(p's value, x's value).
      If p is a min node, set p's value
         to min(p's value, x's value).
      Remove x from L.

   Else if x is a leaf or is at level d, then
      Set x's value to the value of the position, e(x).

   Else if x is an interior node,

      Set x's value to either —∞ if it's a max node

         or +∞ if it's a min.

      Add x's children to the front of the list.

Recap of Minimax

Minimax v3.0 reflects the following ideas:

1. Label each internal state according to whose move it is, based on the value of each of its children.

2. Use depth-first search (to limit the space used).

3. Stop search at some depth, `d`, and use a static evaluation function, `e(x)`, to assign values to those states as if they were leaves (to limit the time used).

What problems can arise?

A Group Exercise

Design a static evaluation function $e(x)$ for states in the game of Tic-Tac-Toe.

You may assume that each state contains information about what is in each cell of the game board and about whose move it is.

Your goal: to design a function that would outperform any other group's function in head-to-head competition!

(So keep your work as private as you can...)

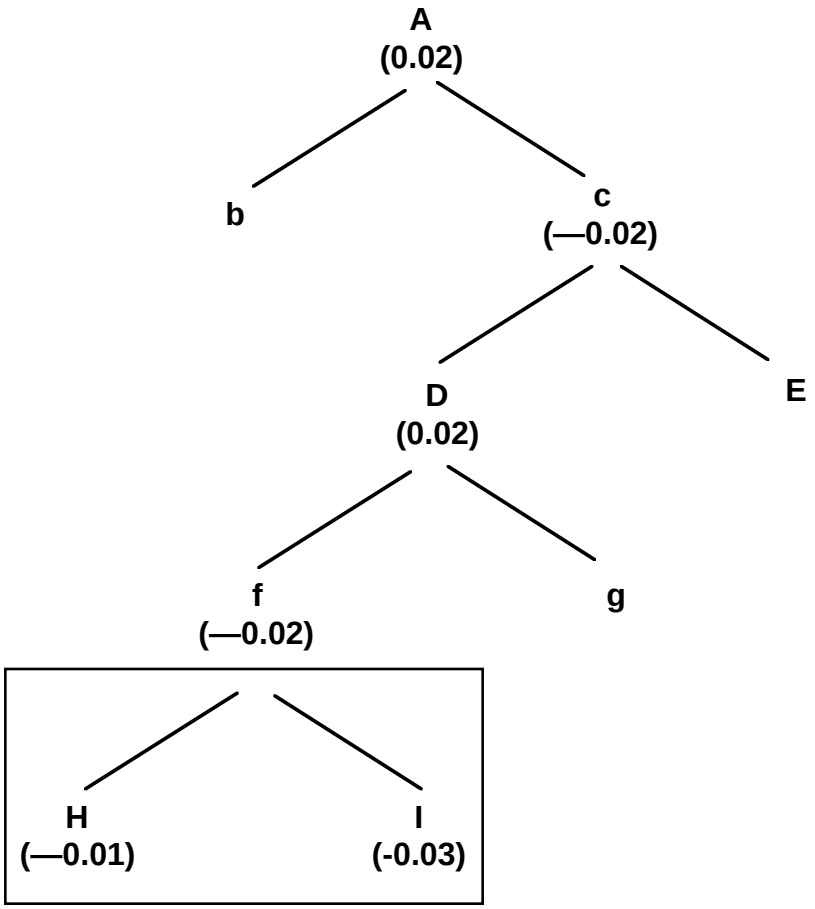
Problem 1, Solved

When the values assigned to a state and its children by `e(x)` are very different, that is,

```
abs( e(x) - e(x's parent) ) >> 0
```

the evaluation function is less likely to be giving an accurate picture of any state.

Solution: Search to "quiescence".



Problem 2, Unsolved

Sometimes, the value of a state in the world is about to change drastically — but at level `(d + 1)`.

This problem is called the “horizon effect”.

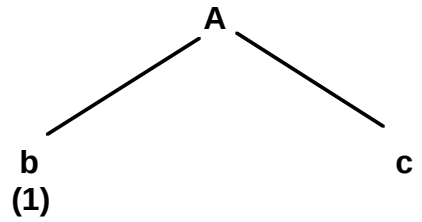
Solution? Do a quick search from the state or two that look like the best choices, to see if a better (or worse) state is coming soon.

Unfortunately, we have not found a general solution to the problem of the horizon effect.

Pruning the Search Space

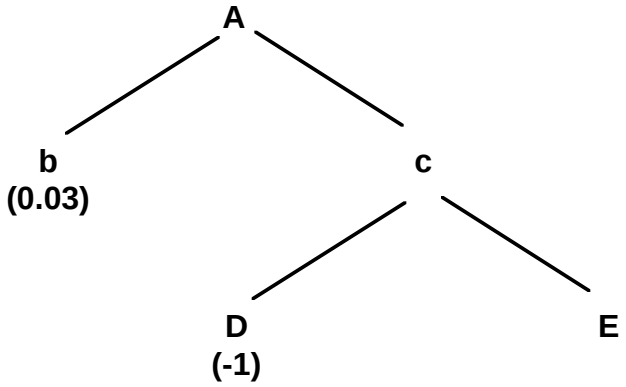
Besides, looking at **all of the states** at levels <= d is a waste of time. A program would be better served looking deeper into the tree, *but only at good moves*. But how?

Consider this situation in Minimax:



We don't need to consider state c to know the value of A.

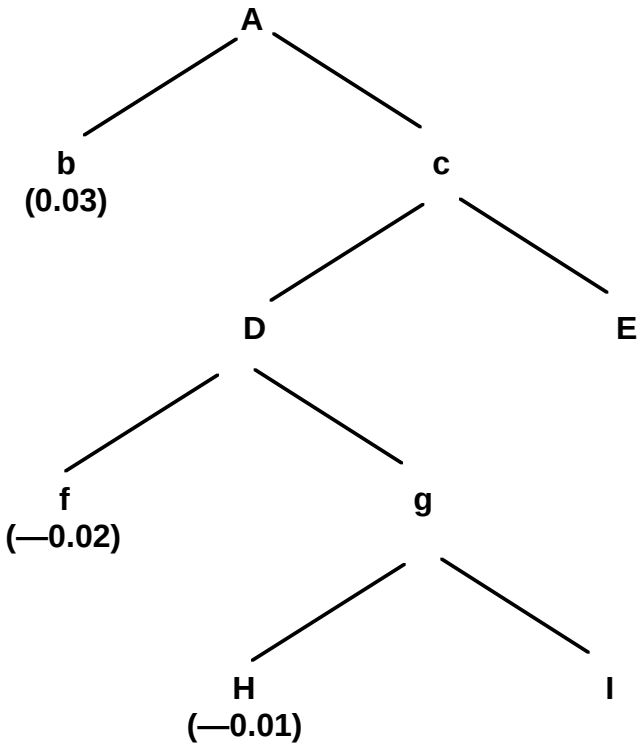
Or consider this situation:



We don't need to consider state E to know the value of A!

Pruning the Search Space: One More Example

Consider **this** situation in minimax:



We don't need to consider state I !!

These ideas are the motivation for **alpha-beta pruning**, a modification to Minimax that eliminates moves that we know aren't legitimate competitors.

Questioning Assumptions

As always, we need to keep in mind the **assumptions** that underlie our techniques. What are they? When do they apply?

Minimax, with or without alpha-beta pruning, assumes ...

a 2-player game with alternating moves

When, if ever, is the assumption violated?

Could we modify Minimax to handle a 3-player game?

Assumption 2

Minimax, with or without alpha-beta pruning, assumes ...

a world with perfect information

When, if ever, is the assumption violated?

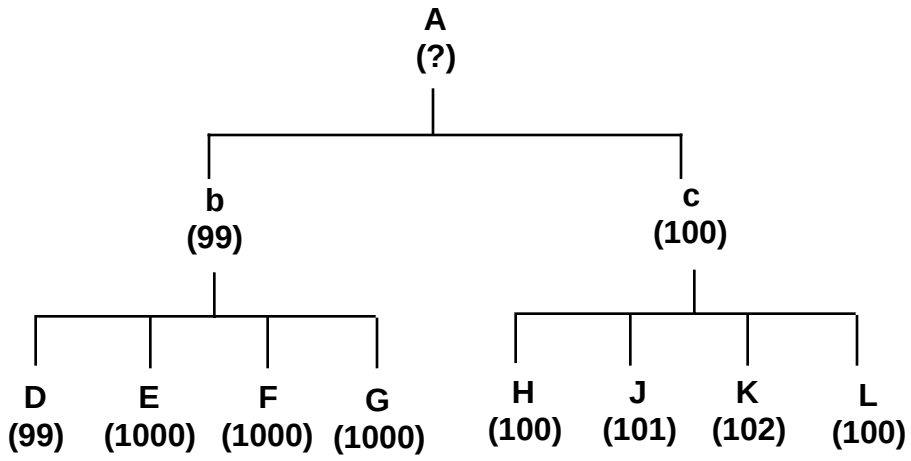
Could we modify Minimax to handle a world in which an agent has access only to partial information?

Questioning Assumptions

Minimax, with or without alpha-beta pruning, assumes ...

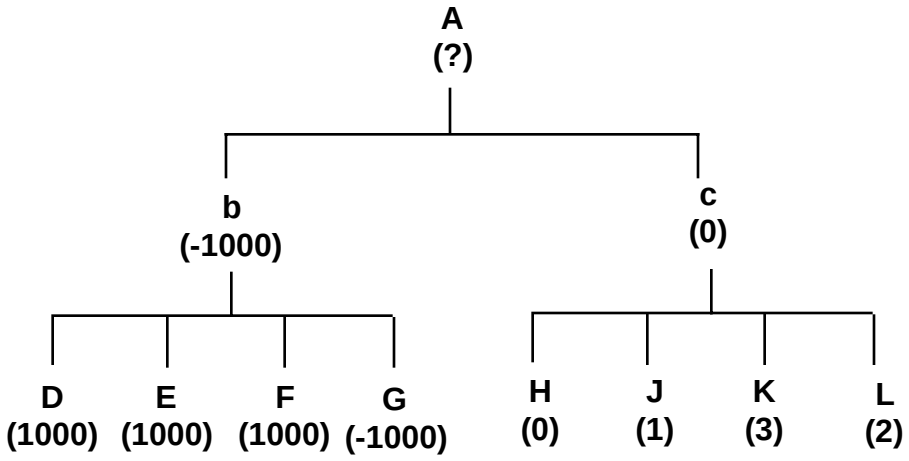
optimal play by the opponent

Consider the following position:



How could we take into account such situations in a Minimax search?

Another Vexing Minimax



Other Issues in Search

base-level reasoning vs. meta-level reasoning

“book” moves

a little history...

- chess
- checkers
- Othello
- backgammon
- Go
- bridge