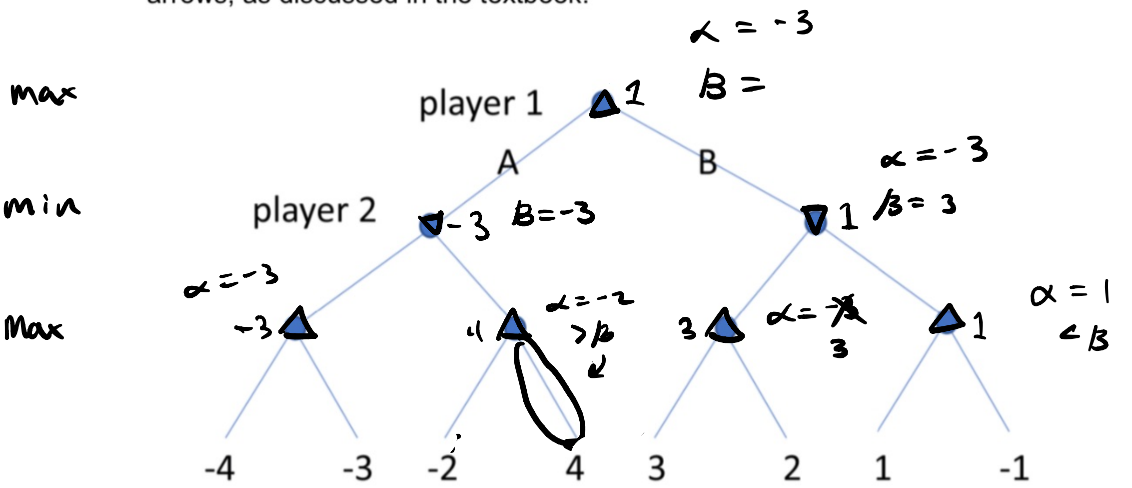
1. Given the simple game tree (binary, depth 3) below, label the nodes with up or down arrows, as discussed in the textbook.



Compute the *minimax* values at the internal nodes (write the values next each node).

Should the player 1 take action A or B at the root?

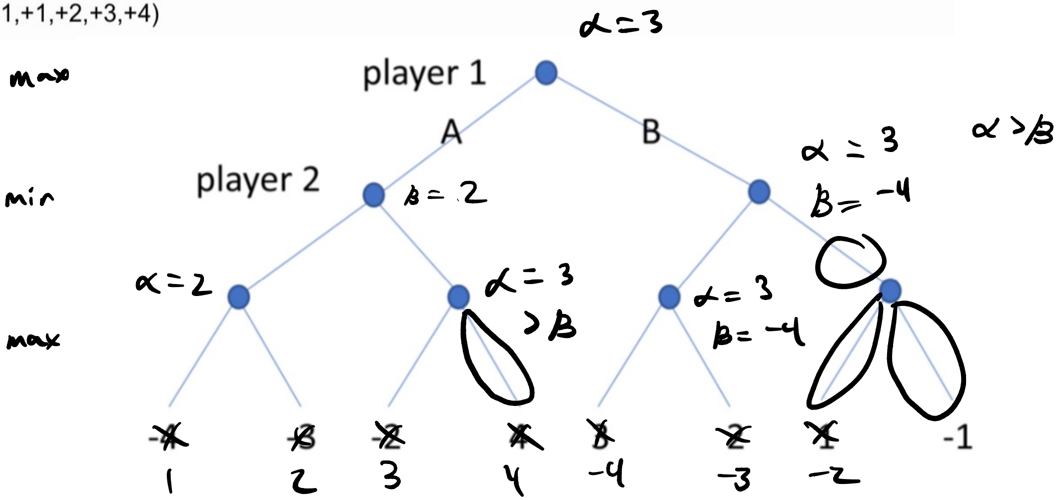
Action B

What is the expected outcome (payoff at the end of the game)?

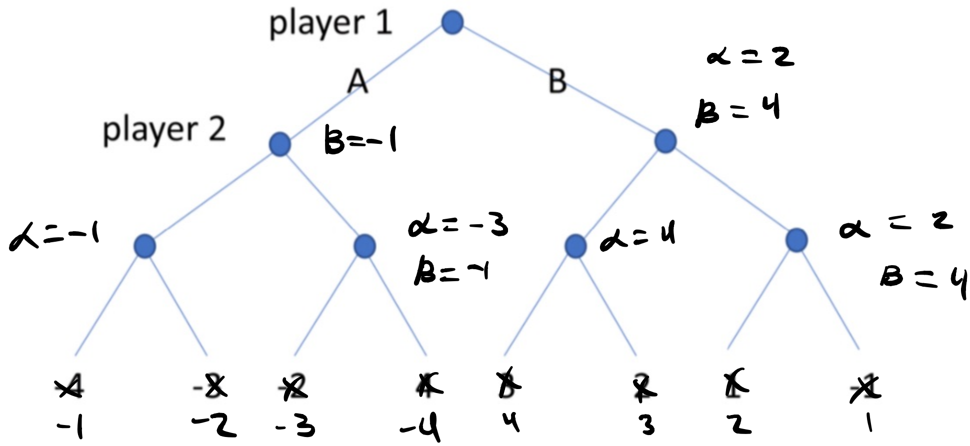
1

Which branches would be pruned by alpha-beta pruning? (circle them)

How could the leaves be relabeled to maximize the number of nodes pruned? (you can move the utilities around arbitrarily to other leaves, but you still have to use -4,-3,-2,-1,+1,+2,+3,+4)



How could the leaves be relabeled to eliminate pruning?

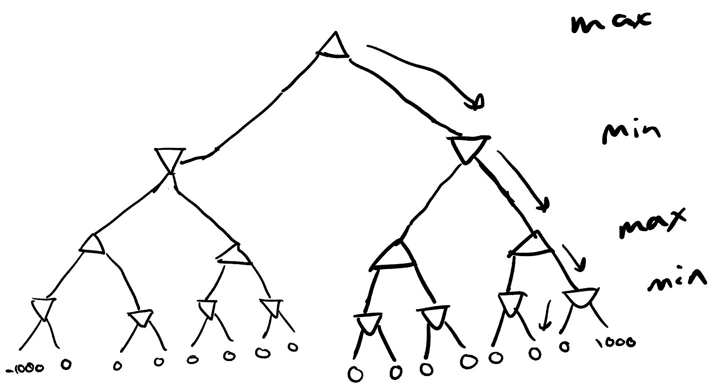


2. In a simple binary game tree of depth 4 (each player gets 2 moves), suppose all the leaves have utility 0 except one winning state (+1000) and one loosing state (-1000).

a) Could the player at the root force a win?

b) Does it matter where the 2 non-zero states are located in the tree? (e.g. adjacent or far apart)

c) If this question was changed to have a different depth, would it change the answers to the two questions above? If yes, how do the answers change? If no, explain why no change would happen.



a) No, the player at the root (Player A) cannot force a win. Player B has the last move in a tree of depth 4. There are 4 possible ending move possibilities:

1. 0 and 1000 as options
2. 0 and -1000 as options
3. 0 or 0 . . .
4. 1000 or -1000

Because player B will always try to minimize (-1000), there is no case for which player A can force a win.

b) In this case, no. A win cannot be forced for any combination of moves, only a draw because player A will never choose to make a move that will result in a loss, and there is only one win/lose state.

c) No. At any given choice, the minimum will never be 1000, because there is only one win state. Thus, only a draw can be forced, because the min/max <u>can</u> be 0. So, for a tree of depth n, the player moving at depth n-1 can only ever make a win impossible for the player at depth n, but cannot force a loss.

3. Hiking Philosophers. Three philosophers, Alex (A), Bob (B), and Charlie (C), are going on a hike and need to decide the order in which they will hike. Alex and Charlie have PhDs, while Bob has a MS degree. Adjacent hikers in the sequence have to have different degrees. Finally, Charlie does not want to be last.

a) Show how to set this up as a Constraint Satisfaction Problem. (what needs to be defined?)

vars = {Alex, Bob, Charlie} = {A, B, C}

domains: dom(s) = {1, 2, 3} → referring to hike order

D_i = Degree of hiker in position $i = 1, 2, 3$

i_x = Position of hiker x

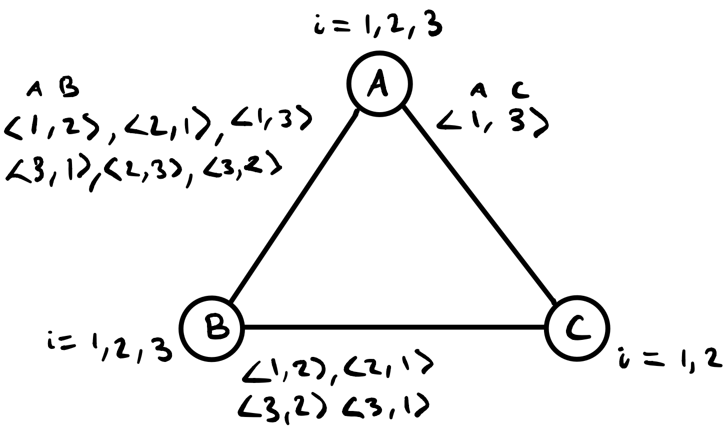
constraints:

1) $D_i \neq D_{i-1}$ and $D_i \neq D_{i+1}$

2) $i_C \neq 3$

3) $i_A \neq i_B \neq i_C$

b) Draw the Constraint Graph (label all nodes and edges)



c) Trace how plain Backtracking (BT) (with no heuristics) would solve this problem, assuming values are processed in *alphanumeric* order. Identify instances where back-tracking happens.

$D_A = PhD = D_C$, $D_B = MS$

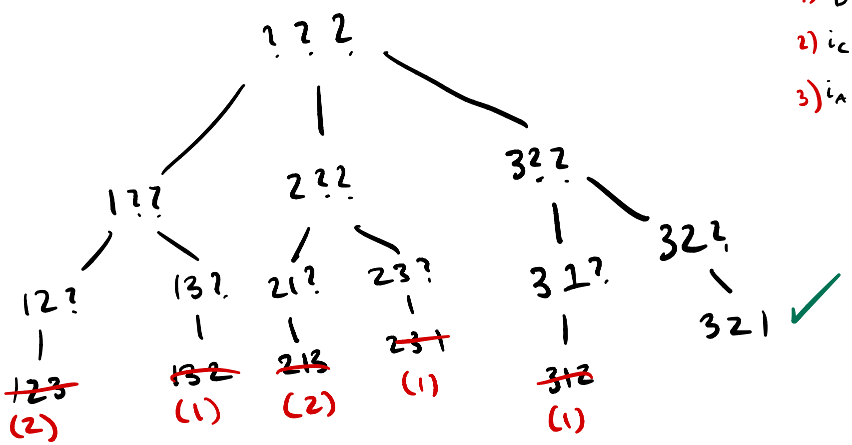
vars = $\{A, B, C\}$, state representation = $\langle i_A, i_B, i_C \rangle$

constraints:

1) $D_i \neq D_{i-1}$ and $D_i \neq D_{i+1}$

2) $i_C \neq 3$

3) $i_A \neq i_B \neq i_C$



Trace on next page

Trace

- Try $i_A = 1$
  - Try $i_B = 2$
    - Try $i_C = 3$
      - violates (2), backtrack
    - No choices for i_C
  - Try $i_B = 3$
    - Try $i_C = 2$
      - violates (1), backtrack
    - No choices for i_C
  - No choices for i_B
- Try $i_A = 2$
  - Try $i_B = 1$
    - Try $i_C = 3$
      - (2), backtrack
    - No choices for i_C
  - Try $i_B = 3$
    - Try $i_C = 1$
      - (1), backtrack
    - No choices for i_C
  - No choices for i_B
- Try $i_A = 3$
  - Try $i_B = 1$
    - Try $i_C = 2$
      - (1), backtrack
    - No choices for i_C
  - Try $i_B = 2$
    - Try $i_C = 1$
      - Solution reached

d) Trace how BT would solve this problem using the MRV heuristic.

constraints:

1) $D_i \neq D_{i-1}$ and $D_i \neq D_{i+1}$ $\Rightarrow$ This means $i_A, i_C \neq 2$

2) $i_C \neq 3$

State representation $= \langle i_A, i_B, i_C \rangle$

3) $i_A \neq i_B \neq i_C$

Try i_C first, as it is the most constrained

Try $i_C = 1$

Try $i_B = 2$

Try $i_A = 3$

solution reached ✓