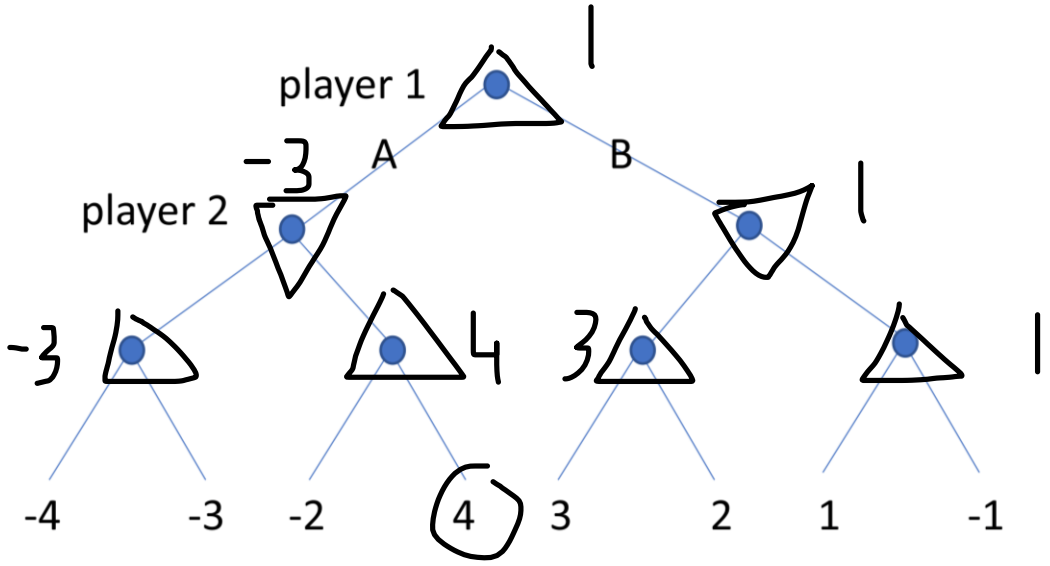
CSCE 420 - Spring 2023
Homework 1 (HW1)
due: Thurs, Feb 23, 5:00pm - Late written homeworks will not receive credit.

Turn-in answers as a Word document (HW1.docx or .pdf) and commit/push it to your class github repo.

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



Label the leaves with utility values for player 1 (who is at the root).

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?

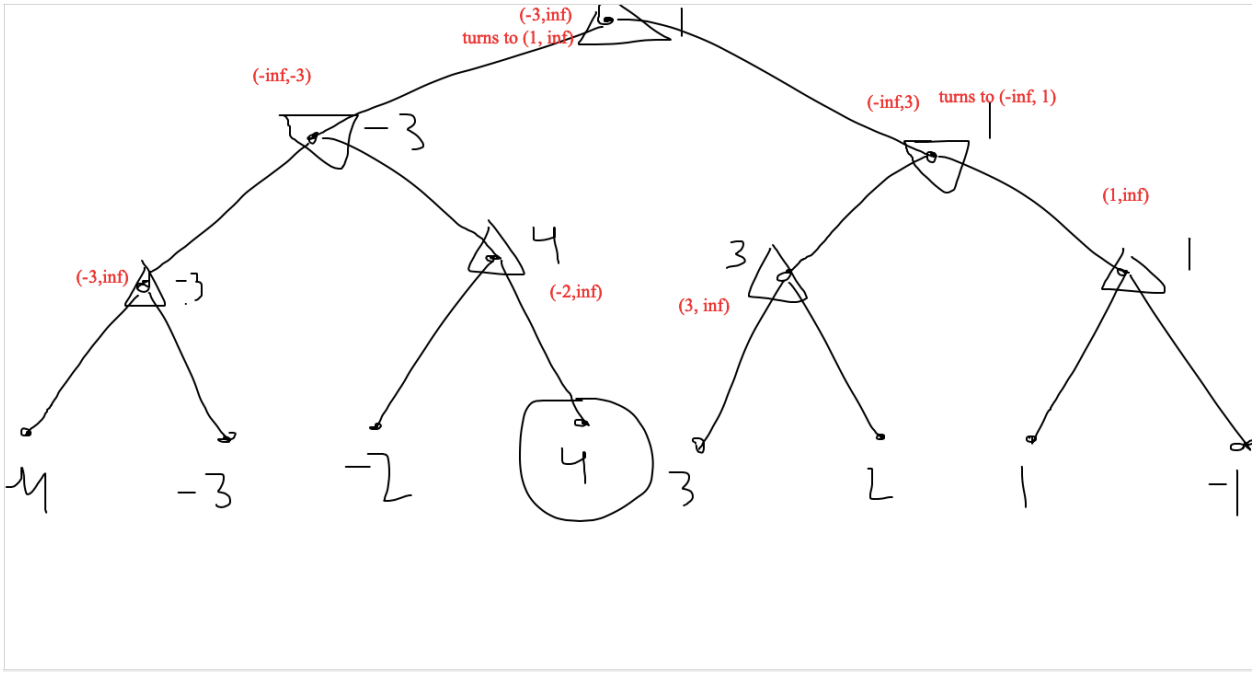
Player 1 should take action B because it leads to the state with the highest minimax value.

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

The outcome will be a minimax value of 1, where Player 1 wins the game.

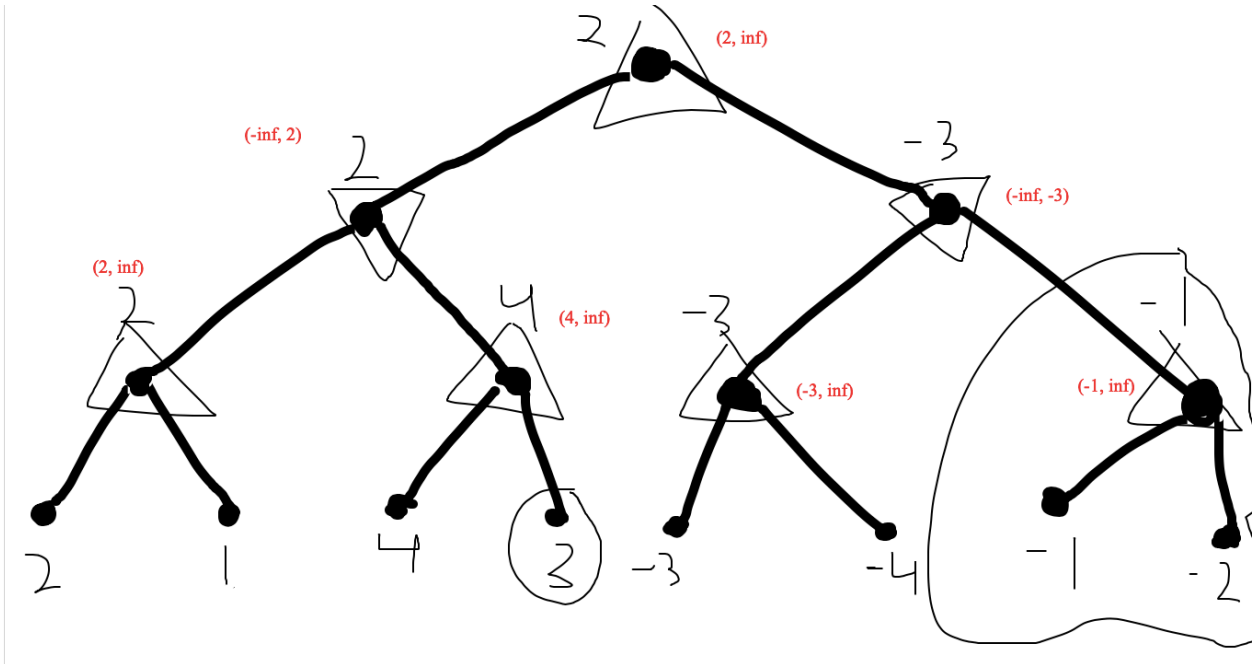
Which branches would be pruned by alpha-beta pruning? (circle them)

Circles are above, while work is shown below.



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)

The leaves could be relabeled to be in the order of 2, 1, 4, 3, -3. -4, -1, -2. Work shown in figure below.



How could the leaves be relabeled to eliminate pruning?

The leaves could be relabeled as 4, -2, -4, -3, 3, 2, 1, -1. This is because I switched the first and second leaves with the third and fourth leaves.

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).

- Could the player at the root force a win?

  No, it could not force a win because either at the first or second node away from the leaf that contains the winning state, the node will choose 0 depending whether it is a max or min node.

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

  When, it comes down to determining if the game is a draw (has minimax value of 0), it doesn't matter where the 2 non-zero states are located in because the nodes will end up picking 0.

- 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.

  No, because the interleaving of the min and max nodes would force a node to eventually choose a minimax value of 0.

3. Consider the task of creating a **crossword puzzle** (choosing words for a predefined layout). Suppose you have a finite dictionary of words (see /etc/dictionaries/ on linux distributions for ~50k English words; used for 'ispell' command).

Given a layout with a list of *n* "across" and *m* "down" words (see the example in the Figure below, the goal is to choose words to fill in (that satisfy all the intersections between words). (Clues for these words can then be defined later by a puzzle expert.)

Show how to model this crossword puzzle design problem as a CSP.

What are the variables, domains, and constraints? (hint: not all variables have to have the same domain)

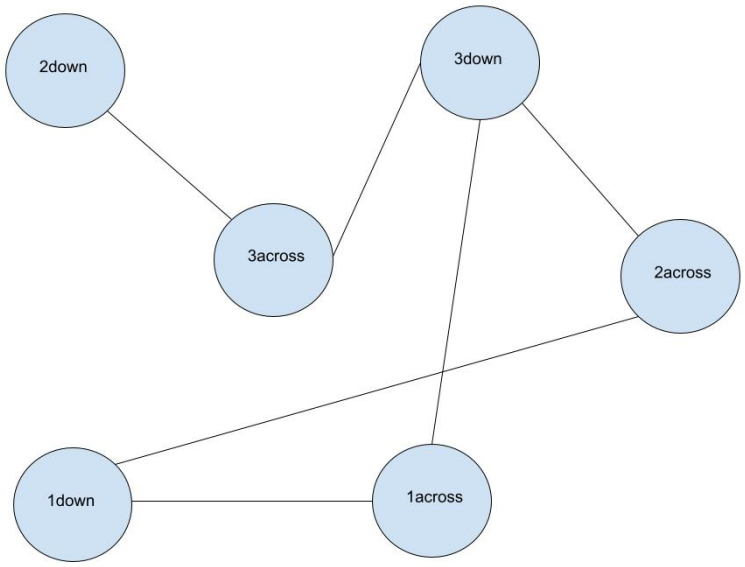
Variables: 1across, …, nacross, 1down, …, mdown

Domains: dom(var) = {finite dictionary of words}

Constraints:

1. Each variable must be unique
2. Intersections must contain the same letter for the corresponding character of the 2 words

Draw the constraint graph.
Label one of the nodes and one of the edges as examples (the nodes are easy; the edges require some thought). (of course, you don't have to write down the labels completely; just explain what they would look like and give a couple examples)



Nodes: 1across -> Domain: {all words with 7 letters), 1 down -> Domain: {all words with 3 letters)

Edges: The edges would be pairs of seven-letter and three-letter words where the third character of the first word and the first letter of the second word are the same -> **<**Abandon, All>, <Academy, Act>, <Admiral, Mug>, …

Describe the first couple of steps of how standard backtracking search would work (selecting variables and values in default order), making reasonable choices as you go. (e.g. you could state: suppose 'ace' is the first 3-letter word starting with 'a'...).

Let {w1, w2, …, w6} be the state of a node where wi = {finite dictionary of words}. The root node will attempt to pick a 7-letter word for 1across (w1). The next node level will attempt to pick a 3-letter word for 1down (w2). This level of nodes will be pruned if it violates the constraint of intersections not being the same letter or the constraint of unique words.

Describe how using the MRV would change the search; describe the first couple of steps again. (hint: you might need to make some assumptions about how many words have 3 letters, 4 letters, etc, or how many 5-letter words start with 'a' or 'y'...)

We begin with an empty {} state where start our backtracking recursion algorithm. Since the state/assignment is not complete, then we select an unassigned variable with least remaining choices. We should pick the 2-letter words because they have the fewest permutations. After that, pick the words with fewest remaining choices based intersections and word length.

```
Crossword layout:

                        1across (len=7)
_ _ _ _ _ _ _           2across (len=5)
    _       _           3across (len=6)
    _ _ _ _ _           1down (len=3)
            _           2down (len=2)
      _     _           3down (len=6)
  _ _ _ _ _ _
```

```
Example of a solution:

a b a l o n e
    c       a
    e a g e r
            b
      h     u
  a b a s e d
```