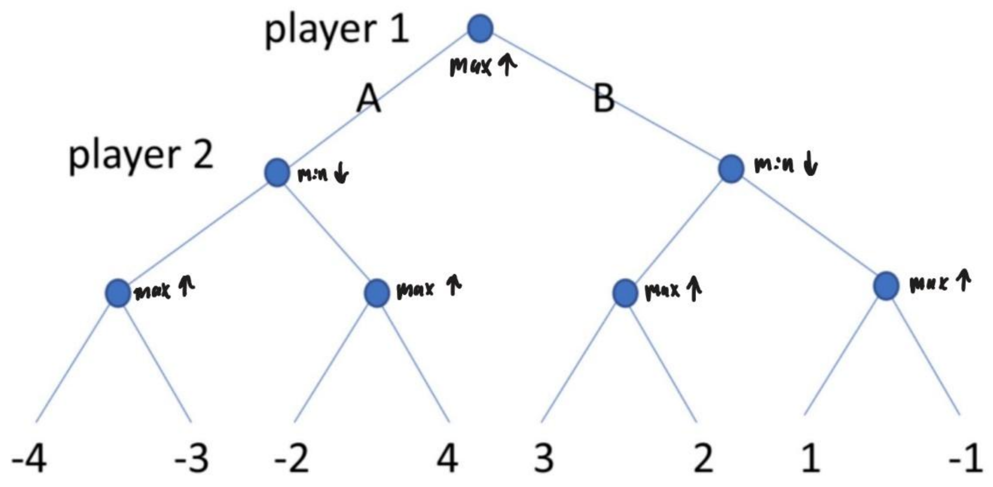
CSCE 420 - Spring 2023

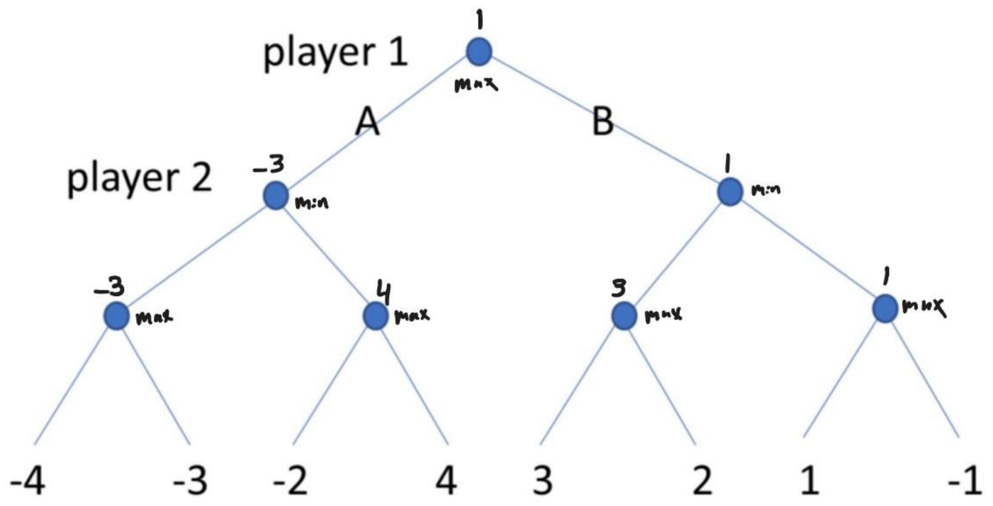
Homework 1

William Allen

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

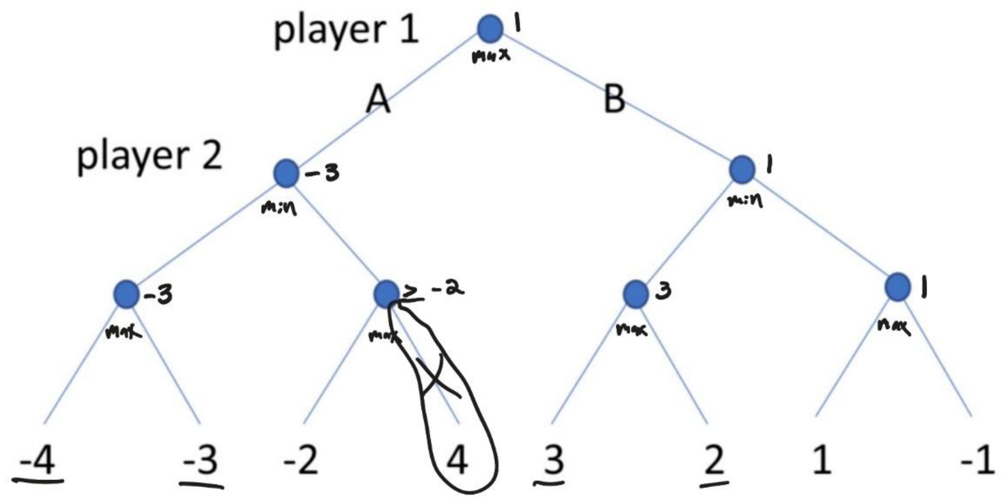


Compute the minimax values at the internal nodes (write the values next each node)



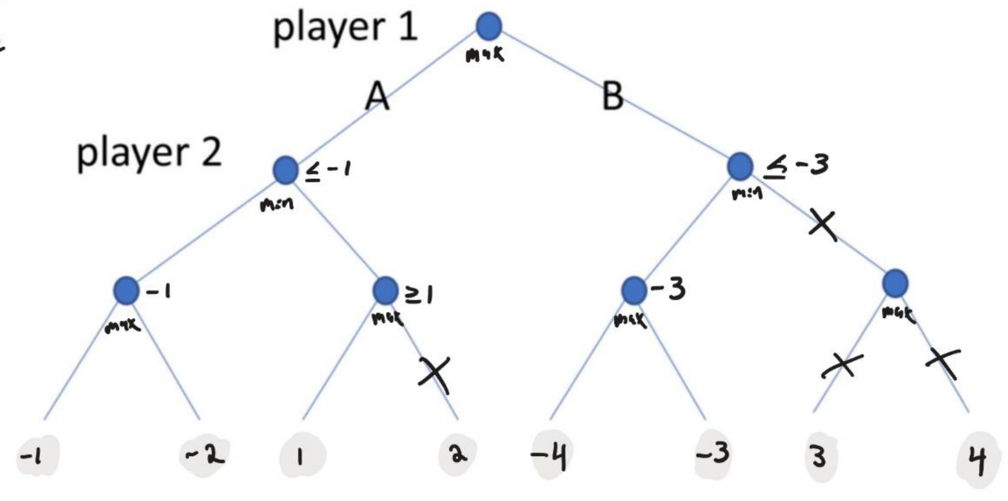
Should the player 1 take action A or B at the root?
Player 1 should take action B
What is the expected outcome (payoff at the end of the game)?
1

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

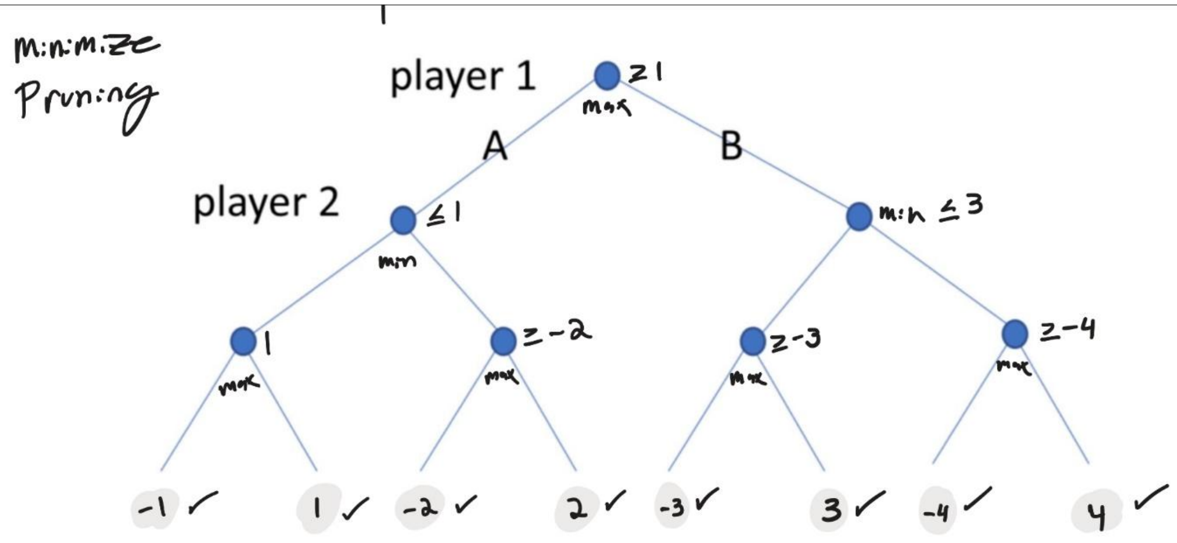


How could the leaves be relabeled to maximize the number of nodes pruned? (you can move the utilities around arbitrarily to other leaves, but you still have to use -4,-3,-2,-1,+1,+2,+3,+4)

maximize Pruning



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).
   • Could the player at the root force a win?
   • Does it matter where the 2 non-zero states are located in the tree? (e.g. adjacent or far apart)
   • 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.

   Given the stated conditions, the player at the root can never force a win. The reason is that once you introduce a min(a,b) on the depth layer 2 of the tree, the (+1000) win will be absorbed into a 0 or a (-1000) loss, if it is not already absorbed further down the tree. It does not matter where the winning and losing nodes are located. In the case that we can vary the depth, player 1 may force a win only if the tree depth is equal to 1 (the root node will immediately pick the winning value).

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?

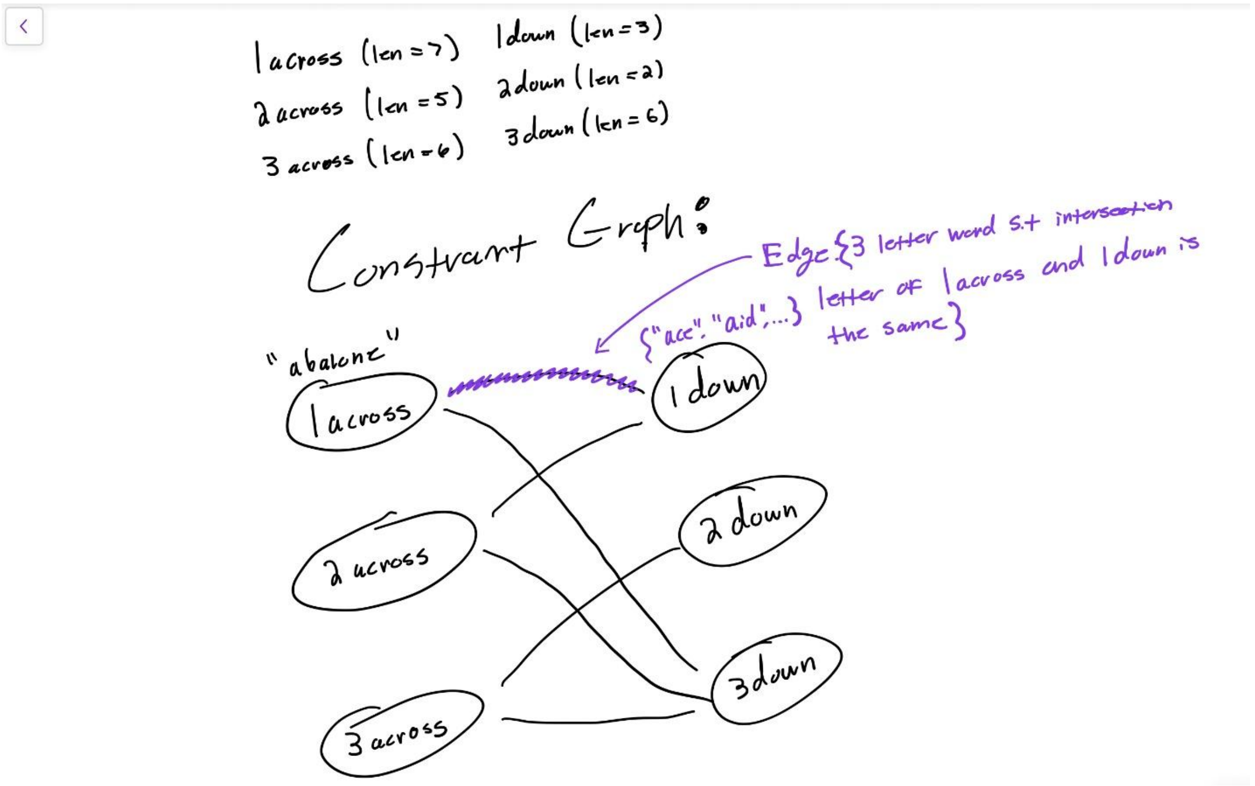
   Draw the constraint graph. Label one of the nodes and one of the edges as examples (the nodes are easy; the edges require some thought). (of course, you don't have to write down the labels completely; just explain what they would look like and give a couple examples) Describe the first couple of steps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

==Answer==
Variables: One variable for each word, which corresponds to a row or a column of the puzzle. The variable names are given by the clues (e.g. "1 across", "2 down").

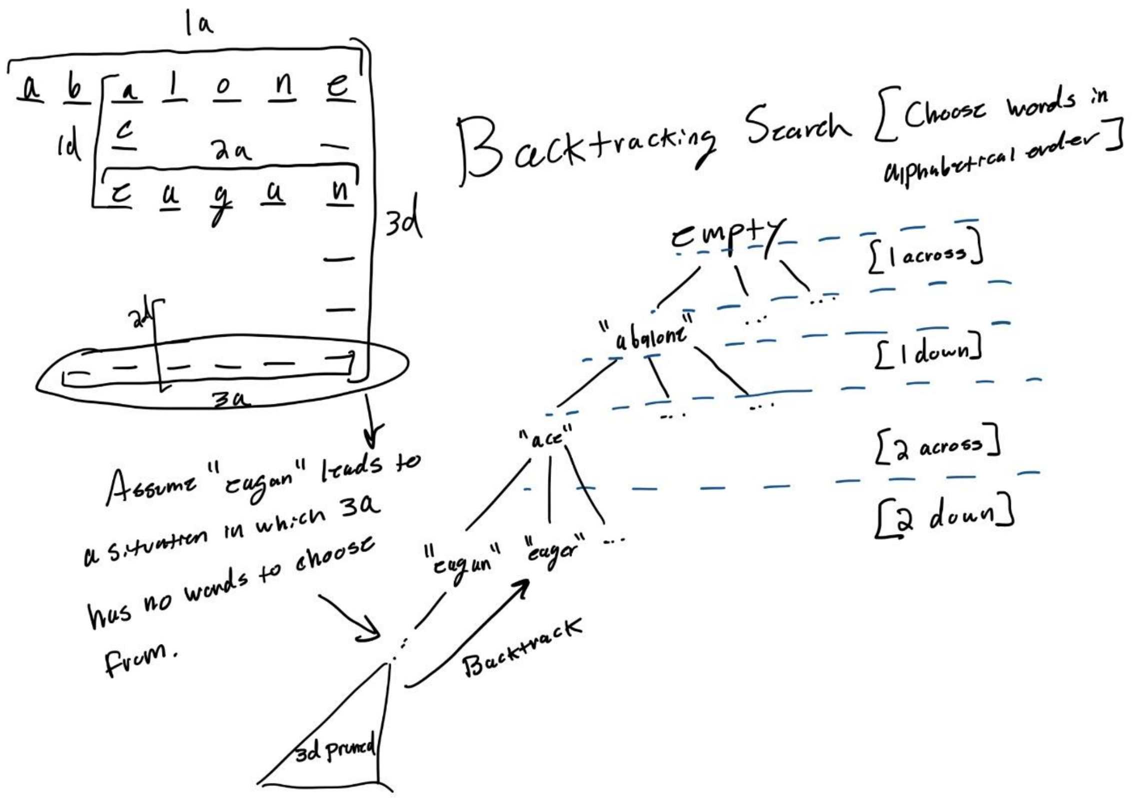
Domains: Each variable has a domain that consists of all the words in the dictionary that fit the length.

Constraints: For each pair of intersecting variables, the constraint specifies that the shared letters must match.

Constraint graph:

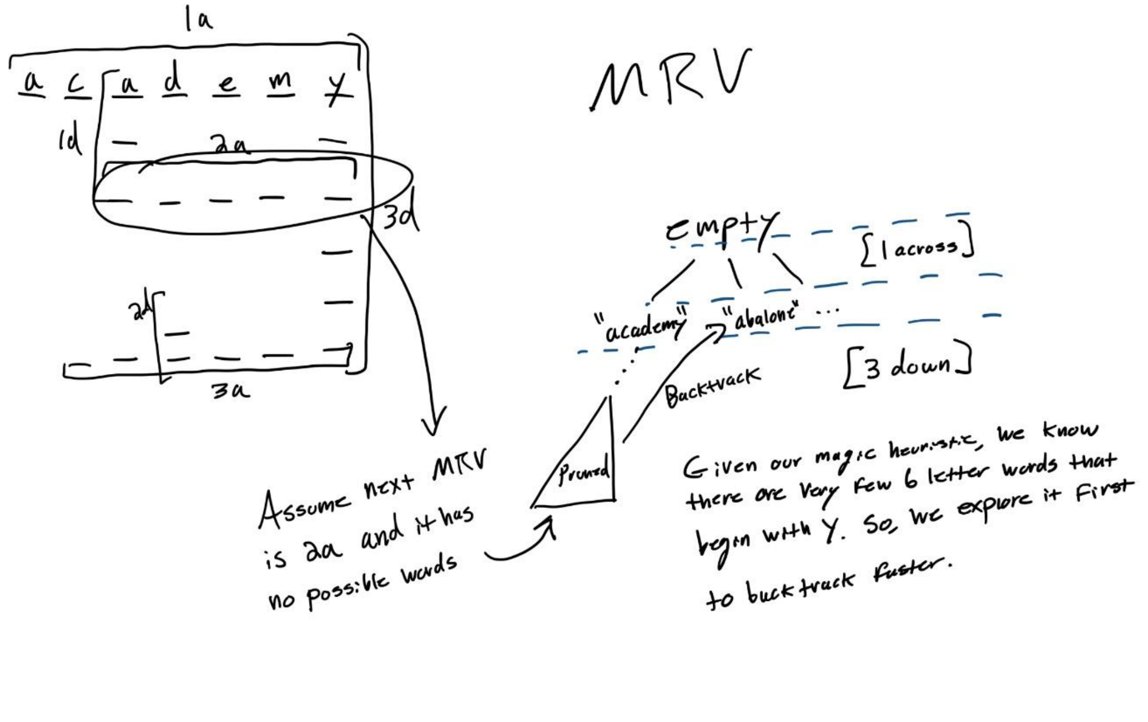


Backtracking



Assume "eagan" leads to a situation in which 3a has no words to choose from.

MRV:



Assume next MRV is 2a and it has no possible words

Given our magic heuristic, we know there are very few 6 letter words that begin with Y. So, we explore it first to backtrack faster.