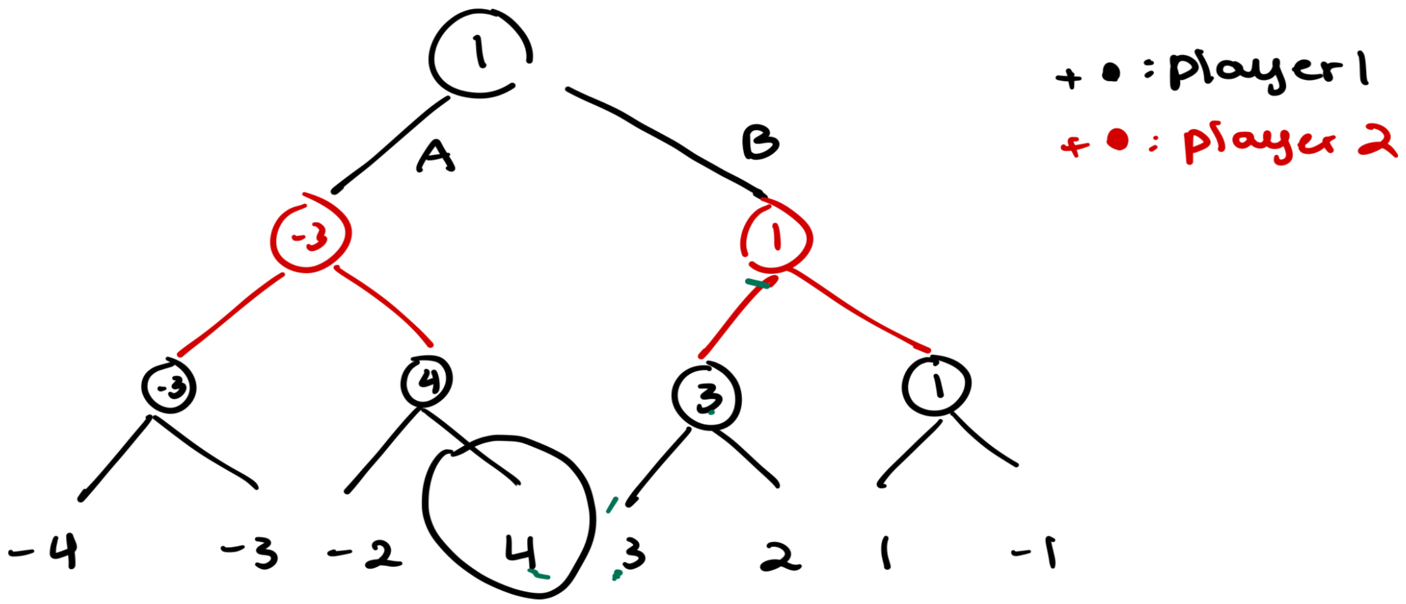
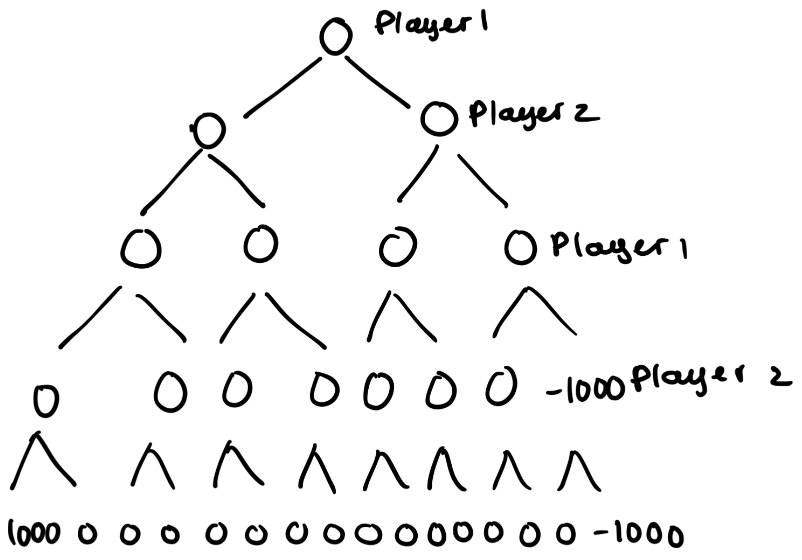
HOMEWORK 1

Question 1:



- Assumption: Player 1 will have an advantage when the number is higher
- Upon the example tree, player 1 should take action B (higher value)
- The expected outcome of the game is 1 when both players are playing optimally.
- We can prune out the leaf that has the utility of 4.
- To maximize the number of nodes being pruned, we can consider re-organize the nodes' utilities from highest to smallest or in this order: 1 2 3 4 -4 -3 -2 -1.
- To make sure that there is no pruning that might happen, we can consider re-organize the nodes' utilities from smallest to highest or in this order: -1 -2 -3 -4 4 3 2 1.

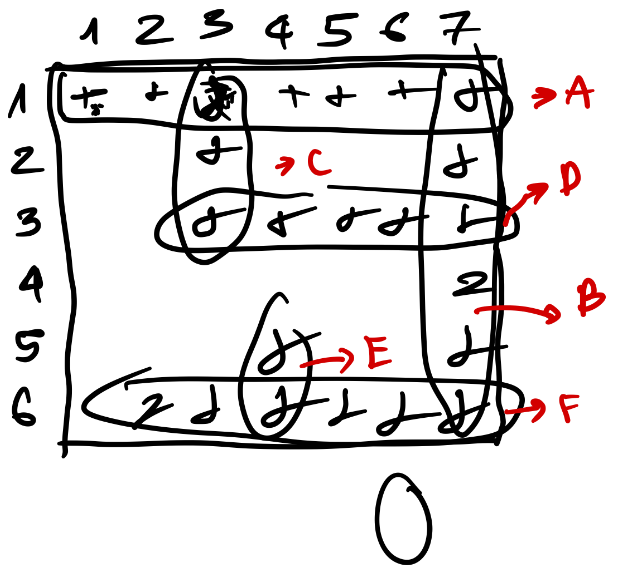
Question 2:



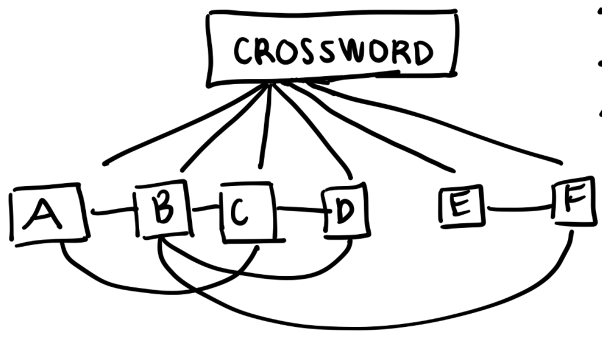
- Assumption: Player 1 will want to have a winning force (+1000), while player 2 will want to take the losing force (-1000).
- The player at the root can not force a win, the only value that they can take is 0.
- It does not matter where the 2 non-zeros states are located because the 1000 will get filtered out at very early stages (even when the depth is 2), and the -1000 will always get eliminated even when it makes it to the last round (not very likely as well).
- Even when we have different depths, the final answer will always be tied (0) because player 2 will always pick either 0 or -1000 while player 1 will always choose 0 compared to -1000 (except for depth 1, but then it does not really a game anymore since there's only 1 move to be made).

Question 3:

- Variables: A,B,C,D,E,F (the 7 words in the crossword puzzle).
- Domains: Ai,Bi,Ci,Di,...Fi belongs to {a,..,z}.
- Constraints: the variables must belong to the finite dictionary, and any matching letter between the variables must be the same.
- NOTES: We will treat each word as A,...F and then each letter in a word would be from a(1),...a(7)(based on the number of letters in the word).



- Constraint graph:



- nodes A are labeled w {a_1, . . , a_7}
- nodes B are labeled w {b_1, ..., b_6}
- nodes C are labeled w {c_1, ... c_3}
- nodes D are labeled w {d_1, .. d_5}
- nodes E are labeled w {e_1, e_2}
- nodes F are labeled w {f_1, .. f_6}

- Edges $A_7 = B_1$, $A_3 = C_1$, $B_3 = D_5$, ...

- **Describe first couple of steps of standard backtracking:**
    - Suppose 'ace' is the first 3-letter word starting with 'a', we can have the combination aaa, however, aaa is not in the dictionary -> prune, we can also have aba,aca,ada,...aza, but all of these are not in the dictionary -> prune, we will do this with each letter until we found the one that is indeed in the dictionary -> ace. Each time that we did not find a word successfully, we will backtrack to the most recent choice. We will keep on trying the letter until we find one that makes sense. Furthermore, d(5)=a(3) as well.
- **Describe how using the MRV would change the search; describe the first couple of steps again:**
    - Rather than going in order, we can start with the word that has the least option (the longer the word, we would assume that it has the least possible option in the dictionary. So we would want to start with the longest word that starts with 'a'). We might want to start with the word that has 7 letters (A). Then, we will go over all the options for each letter, then proceed to the next longest word that has an intersecting letter with word A (if this is the case, we even narrow down the option), so we might want to go with word B next, and make sure that we meet all of the required constraints.