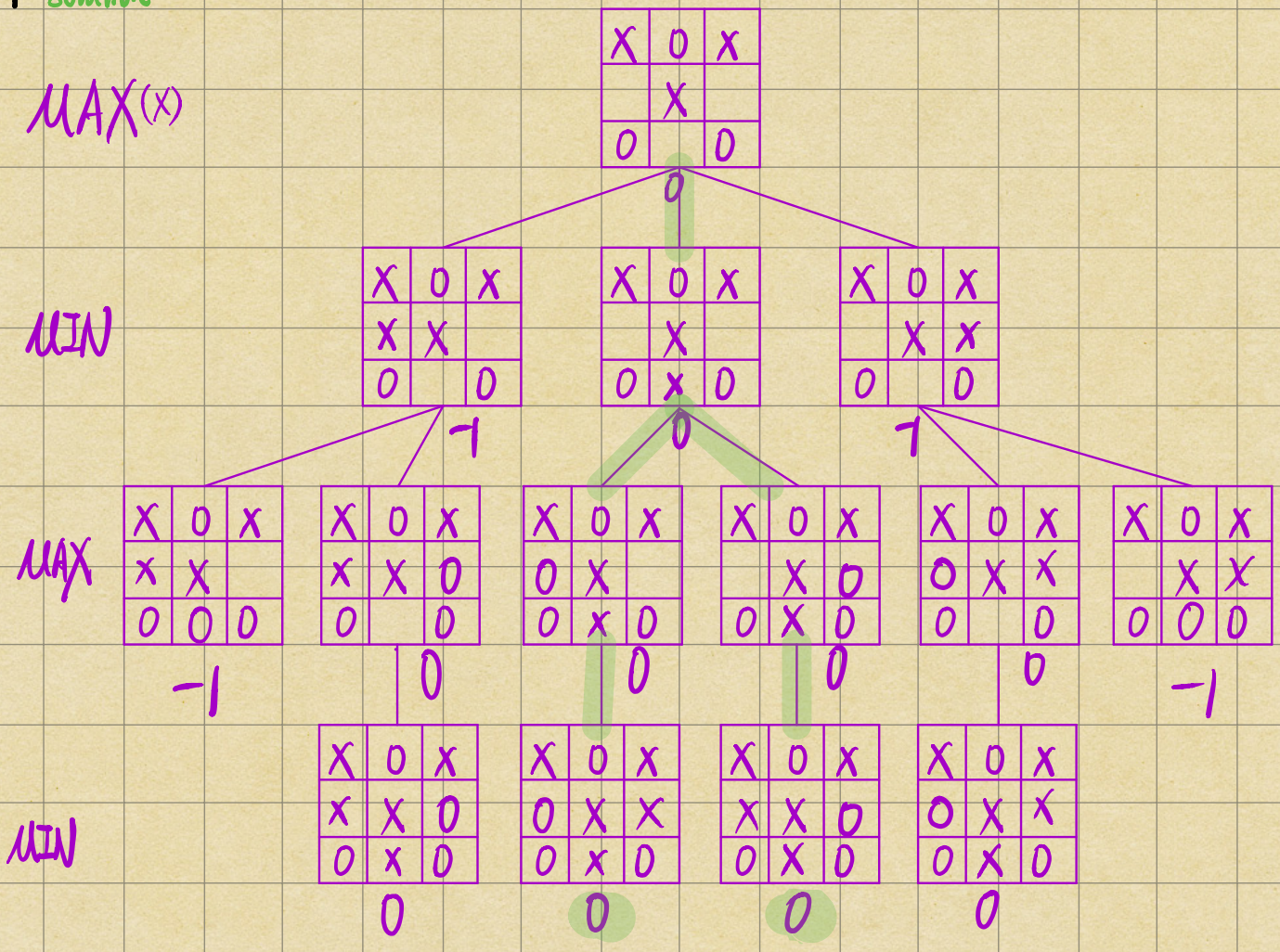
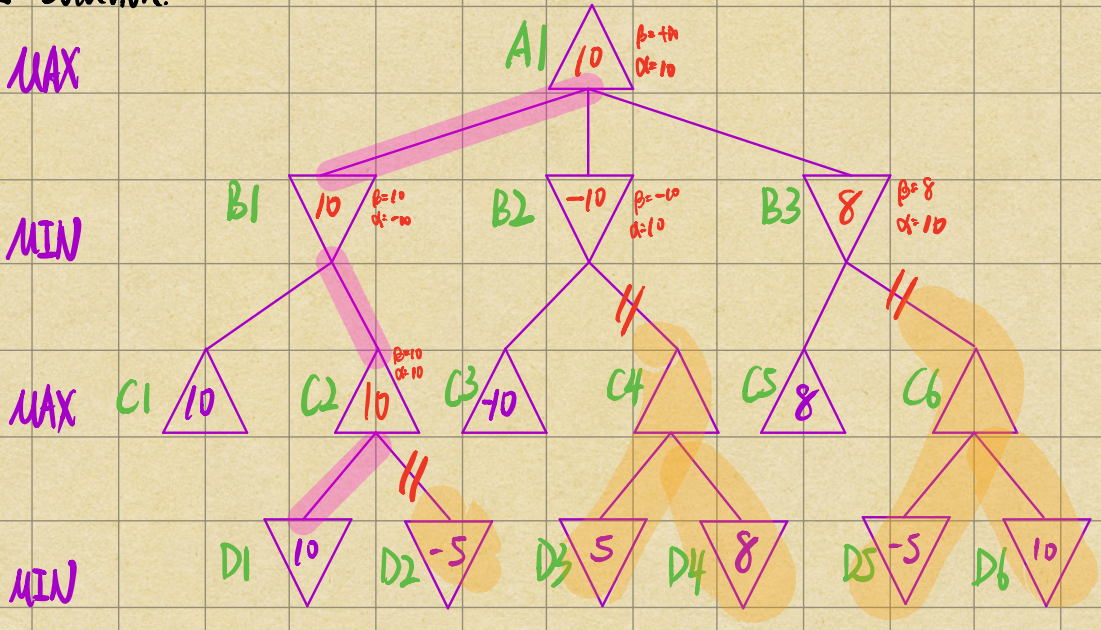
Problem 1 : solution



The optimal action for the MAX player(X) to play is green path

Problem 2: Solution.

a.

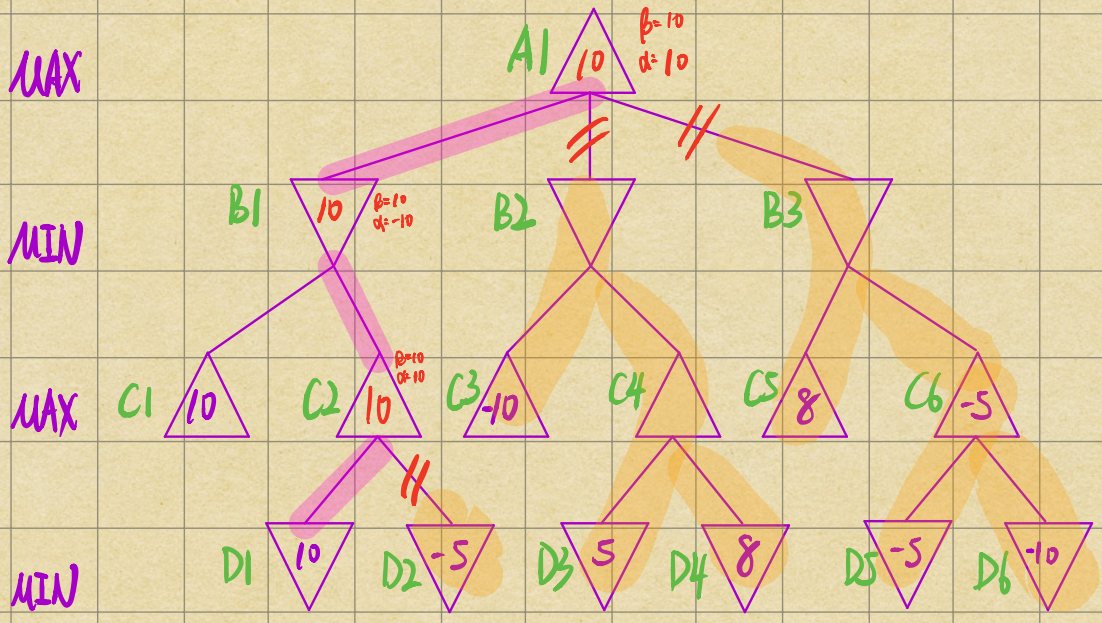


Thus, D2, C4, D3, D4, C6, D5, D6 will be pruned.

The estimated utility values for the rest of the nodes are

A1: 10, B1: 10, B2: 8, B3: 8, C2: 10

b. Given additional knowledge, we obtain $\alpha = -10, \beta = 10$ as prior condition.

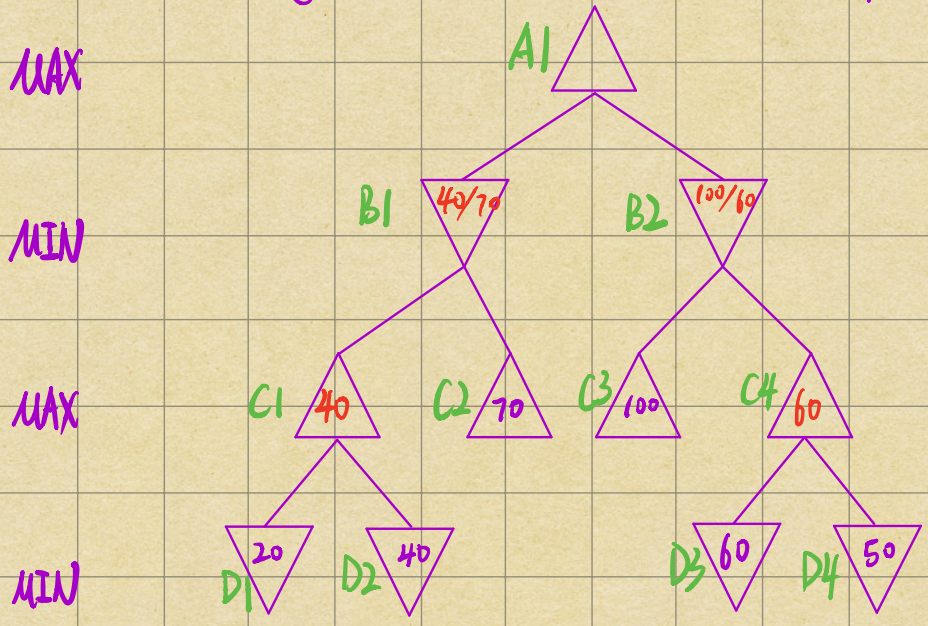


As the pruned tree above, we find that this knowledge can further improve the efficiency.

The pruned nodes are: B2, C3, C4, D3, D4, B3, C5, C6, D5, D6,

Problem 3: Solution.

Because we don't know algorithm the opponents uses, we have four possible second level nodes: (MIN player)

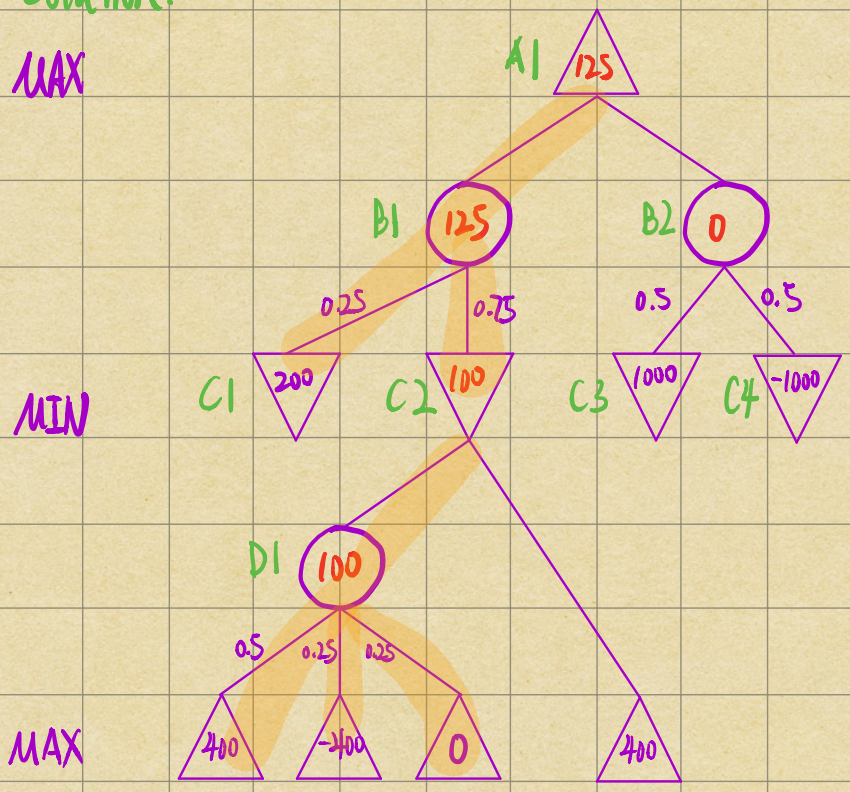


{B1: 40, B2: 100}, {B1: 40, B2: 60}, {B1: 70, B2: 100}, {B1: 70, B2: 60}.

For each case, we obtain 4 possible A: $\{A_{11}: 100, A_{12}: 60, A_{13}: 100, A_{14}: 70\}$.

Thus, the best possible outcome is 100, the worst possible outcome is 60.

Problem 4: solution.



1' The value of non-terminal nodes are A1: 125, B1: 125, B2: 0, C2: 100, D1: 100

2' The strategy will choose left subpath from A1,

3' The MinMax value obtained by the root node represent the expectation of the MiniMax Algorithm. of MAX player.

4' Maximum payoff is 400, minmum payoff is -400. if MIN plays the optimal strategy.

5' If MIN plays a random strategy, C2 could have {100, 400}.

but the Maximum payoff is also 400, the minum payoff is also -400.

Problem 5:

We can just modify the MINMAX algorithm to solve this problem.

1° Adjust the minValue(state) funtion through replace Successor(state) funtion to DeepGreenMove(state).

```
function beatDeepGreen(state) returns an action
    inputs: state, current state in game
    returns: the a in ACTIONS(state) maximizing minValue(RESULT(a, state))
function maxValue(state) return a utility value
    If terminalTest(state) then return utility(state)
    V ← -∞
    for a,s in successors(state) do V ← MAX(v, minValue(s))
    return v
function minValue(state) return a utility value
    If terminalTest(state) then return utility(state)
    V ← -∞
    for a,s in DeepGreenMove(state) do V ← MIN(v, maxValue(s))
    return v
```