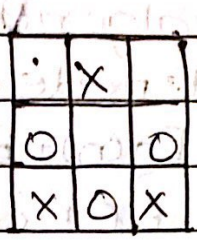
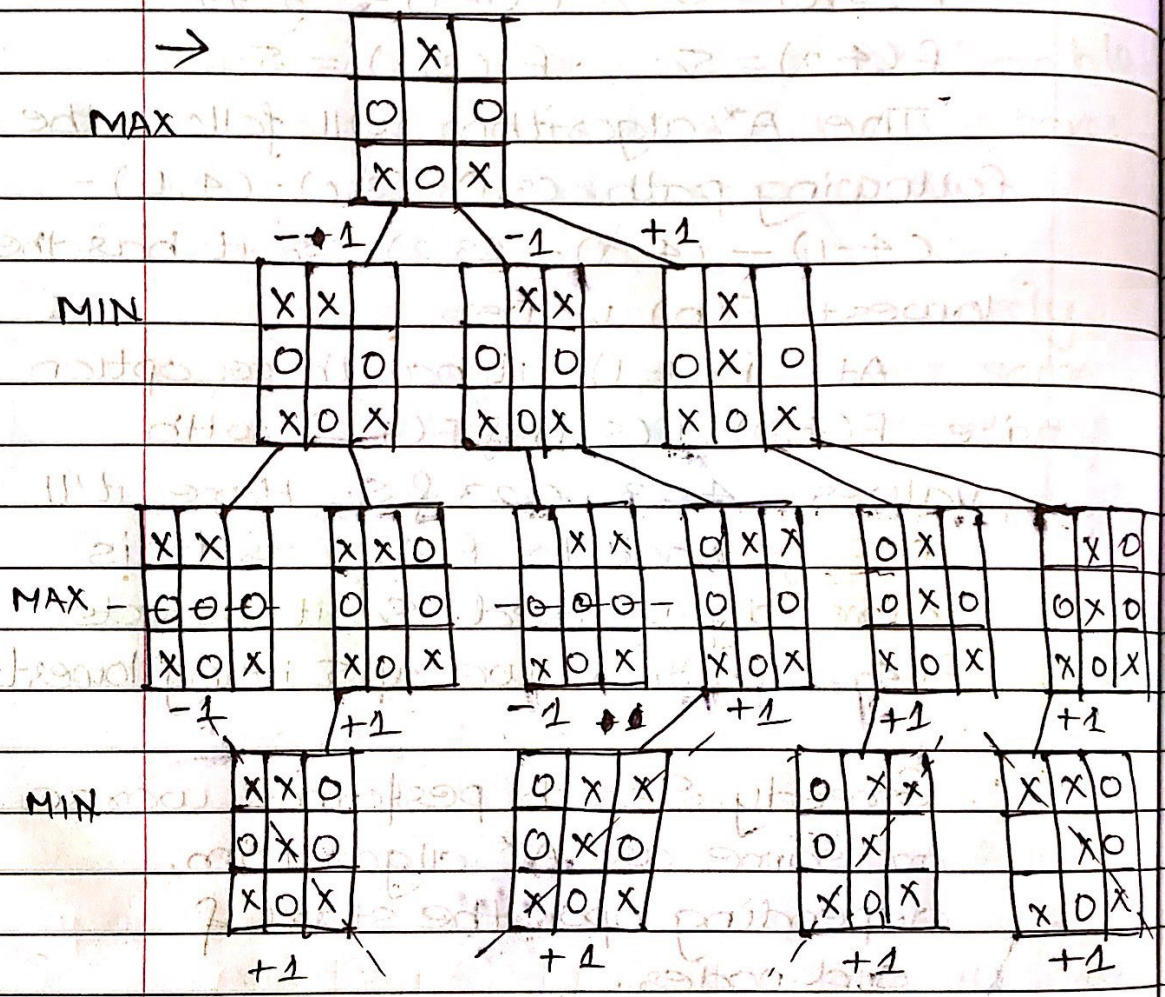
CSE 5360-001
PRATIK PRAMOD SINGH
1001670417 - UTA ID.

Page No. 1
Date

ASSIGNMENT - 3.

Q.1.



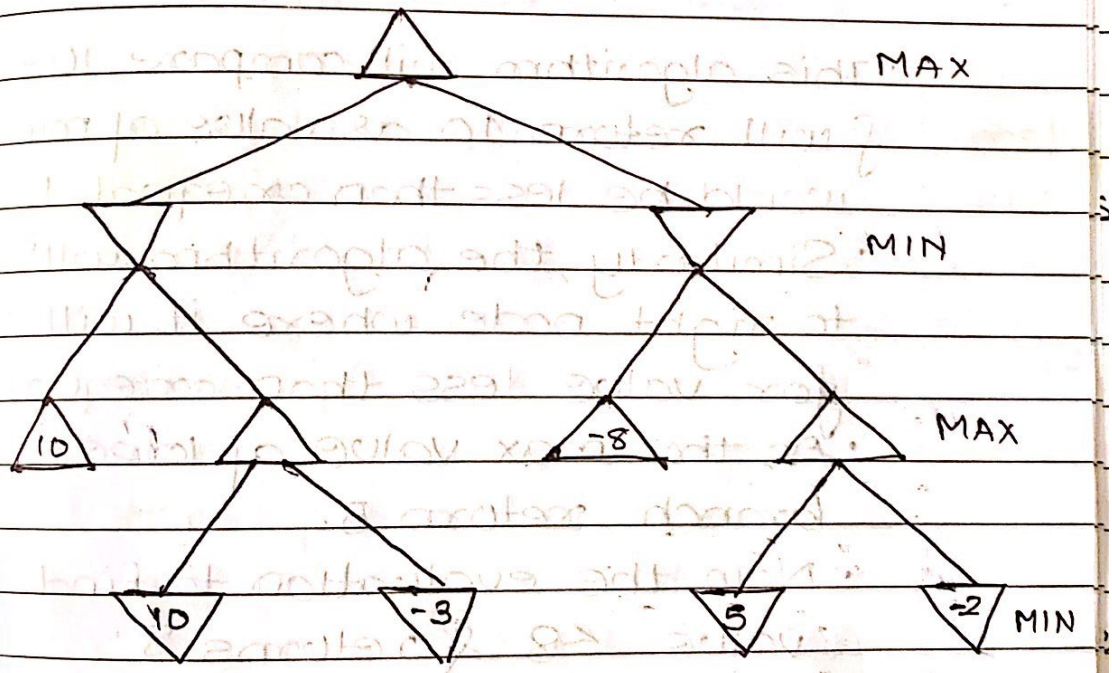


The utility value for each terminal & non-terminal node have represented. As 'X' is the max player, it is shown that it can get max best utility value of '+1'.

There are 4 paths that leads to the optimal value of minmax algorithm can randomly choose any one of them.

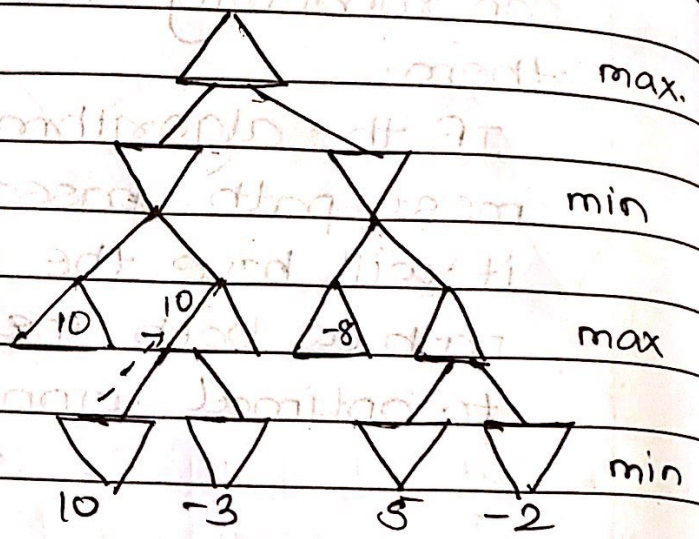
If the algorithm chooses the Right most path based on the solution, it will have the most optimal path as both the path will lead to optimal winning solution.

Q.2)



a) Alpha-Beta search explore node in left to right manner. Given the case, max player plays first explore leftmost value 10 & then goes to it's neighbour node with evaluation of exploring value less than or equal to 10.

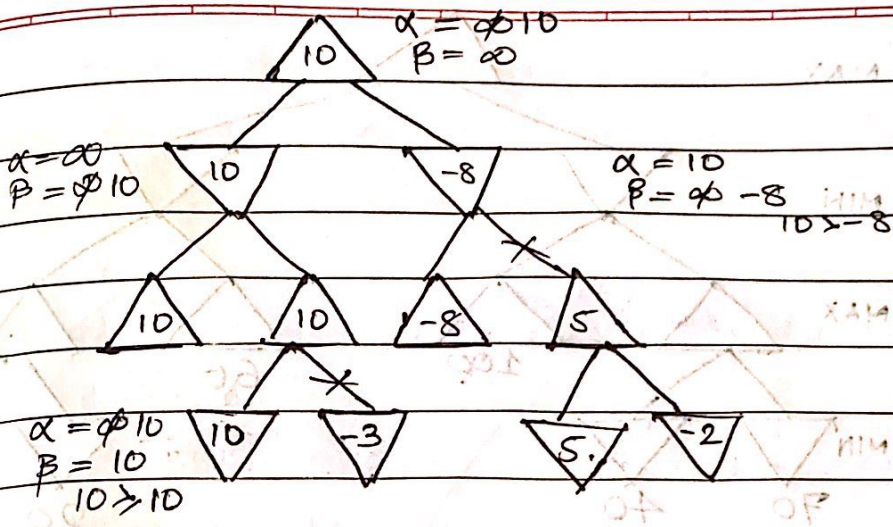
The value of neighbouring node it'll evaluate max of 10 & -3. which return 10.



- This algorithm will compare 10 & 10 & will return 10 as value of min would be less than or equal to 10.
- Similarly, the algorithm will move to right node where it will search for value less than or equal to 8
- As the max value of lower right branch return 5.
- Now, the evaluation to find any value ≤8 & returns 8.

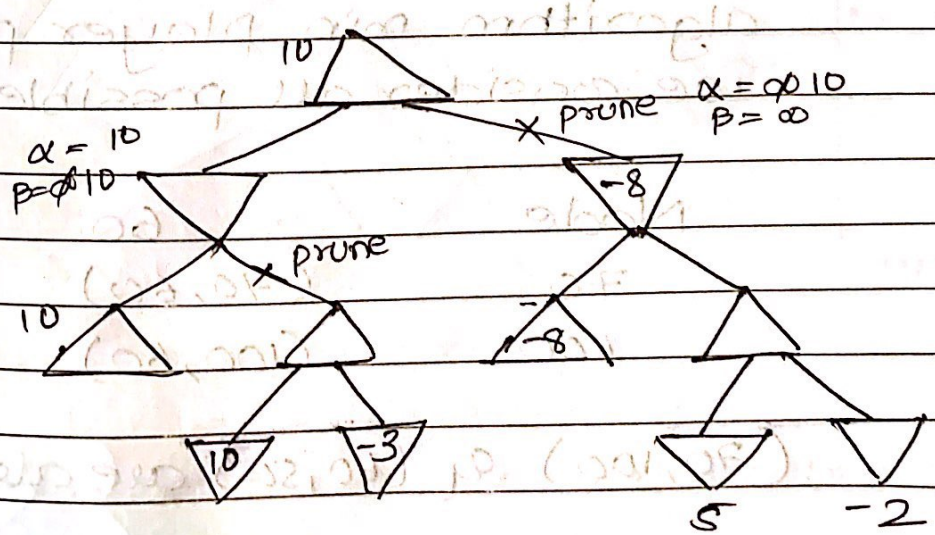
Out of 10 & -8, the algorithm then chooses 10 as the MAX value

- The alpha-beta pruning will prune neighbouring node to -8 as there is no value shorter than -8 at lower level as it return 5.

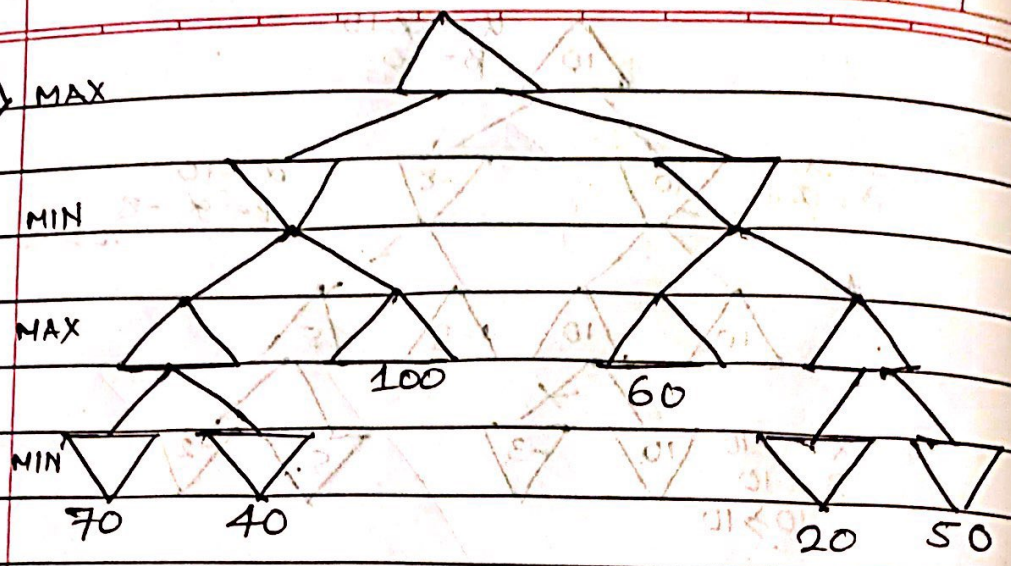


b) The algorithm, now being having the value 10 being the max utility value. This knowledge will improve the efficiency as while searching from left to right, the leftmost being 10, the algorithm will know that there will be no greater value possible as it's neighbouring node or lower node.

Hence, it will stop there & prune other nodes.



Q.3)



→ As we are playing the MAX player & we don't know what algorithm the opponent player uses. As per minmax algorithm, the max player at level-3 will choose the most highest value.

max(70,40) = 70 & Max(20,50) = 50.

Now, in level-2, we don't know the algorithm min player plays. we consider all possible outcomes.

Node	60	50
70	(70,60)	(70,50)
100	(100,60)	(100,50).

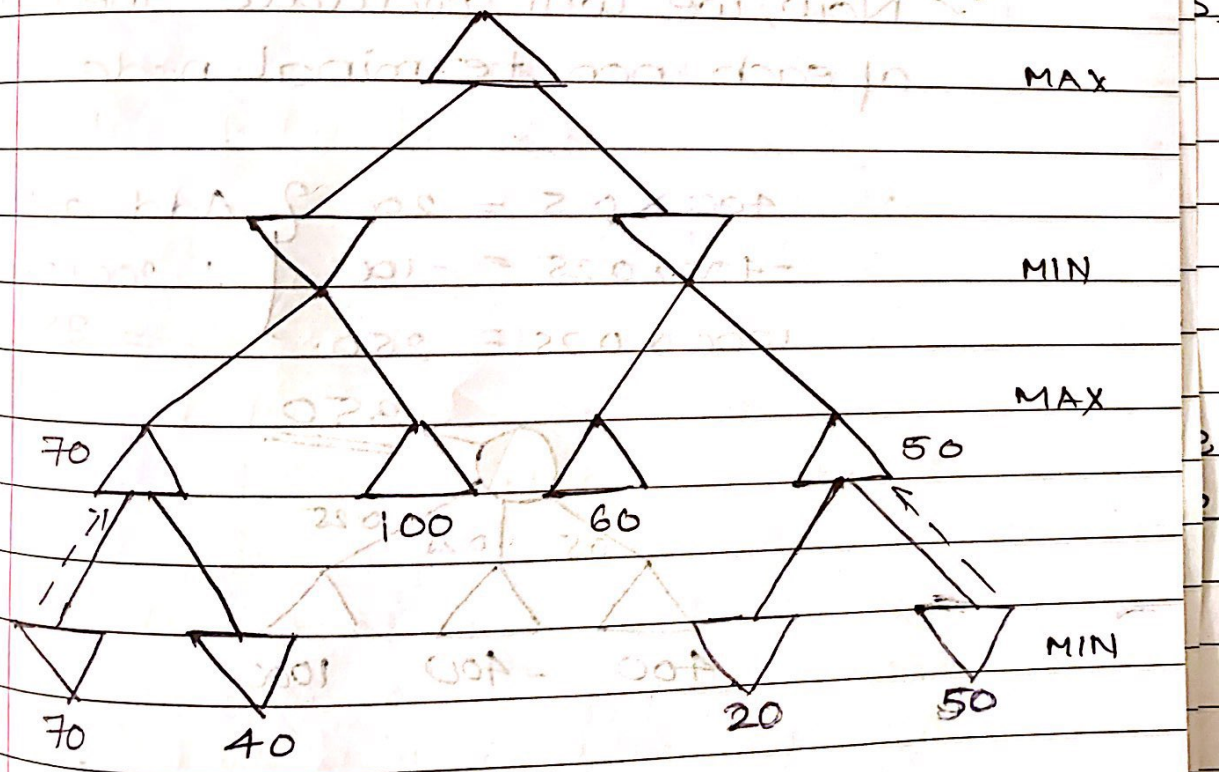
(70,100) & (60,50) are also possible.

Hence, in level-1 max player has 4 different comparison to choose from

∴ Best possible outcome for max = 100.

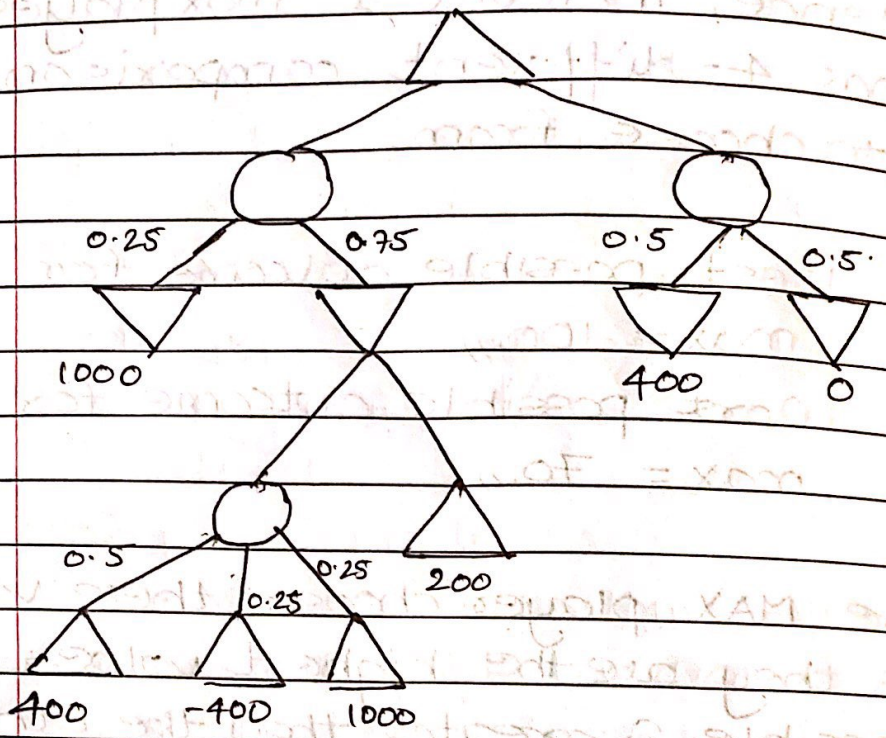
∴ Worst possible outcome for max = 70.

The MAX player chooses these values as they are the highest values possible & greater than the other present values. Also the parent



node being max, it'll choose the best highest value.

0.4)

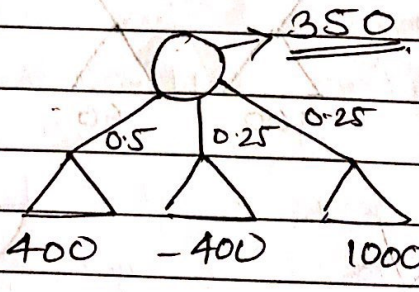


→ Now, we will calculate the value of each non-terminal node.

$400 \times 0.5 = 200$
$-400 \times 0.25 = -100$
$1000 \times 0.25 = 250.$

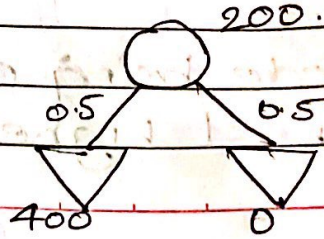
} Add all

∴ 200+250-100 = 350.

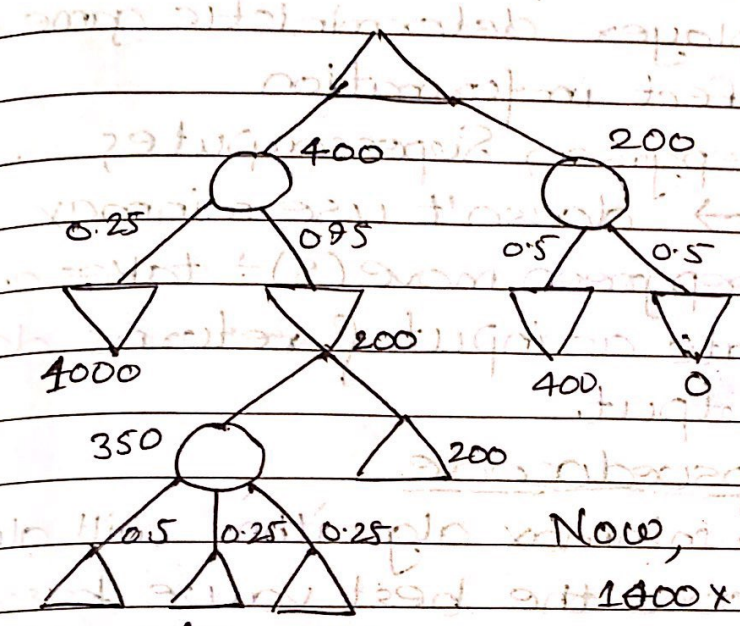


$400 \times 0.5 = 200$
$0 \times 0.5 = 0$

∴ 200



Now the tree looks like.



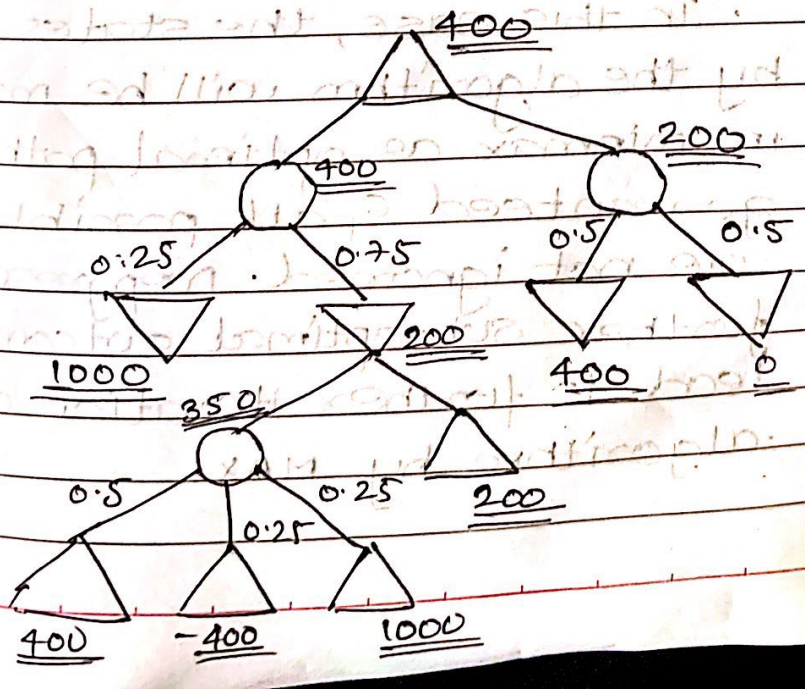
Now,

$1000 \times 0.25 = 250$

$200 \times 0.75 = 150$

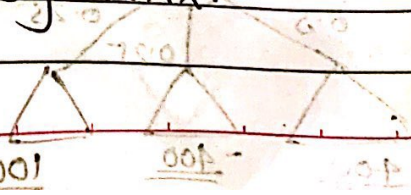
$\therefore 250 + 150 = 400$.//

As, left node value is 400 & greater than the value of right node value. i.e. 200.
The max player or top non-terminal node value will be 400.



Q.5) 2-player deterministic game of perfect information

- Deepgreen Supercomputer
  → doesn't use minmax
- Deepgreen move(S) ÷ takes any state as input & returns deepgreen output
  → pseudocode
  - minmax algorithm will always choose the best value based on what the output Deepgreen move(S) provide from previous state.
  - Deepgreen doesn't use minmax algorithm, which mean it doesn't return an optimal solution to max, thereby providing a sub-optimal solution & output from deepgreen move(S) library function
  - In this case, the states explored by the algorithm will be more than in minmax as optimal path is not guaranteed & all possible states are not ignored. Deepgreen move(S) further sub-optimal outcome will lead to further iteration of algorithm by MAX.



· This will lead to a more optimal outcome will lead for MAX than actual minmax as the range of winning will be much more than what min could've provided.

→ pseudocode : following minmax style.
· function minmax_decision(state) returns an action, input state, current state in game.
· Returns a in action(state) maximizing Deepgreenmove(result(s)).
· Function max_value(s) returns utility value. If terminal_test(s) then return utility(s).
· for a,s in Successors(s) do
v ← max(deepgree move(s))
return v.

· Hence, the optimal solution of the algorithm will explore more states than the minmax, but due to suboptimal outcome of Deepgreen, the max function which maxes over it will provide a greater utility value than the optimal algorithm.