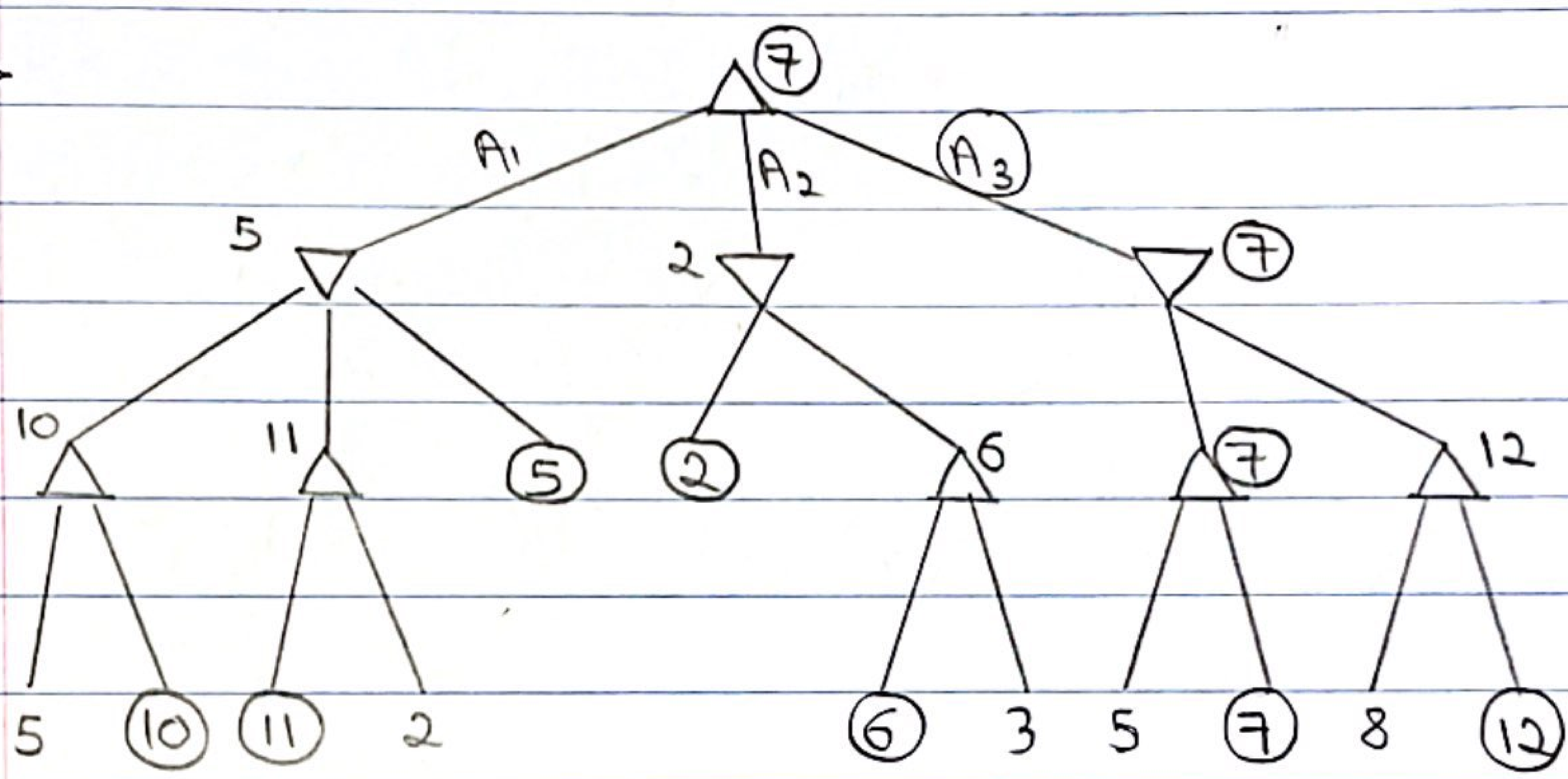
Prem A. Jethwa
1001861810

- Assignment 2 -

* Task 1 :

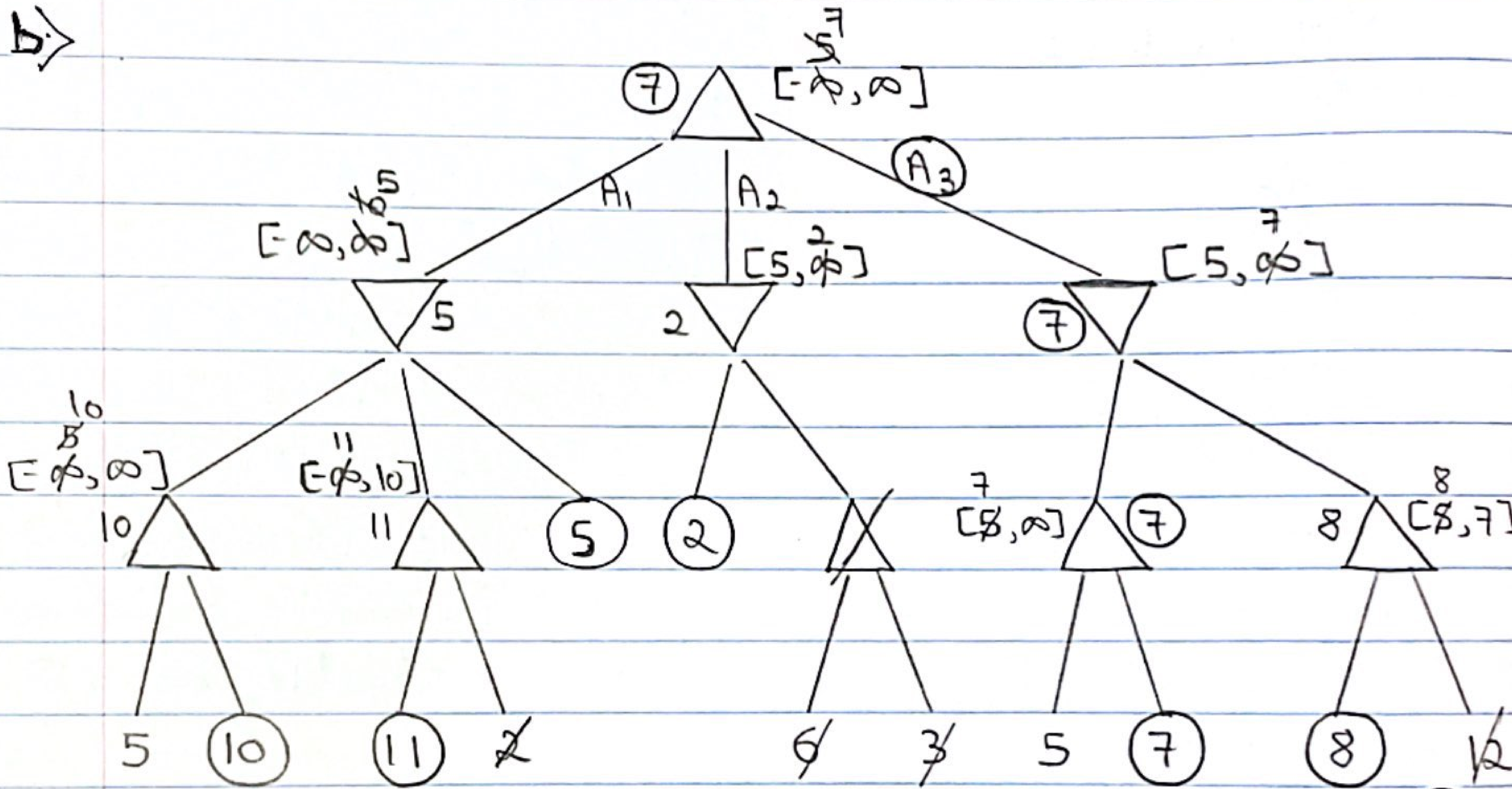
a)



From the figure, the action A_3 maximizes the min values to 7. So, we can say that Minimax algorithm will pick action A_3 to execute.

Scanned with CamScanner

b:>



From the figure, Minimax algorithm will pick action A_3 to execute.

Yes, the answer is same as part a.

c:> For MIN node:

if $v <= \alpha$ or $v == 2$ then prune

if $v < \beta$ then $\beta = v$

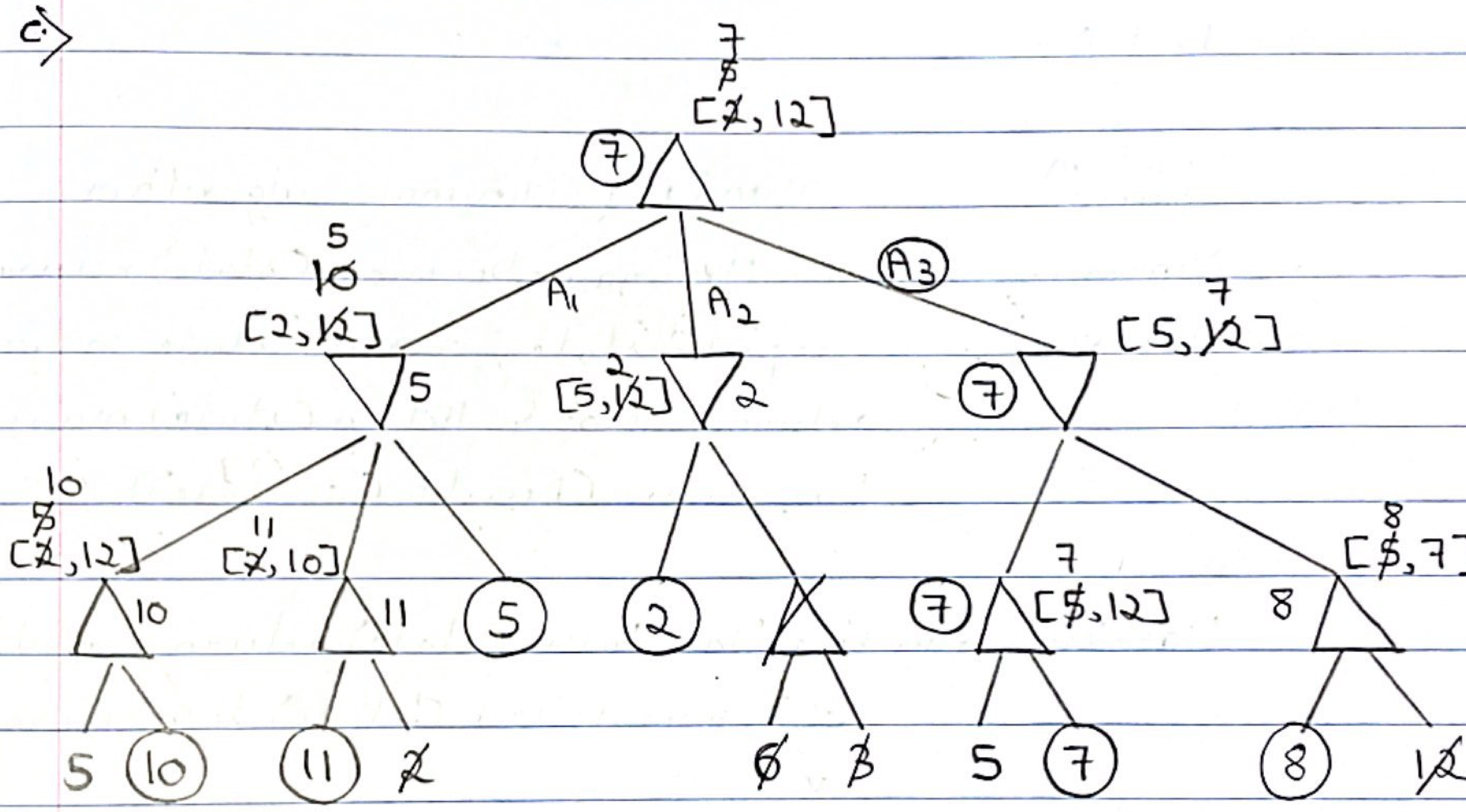
For MAX node:

if $v >= \beta$ or $v == 12$ then prune

if $v > \alpha$ then $\alpha = v$

Scanned with CamScanner

c)



If we know that maximum possible value can 12 and minimum possible value is 2 then we can assign the α & β values accordingly.

$\alpha = 2 \rightarrow$ Highest possible score till now.

$\beta = 12 \rightarrow$ Least possible score till now.

Also, the α-β pruning minimax algorithm can be improved as follows to prune the node if any successor of max node has value = 12 or any successor of min node has value = 2.

Scanned with CamScanner

* Task 2 -

Function Name	Standard Minimax algorithm
Minimax-Decision	function Minimax-Decision (state) returns an action input: state, current state in game return the a in Action (state) maximizing Min-value (Result (a, state))
Max-Value	function Max-Value (state) returns a utility value if Terminal-Test (state) then return Utility (state) $v \leftarrow -\infty$ for a, s in successor (state) do $v \leftarrow$ Max (v, Min-value (s)) return v.
Min-Value	function Min-Value (state) returns a utility value if Terminal-Test (state) then return Utility (state) $v \leftarrow \infty$ for a, s in Successor (state) do $v \leftarrow$ Min (v, Max-value (s)) return v.

Scanned with CamScanner

Function name	Modified Minimax Algorithm
Minimax-Decision	function Minimax-Decision (State) returns an action inputs: state, current state in game returns the a in Actions (state) maximizing Min-Value (Result (a, state))
Max-Value	function Max-Value (state) returns a utility value if Terminal-Test (state) then return Utility (state) $v \leftarrow -\infty$ for a, s in Successor (state) do $v \leftarrow$ Max (v, Min-Value (s)) return v.
Min-Value	function Min-Value (state) returns a utility value if Terminal-Test (state) then return Utility (state) $v \leftarrow$ Max-value (DeepGreenMove (state)) return v.
DeepGreen Move	function DeepGreenMove (state) returns state from min players move # do something to return state after # min players move without using # minimax algorithm. return state.

Scanned with CamScanner

Here, the only required change is in the Min-Value function.

As given in the question, the function DeepGreenMove returns the state after the Min player plays. So now we can use this function in the Min-Value function to predict the Min player's moves and prune the rest of the unnecessary branches in the tree instead of iterating through every possible move that the Min player can play. This can reduce the number of nodes in the tree to almost half.

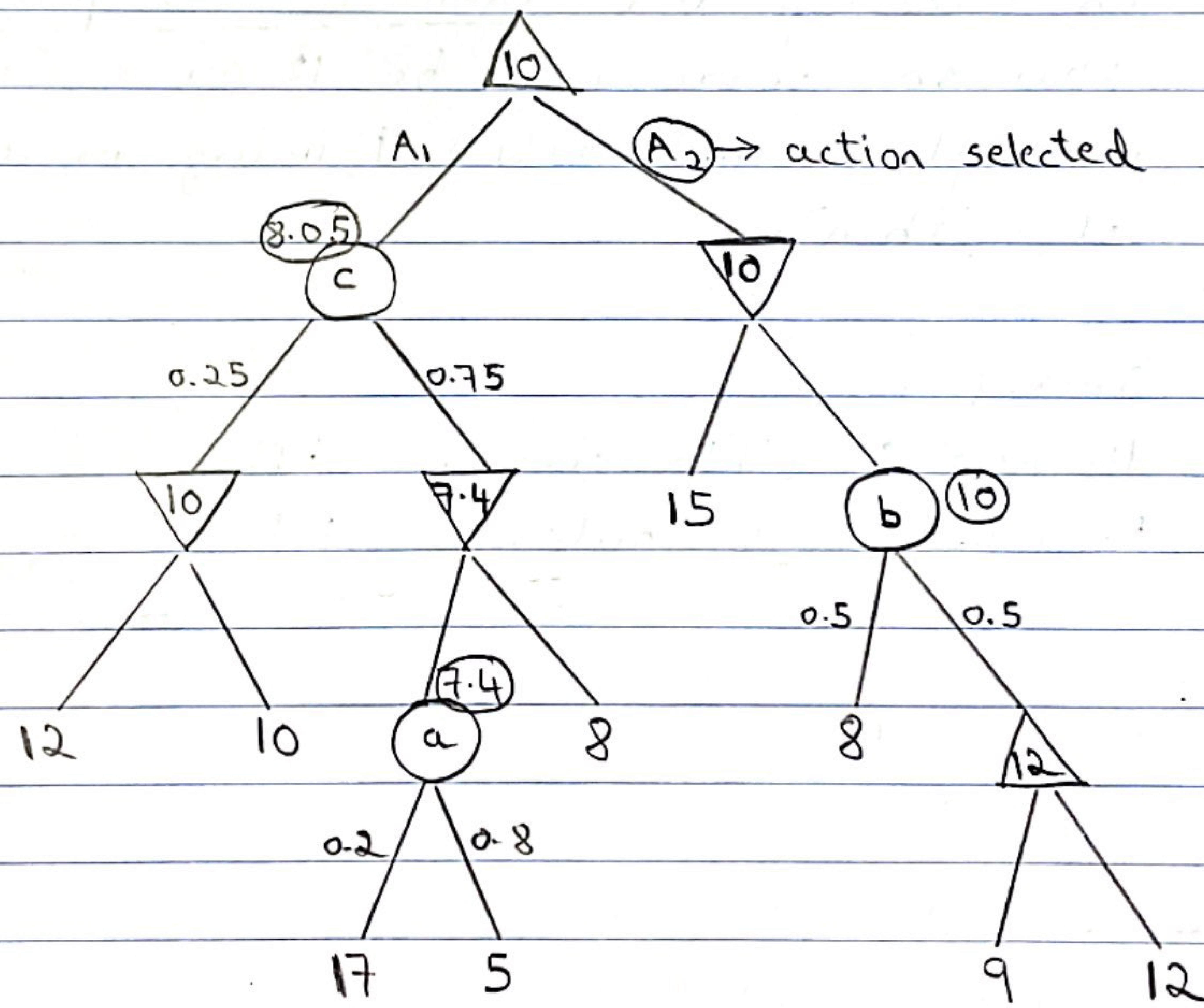
* Advantage of modified Minimax Algorithm using DeepGreenMove function.

As mentioned above this will result in fewer branches i.e. pruning the branches corresponding to the moves that the Min player will never play and reducing the tree size to approx. half. For an optimal player this algorithm will return the same action as the standard Minimax algorithm using less memory. And for non-optimal player

Scanned with CamScanner

This algorithm will gives the best action resulting in a state with maximum utility value.

* Task 3:



For node a, value = (17)(0.2) + (5)(0.8) = 7.4

For node b, value = (8)(0.5) + (12)(0.5) = 10

For node c, value = (10)(0.25) + (7.4)(0.75) = 8.05

Scanned with CamScanner

The action selected by the minimax algorithm is A_2 as it maximizes the minimum values.

After selecting action A_2, an optimal opponent will select the move corresponding to node b. On selecting that there is 0.5 probability that the score will be 12 or 8 (since the max player will select 8 using minimax algorithm).

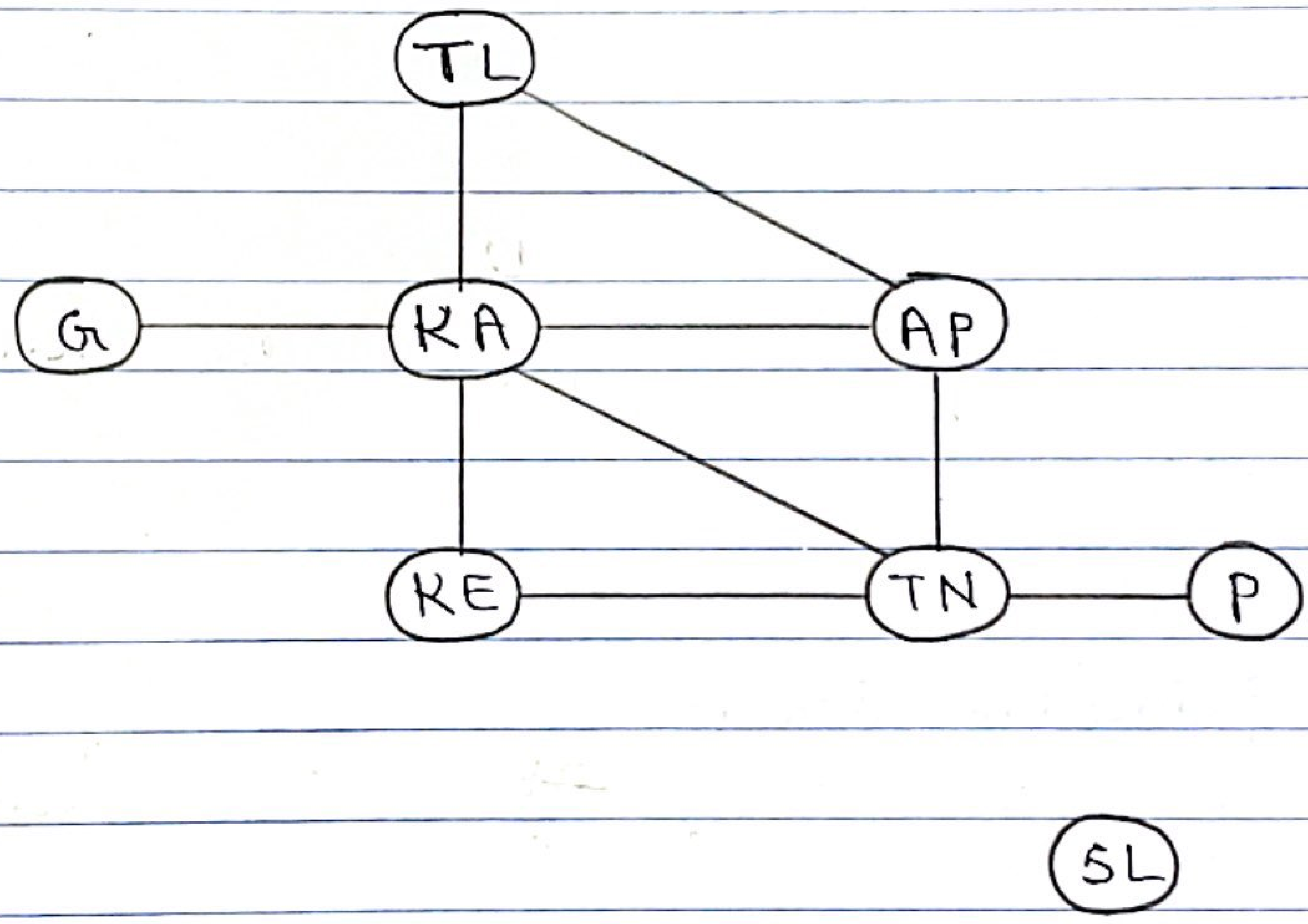
Therefore,
Highest possible outcome : 12
Lowest possible outcome : 8

Scanned with CamScanner

* Task 4-

a)

Constraint graph-



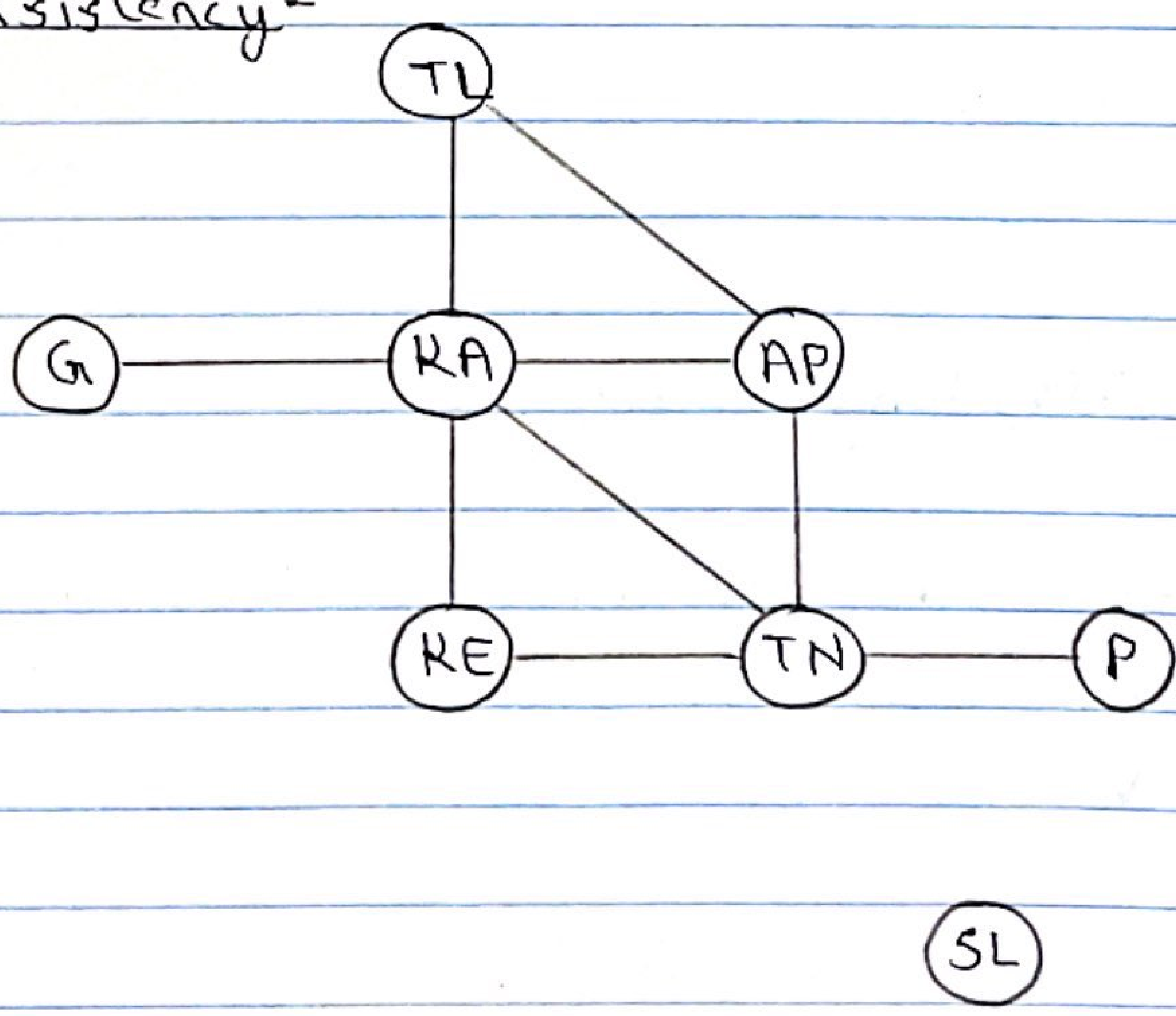
b) Backtracking Search-

Node	KA	TN	AP	KE	TL	G	P	SL
Degree	5	$\cancel{4}3$	$\cancel{3}\cancel{2}1$	$\cancel{2}\cancel{1}0$	$\cancel{2}\cancel{1}0$	$\cancel{1}0$	$\cancel{1}0$	0
MRV	3	$\cancel{3}2$	$\cancel{3}\cancel{2}1$	$\cancel{3}\cancel{2}1$	$\cancel{3}\cancel{2}1$	$\cancel{3}2$	$\cancel{3}2$	3

Scanned with CamScanner

Level	MRV	Degree Values	Node/Variable Selected
1	3	5	KA
2	2	3	TN
3	1	1	AP
4	1	0	KE
5	1	0	TL
6	2	0	G
7	2	0	P
8	3	0	SL

c) Arc Consistency -



Scanned with CamScanner

Assigning 'Red' to first variable ie. 'KA'.

KA	TN	AP	KE	TL	G	P	SL
R	~~R~~GB	~~R~~GB	~~R~~GB	~~R~~GB	~~R~~GB	RGB	RGB

1 ~~G → KA~~
~~TL → KA~~
~~AP → KA~~
~~KE → KA~~
~~TN → KA~~

2 ~~KA → G~~
3 ~~KA → TL~~
~~AP → TL~~
4 ~~TL → AP~~
~~KA → AP~~
~~TN → AP~~

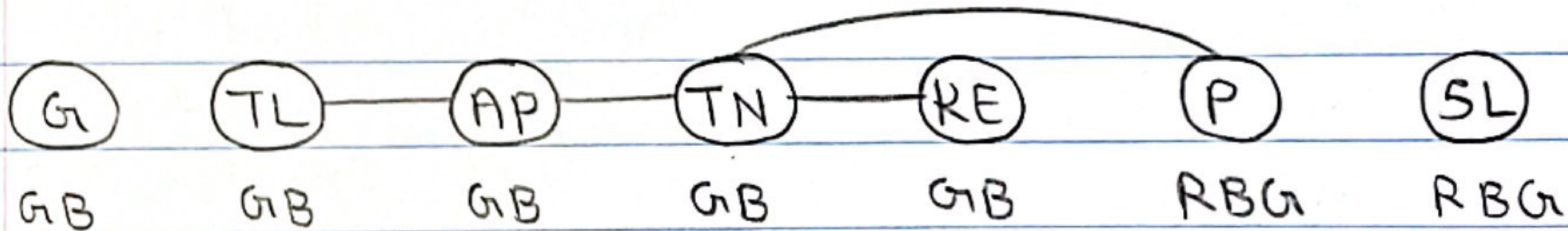
5 ~~KA → KE~~
~~TN → KE~~
6 ~~AP → TN~~
~~KA → TN~~
~~KE → TN~~
~~P → TN~~

d) Yes, we can use the structure of the problem to make solving it more efficient.
There are 2 possible ways to do so:

I. First, we can see that the region 'SL' is having no arcs connecting to any other regions. So, we can separate it from the main region and consider to any other subproblem and solve both the problems separately and concatenate the final result.

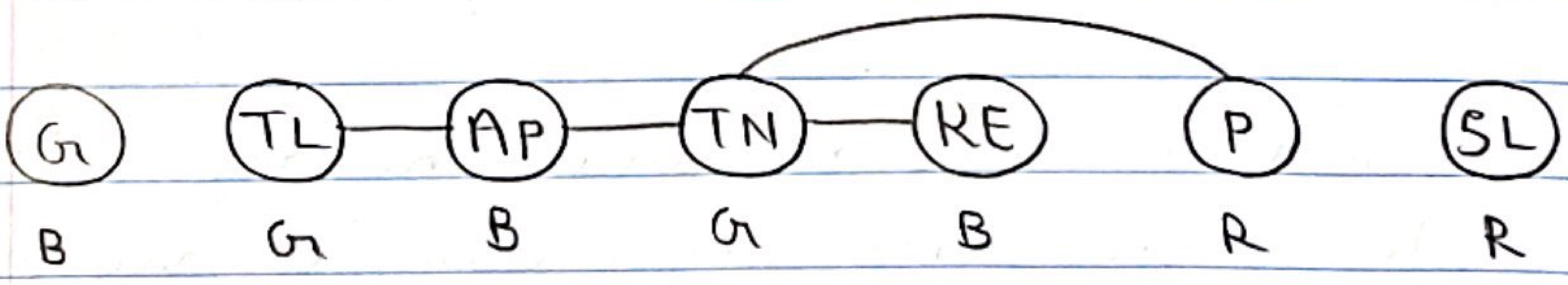
Scanned with CamScanner

II. We can see that the region/graph is not cyclic and cannot be solved considering it as a tree (which would be efficient). But we can see that it is nearly tree and we can determine the cut-set here consist of node 'KA' and assigning it to 'Red'. And then solve these subproblem using trees as shown below.



The above sub-problem can be easily solved in $O(nd^2)$ time.

e) One valid Solution -



Assign Blue to G
Assign Green to TL
Assign Blue to AP
Assign Green to TN
Assign Blue to KE
Assign Red to P
Assign Red to SL

Possible Solution:

Red ⇒ (KA, P, SL)
Green ⇒ (TL, TN)
Blue ⇒ (G, AP, KE)

Scanned with CamScanner