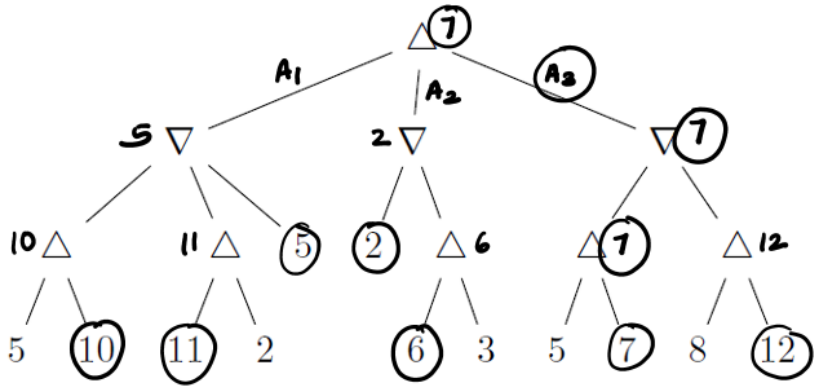
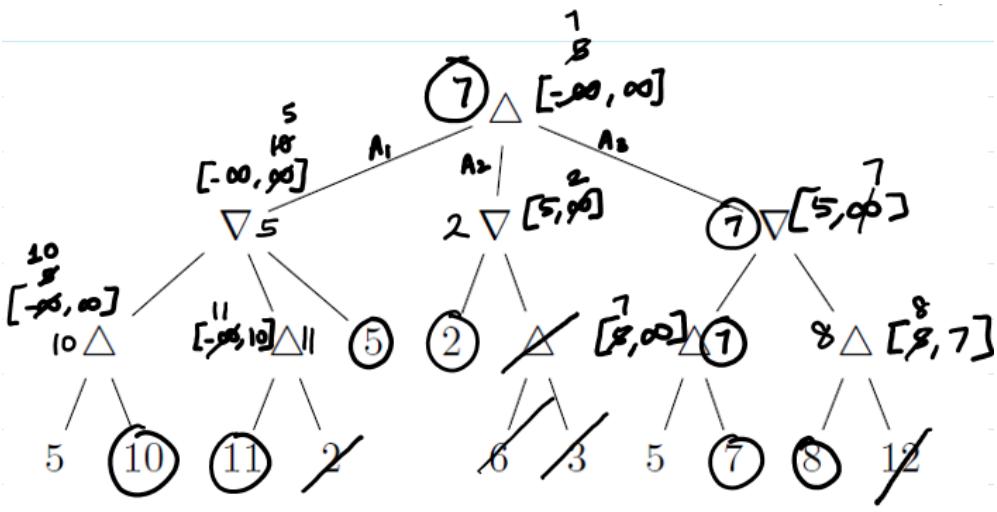
**Task 1:**

**a. (4308: 8 points, 5360: 8 points) In the game search tree of Figure 1, if we use the Minimax algorithm to find the strategy to use, indicate the minmax values are for all the nodes. Also indicate which action the Minimax algorithm will pick to execute.**



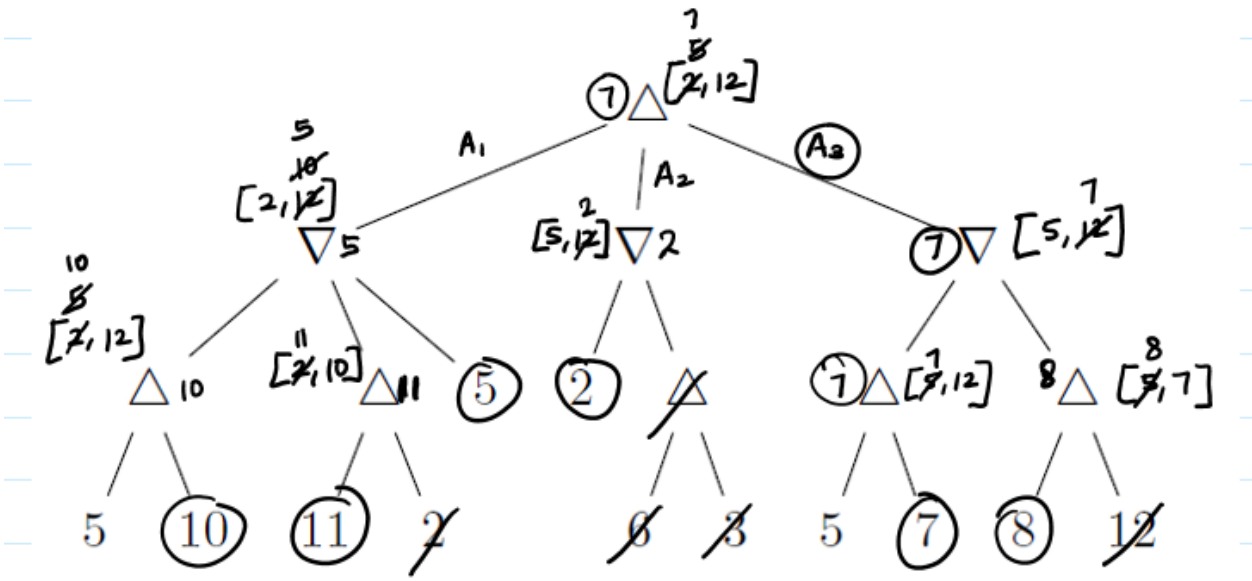

As we can see, the action $A_3$ maximizes the min values to 7
we can say that action $A_3$ is best action as per mini max algorithm.

**b. (4308: 7 points, 5360: 7 points) In the game search tree of Figure 1, indicate what nodes will be pruned using alpha-beta search, and what the minmax values are for the rest of the nodes. Assume that alpha-beta search expands nodes in a left-to-right order. Finally indicate which action the Minimax algorithm will pick to execute. Is this answer different from part a?**



Here the action selected is $A_3$ as it maximizes the minimum values of min nodes. This answer is the same as part a.

**c. (4308: 5 points, 5360: 5 points) This question is also on the game search tree of Figure 1. Suppose we are given some additional knowledge about the game: the maximum utility value is 12 i.e., it is not mathematically possible for the MAX player to get an outcome greater than 12 and the minimum utility value is 2 i.e., it is not mathematically possible for the MIN player to get an outcome lower than 2. How can this knowledge be used to further improve the efficiency of alpha-beta search? Indicate the nodes that will be pruned using this improvement. Again, assume that alpha-beta search expands nodes in a left-to-right order.**




If we know that maximum possible value can be 12 & minimum possible value is 2 then we can assign the $\alpha$ & $\beta$ values accordingly.

$\alpha = 2$ → highest possible score till now

$\beta = 12$ → least possible score till now

Also the $\alpha - \beta$ 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.

**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

**Task 2: Suppose that you want to implement an algorithm that will compete on a two-player deterministic game of perfect information. Your opponent is a supercomputer called DeepGreen. DeepGreen does not use Minimax. You are given a library function DeepGreenMove(S), that takes any state S as an argument, and returns the move that DeepGreen will choose for that state S (more precisely, DeepGreenMove (S) returns the state resulting from the opponent's move).**
**Write an algorithm in pseudocode (following the style of the Minimax pseudocode) that will always make an optimal decision given the knowledge we have about DeepGreen. You are free to use the library function DeepGreenMove(S) in your pseudocode. What advantage would this algorithm have over Minimax? (if none, Justify).**

| Function name | Standard Minimax algorithm | Modified minimax algorithm |
|---|---|---|
| Minimax-Decision | function Minimax-Decision(state) returns *an action*<br>inputs: state, current state in game<br>return the *a* in Actions(state) maximizing Min-Value(Result(*a*, state)) | function Minimax-Decision(state) returns *an action*<br>inputs: state, current state in game<br>return the *a* in Actions(state) maximizing Min-Value(Result(*a*, state)) |
| Max-Value | function Max-Value(state) returns *a utility value*<br>if Terminal-Test(state) then return Utility(state)<br>v ← −∞<br>for *a, s* in Successors(state) do v ← Max(v, Min-Value(*s*))<br>return v | function Max-Value(state) returns *a utility value*<br>if Terminal-Test(state) then return Utility(state)<br>v ← −∞<br>for *a, s* in Successors(state) do v ← Max(v, Min-Value(*s*))<br>return v |
| Min-Value | function Min-Value(state) returns *a utility value*<br>if Terminal-Test(state) then return Utility(state)<br>v ← ∞<br>for *a, s* in Successors(state) do v ← Min(v, Max-Value(*s*))<br>return v | function Min-Value(state) returns *a utility value*<br>if Terminal-Test(state) then return Utility(state)<br>**v ← Max-Value(DeepGreenMove(state))**<br>return v |
| DeepGreenMove | | function DeepGreenMove(state) returns *state from min players move*<br>*# Do something to return state after min players move*<br>*# without using minimax algorithm*<br>return *state* |

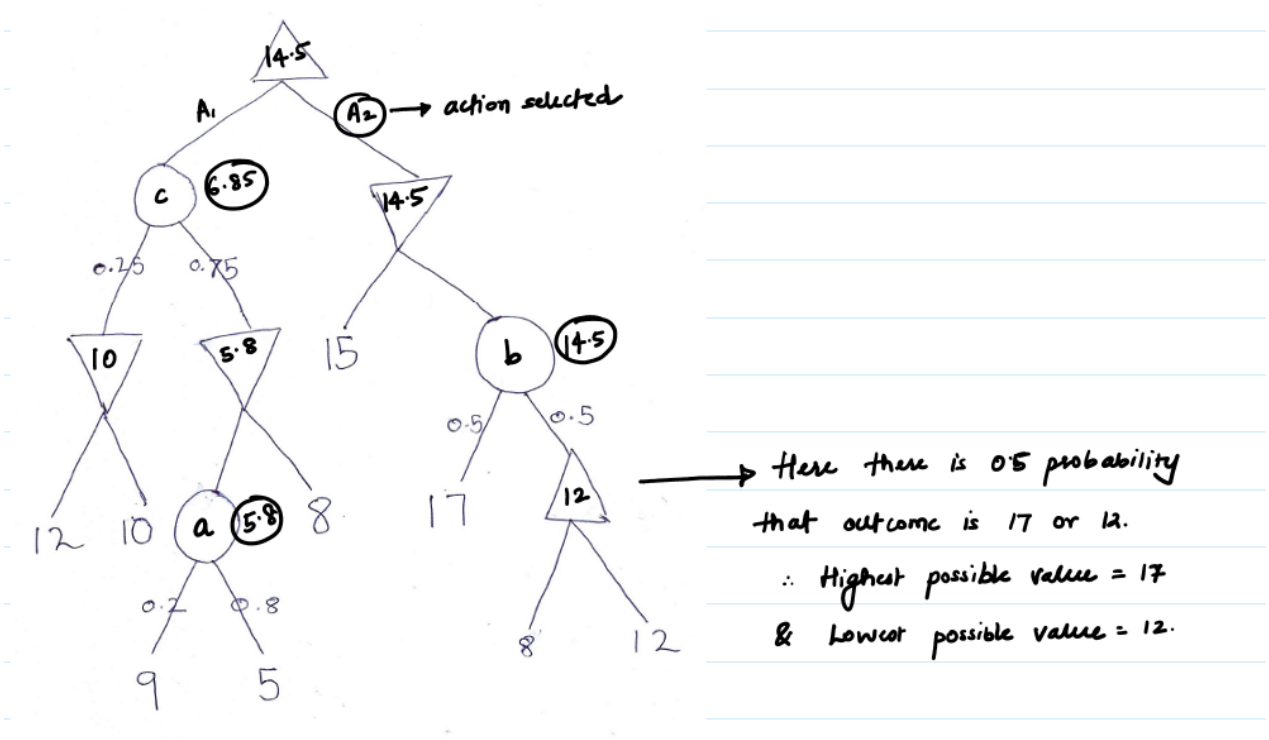
Here, the only required change is in the **Min-Value function**.

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 approximately half. For an optimal player this algorithm will return the same action as the standard Minimax algorithm using less memory. And for a non-optimal player this algorithm will give the best action resulting in a state with maximum utility value.

**Task 3: Find the value of every non-terminal node in the expectiminimax tree given above. Also indicate which action will be performed by the algorithm. What is the lowest and highest possible outcome of a single game if the minimax strategy is followed against an optimal opponent?**



for node a, value = $(0.2)(9) + (0.8)(5) = 5.8$

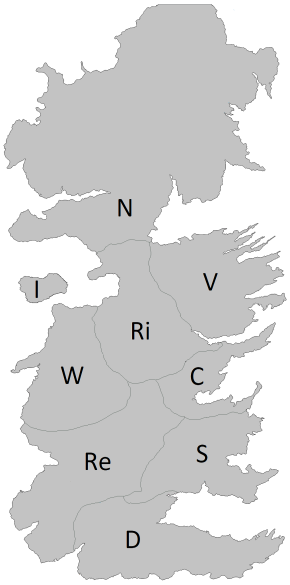
for node b value = $(0.5)(17) + (0.5)(12) = 14.5$

for node c, value = $(0.25)(10) + (0.75)(5.8) = 6.85$

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 17 or 12 (since the max player will select 12 using minimax algorithm). Therefore,
**Highest possible outcome:** 17
**Lowest possible outcome:** 12

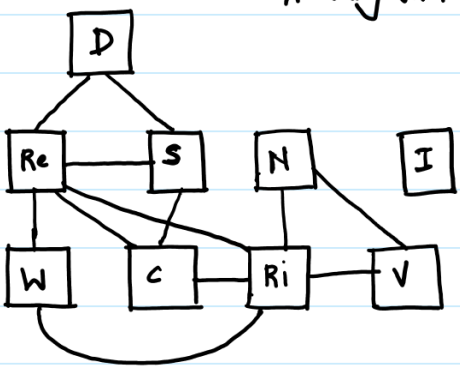
**Task 4: Consider the following map.**




**The problem is to color the sections such that no two sections sharing a border have the same color. You are allowed to use the colors (Red, Green, Blue).**

**Part a (8 points): Draw the Constraint Graph for this problem.**

Assuming that V & N regions are connected for a small portion.

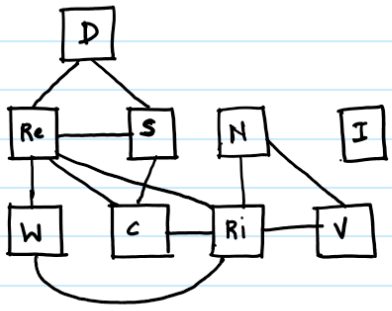



**Part b (10 points): Assuming you are using Backtracking search to solve this problem and that you are using MRV with Degree heuristic to select the variable, Which variable will be selected at each level of the search tree [You do not need to draw the tree. Just let me know which variable will be selected and why (MRV and degree values)]. Note: Multiple possible correct answers. You only have to give one.**

| Node | D | Re | S | W | C | Ri | N | V | I |
|---|---|---|---|---|---|---|---|---|---|
| Degree | ~~2~~ ~~1~~ 0 | 5 | ~~2~~ ~~1~~ 1 | ~~2~~ ~~1~~ 0 | ~~3~~ ~~2~~ ~~1~~ | ~~5~~ ~~4~~ | ~~2~~ 1 | ~~2~~ ~~1~~ 0 | 0 |
| MRV | ~~3~~ ~~2~~ 1 | 3 | ~~3~~ ~~2~~ 1 | ~~3~~ ~~2~~ 1 | ~~3~~ ~~2~~ 1 | ~~3~~ 2 | ~~3~~ 2 | ~~3~~ ~~2~~ 1 | 3 |

| Level | MRV | Highest degree | Node selected. |
|---|---|---|---|
| 1 | 3 | 5 | Re or Ri → Let's select Re |
| 2 | 2 | 4 | Ri |
| 3 | 1 | 1 | C |
| 4 | 1 | 1 | S |
| 5 | 1 | 0 | W or D → Let's select W |
| 6 | 1 | 0 | D |
| 7 | 2 | 1 | N or V → Let's select N |
| 8 | 1 | 0 | V |
| 9 | 3 | 0 | I |

**Part c (12 points): Assume you assign the color 'Red' to the first variable selected in part b. Show the steps involved in checking the remaining legal values for all other variables using Arc Consistency.**




Assign Red to Re.

| Re | D | S | W | C | N | Ri | V | I |
|---|---|---|---|---|---|---|---|---|
| R | ~~RGB~~ | ~~RGB~~ | ~~RGB~~ | ~~RGB~~ | RGB | ~~RGB~~ | RGB | RGB |

1. ~~D → Re~~
   ~~S → Re~~
   ~~C → Re~~
   ~~Ri → Re~~
   ~~W → Re~~
2. ~~S → D~~ NAA
   ~~Re → D~~ NAA
3. ~~Re → S~~ NAA
   ~~D → C~~ NAA
   ~~C → S~~ NAA
4. ~~Re → C~~ NAA
   ~~S → C~~ NAA
   ~~Ri → C~~ NAA
5. ~~N → Ri~~ NAA
   ~~V → Ri~~ NAA
6. ~~Re → W~~ NAA
   ~~C → Ri~~ NAA
   ~~W → Ri~~ NAA
   ~~Ri → W~~ NAA

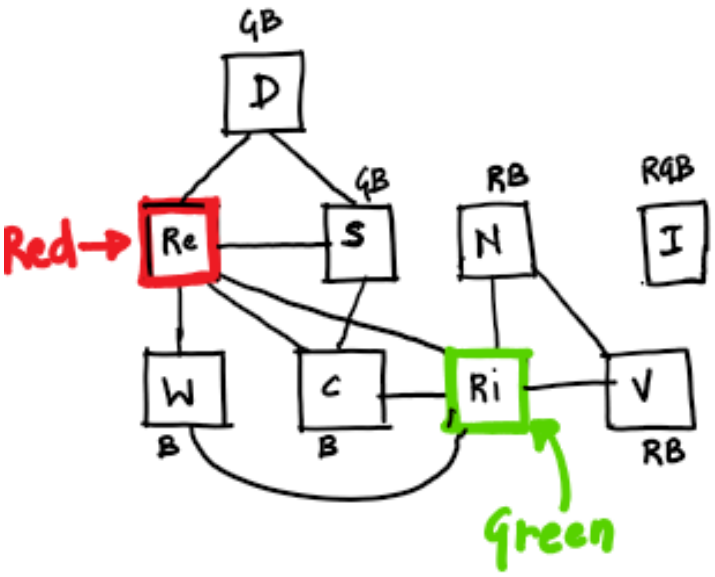
(NAA stands for No Arcs Added)

**Part d (5 points): Can you use the structure of the problem to make solving it more efficient?**

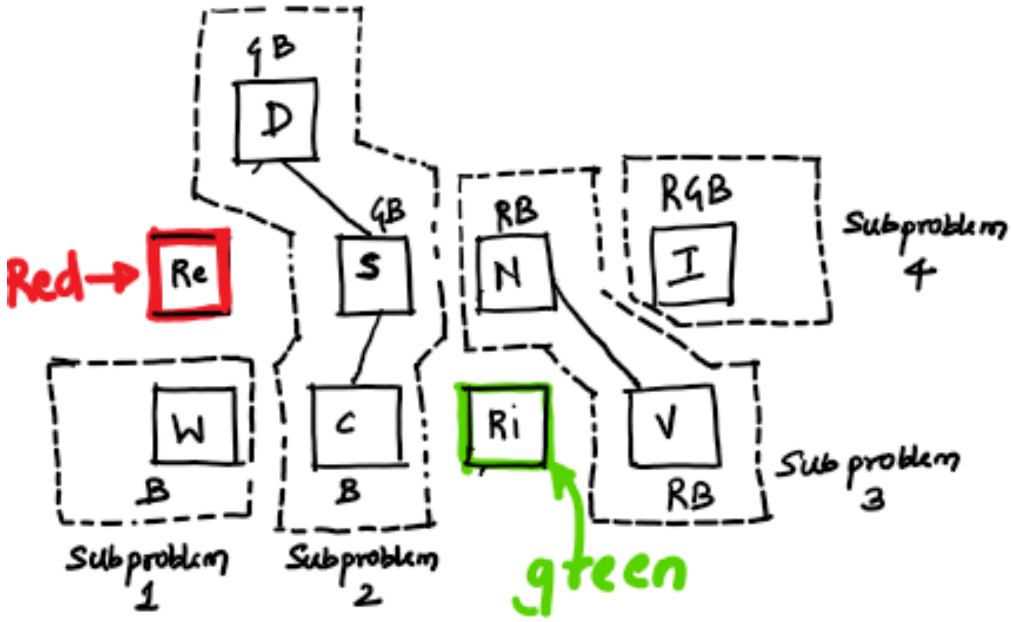
**Solution:**
Yes, we can use the structure of the problem to make solving it more efficient. There are 2 possible ways to do so:

1. First we can see that the region "I" is having no arcs connecting to any other regions. So we can separate it from the main region and consider it as a subproblem and solve both the problems separately and concatenate the final results.
2. We can see that the constraint graph is not acyclic 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 nodes using certain algorithms. Here as the problem is small we can intuitively say that the cut-set here consists of nodes "Re" and "Ri" and assigning values to them can convert the CSG to multiple subproblems that can be solved as trees (shown below).

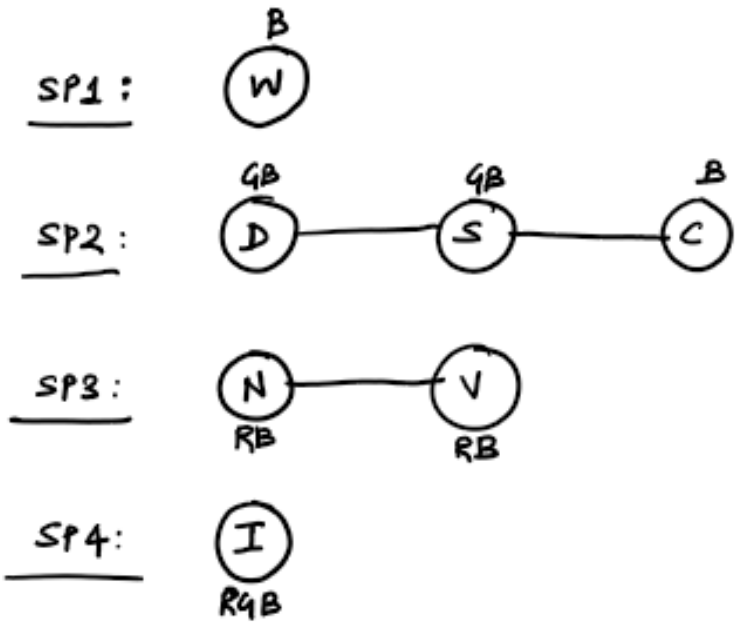
**Assigning values to nodes “Re” and “Ri”:**



**After removing connections of assigned regions:**



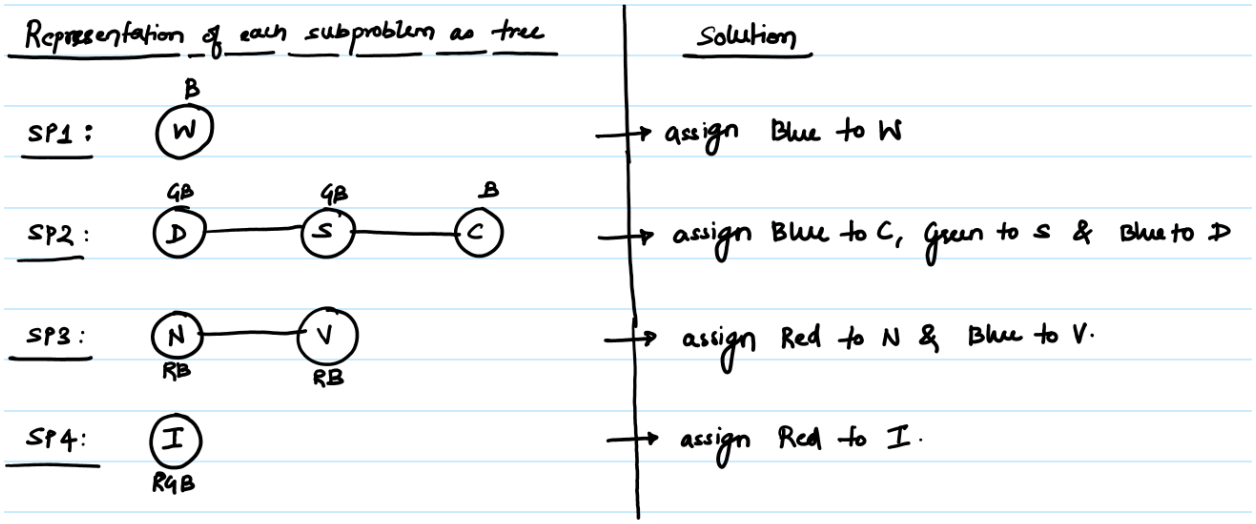

**Representation of each sub problem as tree:**

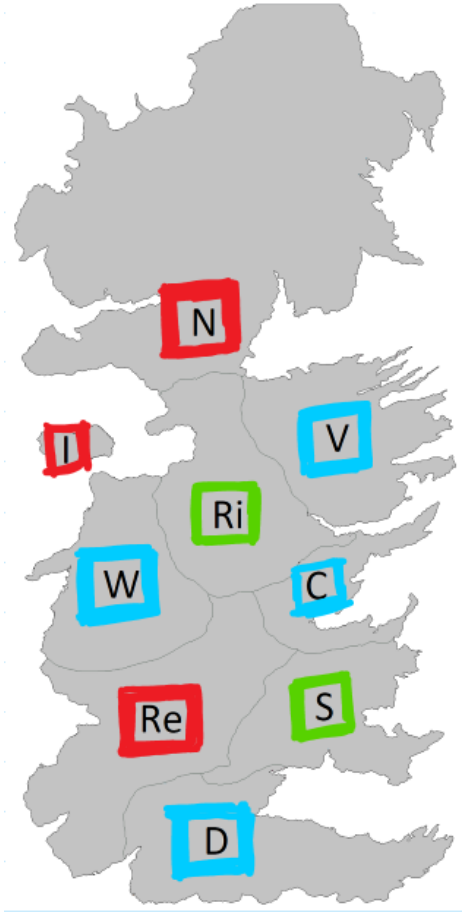



The above subproblems can be easily solved in $O(n\,d^2)$ time.

**Part e: (5 points EC): Give one valid solution to this problem. (You just have to give the solution. No need to give all the steps)**

**Solution:**




Solution:

Red ⇒ (I, N, Re)

Blue ⇒ (W, V, C, D)

Green ⇒ (Ri, S)

**Task 5:**

| A | B | C | KB | S1 |
|---|---|---|---|---|
| True | True | True | True | True |
| True | True | False | False | True |
| True | False | True | True | True |
| True | False | False | False | True |
| False | True | True | False | False |
| False | True | False | False | False |
| False | False | True | True | True |
| False | False | False | False | False |

**KB and S1 are two propositional logic statements that are constructed using symbols A, B, C, and using various connectives. The above truth table shows, for each combination of values of A, B, C, whether KB and S1 are true or false.**

**Part a: Given the above information, does KB entail S1? Justify your answer.**

Yes, KB entails S1.
**Reason:** We can see that the S1 is True for all values where KB is True. Thus we can say that KB entails S1.

**Part b: Given the above information, does statement NOT(KB) entail statement NOT(S1)? Justify your answer.**

| A | B | C | NOT(KB) | NOT(S1) |
|---|---|---|---|---|
| True | True | True | False | False |
| True | True | False | True | False |
| True | False | True | False | False |
| True | False | False | True | False |
| False | True | True | True | True |
| False | True | False | True | True |
| False | False | True | False | False |
| False | False | False | True | True |

**Solution:**
No, NOT(KB) does not entail NOT(S1).
**Reason:** We can see that in row 2 NOT(KB) is True where NOT(S1) is False. That is, NOT(S1) is not True for all values where NOT(KB) is True. Thus we can say that NOT(KB) does not entail NOT(S1).

**Task 6: Suppose that some knowledge base contains various propositional-logic sentences that utilize symbols A, B, C, D (connected with various connectives). There are only two cases when the knowledge base is false:**
**- First case: when A is true, B is false, C is true, D is true.**
**- Second case: when A is false, B is false, C is true, D is false.**

**In all other cases, the knowledge base is true. Write a conjunctive normal form (CNF) for the knowledge base.**

From given 2 statements, we can say that KB is false when,

- First case: when A is true, B is false, C is true, D is true. → $(A \wedge \neg B \wedge C \wedge D)$
- Second case: when A is false, B is false, C is true, D is false. → $(\neg A \wedge \neg B \wedge C \wedge \neg D)$

∴ We can say that KB is true when,

$$\neg [(A \wedge \neg B \wedge C \wedge D) \vee (\neg A \wedge \neg B \wedge C \wedge \neg D)]$$

This is nothing but the required expression for KB.
But it is not in CNF form [rather it is in DNF].

* Convert Non CNF to CNF form:

$$\neg\left[(A \wedge \neg B \wedge C \wedge D) \vee (\neg A \wedge \neg B \wedge C \wedge \neg D)\right]$$

Step 1: Eliminate double implication ($\Leftrightarrow$) or equivalence

No double implication is present in the expression

Step 2: Eliminate implication ($\Rightarrow$)

No implication is present in the expression.

Step 3: Eliminate negation ($\neg$) over brackets. [Using De Morgan's]

$$\neg(A \wedge \neg B \wedge C \wedge D) \wedge \neg(\neg A \wedge \neg B \wedge C \wedge \neg D) \text{ .... using DM}$$

$$(\neg A \vee B \vee \neg C \vee \neg D) \wedge (A \vee B \vee \neg C \vee D) \text{ .... using DM}$$

Step 4: Distribute disjunction ($\vee$) over conjunction ($\wedge$) & flatten.

No distribution or flattening required

Required expression in CNF form:

$$(\neg A \vee B \vee \neg C \vee \neg D) \wedge (A \vee B \vee \neg C \vee D)$$

**Task 7: Consider the KB**

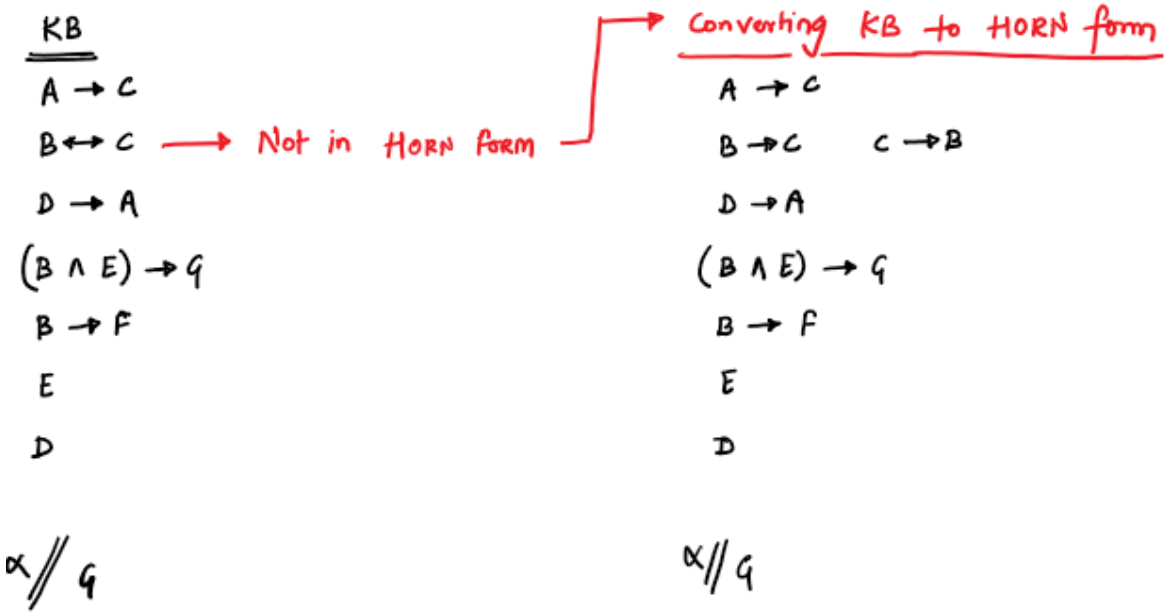
**A => C**
**B <=> C**
**D => A**
**(B AND E) => G**
**B => F**
**E**
**D**

**Show that this entails G (if possible) by**

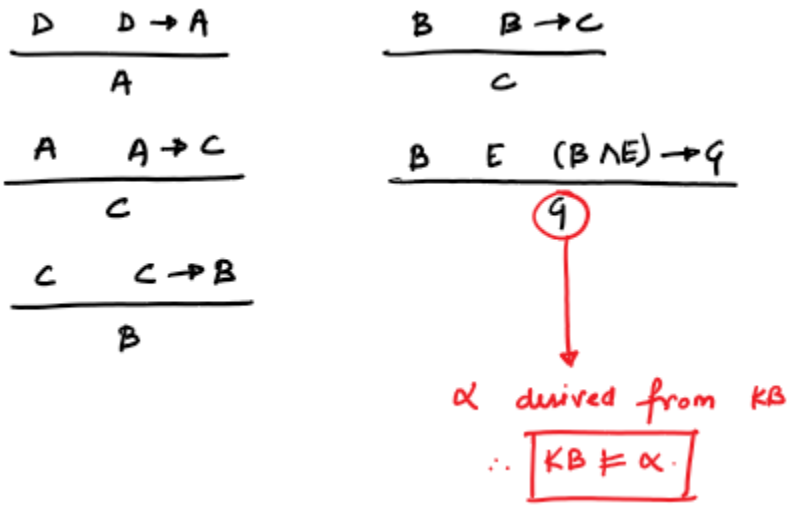
**i. Forward Chaining**
**ii. Backward Chaining**
**iii. Resolution**

**Solution:**

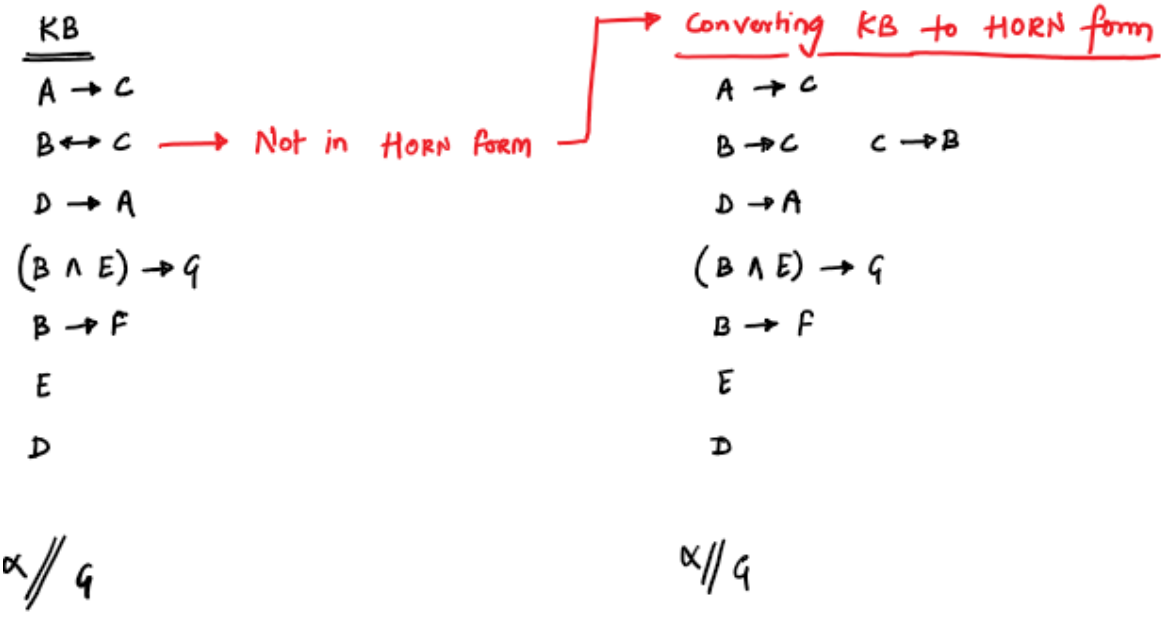
**Forward chaining:** For forward chaining the $KB$ and $\alpha$ both need to be in HORN form.



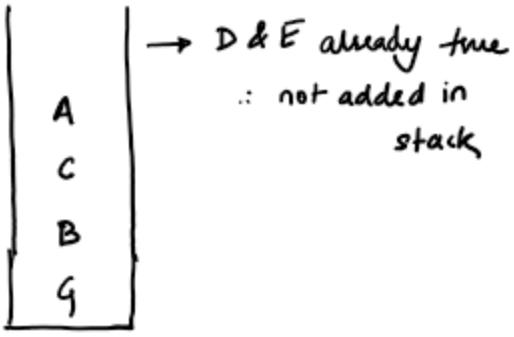
Forward chaining steps:



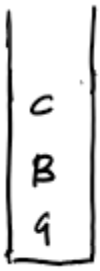

**Backward chaining:** For Backward chaining as well, the $KB$ and $\alpha$ both need to be in HORN form.



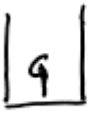

For backward chaining we need to find the dependent literals and add them to the stack. It is done as follows:




**Steps:**

1. $\dfrac{D \quad D \rightarrow A}{A}$ 


2. $\dfrac{A \quad A \rightarrow C}{C}$ 


3. $\dfrac{C \quad C \rightarrow B}{B}$ 


4. $\dfrac{B \quad E \quad (B \wedge E) \rightarrow Q}{Q}$

$\alpha$ desired from KB ∴ $KB \models \alpha$

**Resolution:** The $KB$ must be in CNF form for applying resolution.

$(A \rightarrow C) \wedge (B \leftrightarrow C) \wedge (D \rightarrow A) \wedge ((B \wedge E) \rightarrow G) \wedge (B \rightarrow F) \wedge E \wedge D$ .... (given)

$(\neg A \vee C) \wedge (B \rightarrow C) \wedge (C \rightarrow B) \wedge (\neg D \vee A) \wedge [\neg (B \wedge E) \vee G] \wedge (\neg B \vee F) \wedge E \wedge D$ .... (eliminating $\rightarrow$ & $\leftrightarrow$)

$(\neg A \vee C) \wedge (\neg B \vee C) \wedge (\neg C \vee B) \wedge (\neg D \vee A) \wedge [(\neg B \vee \neg E) \vee G] \wedge (\neg B \vee F) \wedge E \wedge D$.... (eliminating $\rightarrow$ & using DeMorgan's)

$(\neg A \vee C) \wedge (\neg B \vee C) \wedge (\neg C \vee B) \wedge (\neg D \vee A) \wedge (\neg B \vee \neg E \vee G) \wedge (\neg B \vee F) \wedge E \wedge D$.... (dist $\vee$ over $\wedge$ & flattening)

Applying Resolution :

$(\neg A \vee C)$ $(\neg B \vee C)$ $(\neg C \vee B)$ $(\neg D \vee A)$ $(\neg B \vee \neg E \vee G)$ $(\neg B \vee F)$ $E$ $D$ $\neg G$ → complement of $\alpha$ added

$\frac{(\neg A \vee C)(\neg C \vee B)}{(\neg A \vee B)}$ $\frac{(\neg A \vee C)\ (\neg D \vee A)}{(C \vee \neg D)}$ $\frac{(\neg B \vee C)\ (\neg C \vee B)}{(\neg B \vee B)\ (C \vee \neg C)}$ $\frac{(\neg C \vee B)\ (\neg B \vee \neg E \vee G)}{(\neg C \vee \neg E \vee G)}$

$\frac{(\neg C \vee B)\ (\neg B \vee F)}{(\neg C \vee F)}$ $\frac{(\neg C \vee B)\ (C \vee \neg D)}{(B \vee \neg D)}$ $\frac{(\neg D \vee A)\ D}{A}$ $\frac{A\ (\neg A \vee C)}{C}$

$\frac{C\ (\neg C \vee B)}{B}$ $\frac{B\ (\neg B \vee \neg E \vee G)}{(\neg E \vee G)}$ $\frac{E\ (\neg E \vee G)}{G}$ $\frac{G\ (\neg G)}{\square}$

empty clause

∴ We can say using Resolution for proof by contradiction

$KB \vee \neg \alpha \rightarrow$ is unsatisfiable

i.e. $KB \models \alpha$ ($KB$ entails $\alpha$)

**Task 8: John and Mary sign the following contract:**

**- If it rains on Monday, then John must give Mary a check for $100 on Tuesday**
**- If John gives Mary a check for $100 on Tuesday, Mary must mow the lawn on Wednesday.**

**What truly happened those days is the following:**
**- It did not rain on Monday**
**- John gave Mary a check for $100 on Tuesday**
**- Mary mowed the lawn on Wednesday.**

**Part a (10 pts): Write a first order logic statement to express the contract. Make sure that you clearly define what constants and predicates that you use are. (NOTE: DO NOT use functions)**

**Constants:** $J$ - John, $M$ - Mary, $M$ - Monday
**Relations:** $Rains(x)$ - It rains on $x$, $Check(x, y)$ - $x$ gives $y$ $100 check on Tuesday, $Mows(x)$ - $x$ mows lawn on Wednesday.

**Contract:**
If it rains on Monday, then John must give Mary a check for $100 on Tuesday.

$$Rains(Mon) \rightarrow Check(J, M)$$

If John gives Mary a check for $100 on Tuesday, Mary must mow the lawn on Wednesday.

$$Check(J, M) \rightarrow Mows(M)$$

**Part b (8 pts): Write a logical statement to express what truly happened. When possible, use the same predicates and constants as in question 6a. If you need to define any new predicates or constants, clearly define what they stand for.**
**- It did not rain on Monday**
**- John gave Mary a check for $100 on Tuesday**
**- Mary mowed the lawn on Wednesday.**

**Events:**
It did not rain on Monday: $\neg Rains(Mon)$
John gave Mary a check for $100 on Tuesday: $Check(J, M)$
Mary mowed the lawn on Wednesday: $Mows(M)$
**Logical statement:** $(\neg Rains(Mon)) \wedge Check(J, M) \wedge Mows(M)$

**Part c (12 pts): Define the symbols required to convert any KB involved in the above domain from FOL to Propositional logic. Use this to convert the answers to part a and b to Propositional Logic.**

**All possible combinations of relations and constants defined above:**

$A_1 - Rains(J)$, $A_2 - Rains(M)$, $A_3 - Rains(Mon)$

$B_1 - Mows(J)$, $B_2 - Mows(M)$, $B_3 - Mows(Mon)$

$C_1 - Check(J, J)$, $C_2 - Check(J, M)$, $C_3 - Check(J, Mon)$

$C_4 - Check(M, J)$, $C_5 - Check(M, M)$, $C_6 - Check(M, Mon)$

$C_7 - Check(Mon, J)$, $C_8 - Check(Mon, M)$, $C_9 - Check(Mon, Mon)$

**Contract:**
If it rains on Monday, then John must give Mary a check for $100 on Tuesday

$$A_3 \rightarrow C_2$$

If John gives Mary a check for $100 on Tuesday, Mary must mow the lawn on Wednesday.

$$C_2 \rightarrow B_2$$

**Events:**
It did not rain on Monday: $\neg A_3$

John gave Mary a check for $100 on Tuesday: $C_2$

Mary mowed the lawn on Wednesday: $B_2$

**Part d (5 pts): Was the contract violated or not, Justify your answer [Note: Contract is definitely not violated if the events entail the contract. Contract is definitely violated if the events entail the opposite of the contract. Unknown otherwise]**

Here we need to check if events entail contract. The events and contract can be written as follows:
**Events:** $(\neg A_3) \wedge C_2 \wedge B_2$
**Contract:** $(A_3 \rightarrow C_2) \wedge (C_2 \rightarrow B_2)$

Checking if events entail contract using the truth table.

| $A_3$ | $B_2$ | $C_2$ | $(\neg A_3)$ | $(\neg A_3) \wedge C_2 \wedge B_2$ **Events** | $(A_3 \rightarrow C_2)$ | $(C_2 \rightarrow B_2)$ | $(A_3 \rightarrow C_2) \wedge (C_2 \rightarrow B_2)$ **Contract** |
|---|---|---|---|---|---|---|---|
| T | T | T | F | F | T | T | T |
| T | T | F | F | F | T | F | F |
| T | F | T | F | F | F | T | F |
| T | F | F | F | F | F | T | F |
| F | T | T | T | T | T | T | T |
| F | T | F | T | F | T | F | F |
| F | F | T | T | F | T | T | T |
| F | F | F | T | F | T | T | T |

Here we can see that for all values where Events is True, the contract is True as well. Therefore we can say that the event entails contract i.e. $Events \vDash Contract$. And thus the contract is not violated.