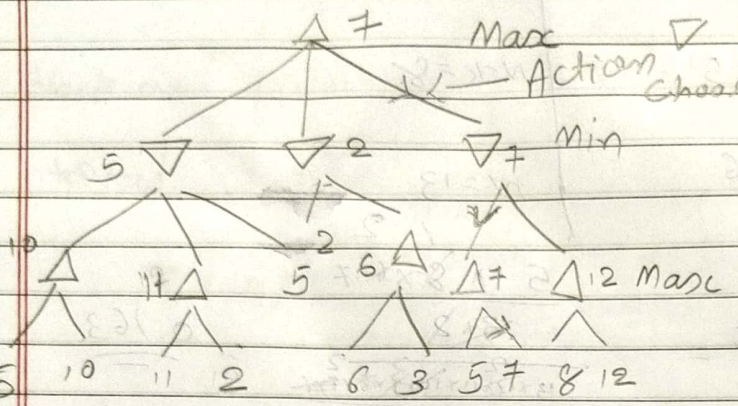
AI

Assignment-2 UTA ID - 1001982478

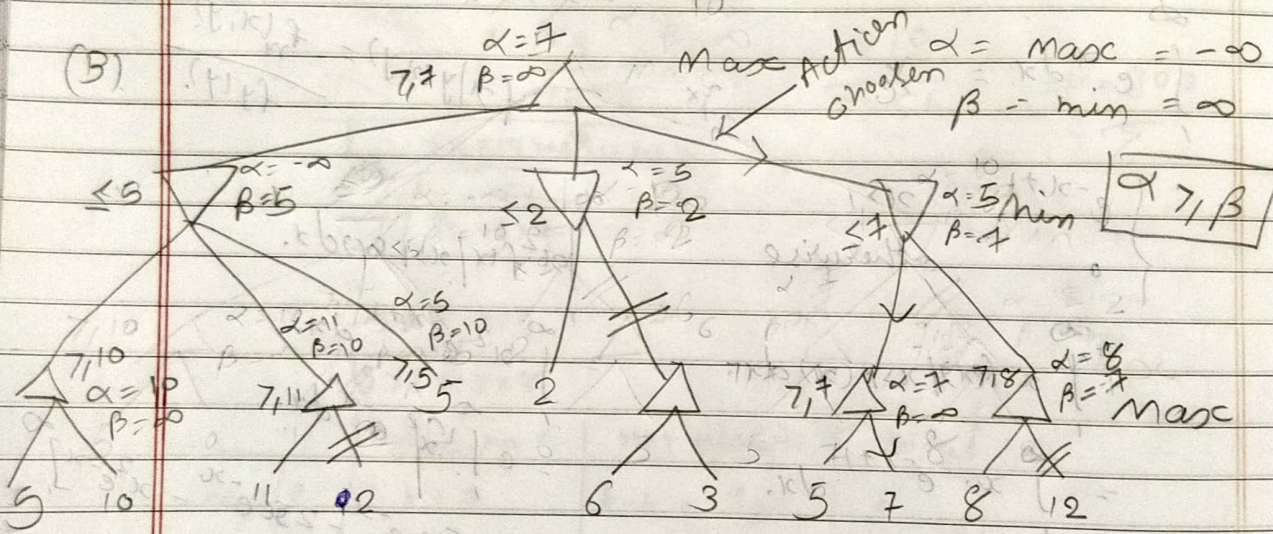
Miracle
Pg.No :
Date : / /201

* Task-1

(A)



(B)

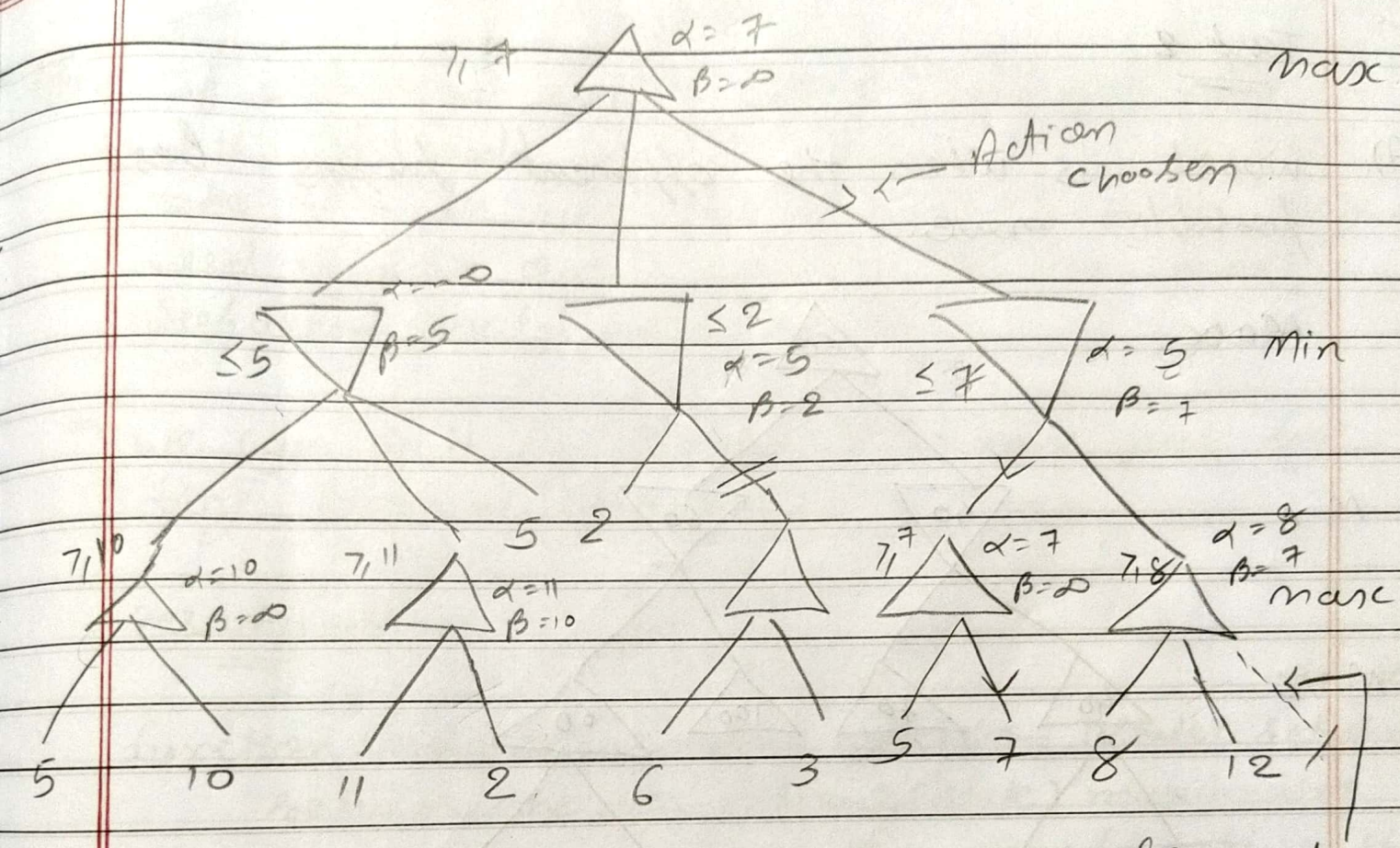


⇒ No, ans is same.

(C) max. utility 12
min " 2

⇒ Than means we can stop exploring a max node after we find child of it returning a value of 12. Similary, we can stop exploring a min node after we find a child of it returning a value of 2.

UTA ID - 1001982478



for α-β pruning,
we maximize α
minimize β
also, if α ≥ β we prune

Given $\alpha \leq 12$
$\beta \geq 2$

ID - 1001982478

Task-2

* If we know exactly what DeepGreen will do with Min node in the search tree do not take to have all the successors. They only need to have one successor for the move DeepGreen would have made

⇒ we can do it by regular min-max just by doing few modifications.

Pseudocode:-

function modified-Minimax-Decision (State) returns an action
    return the a in Actions(State) maximizing Min-value (Result, (a, State))
function Max-value (State) returns a utility value
    if Terminal-Test (State) the return Utility (State)
    v <— - Infinity
    for a, s in Successors (State) do v <— Max (v, Min-value (s))
    return v
function Min-value (State) returns a utility value
    if Terminal-Test (State) the return Utility (State)
    return Max-value (DeepGreen Move (State))

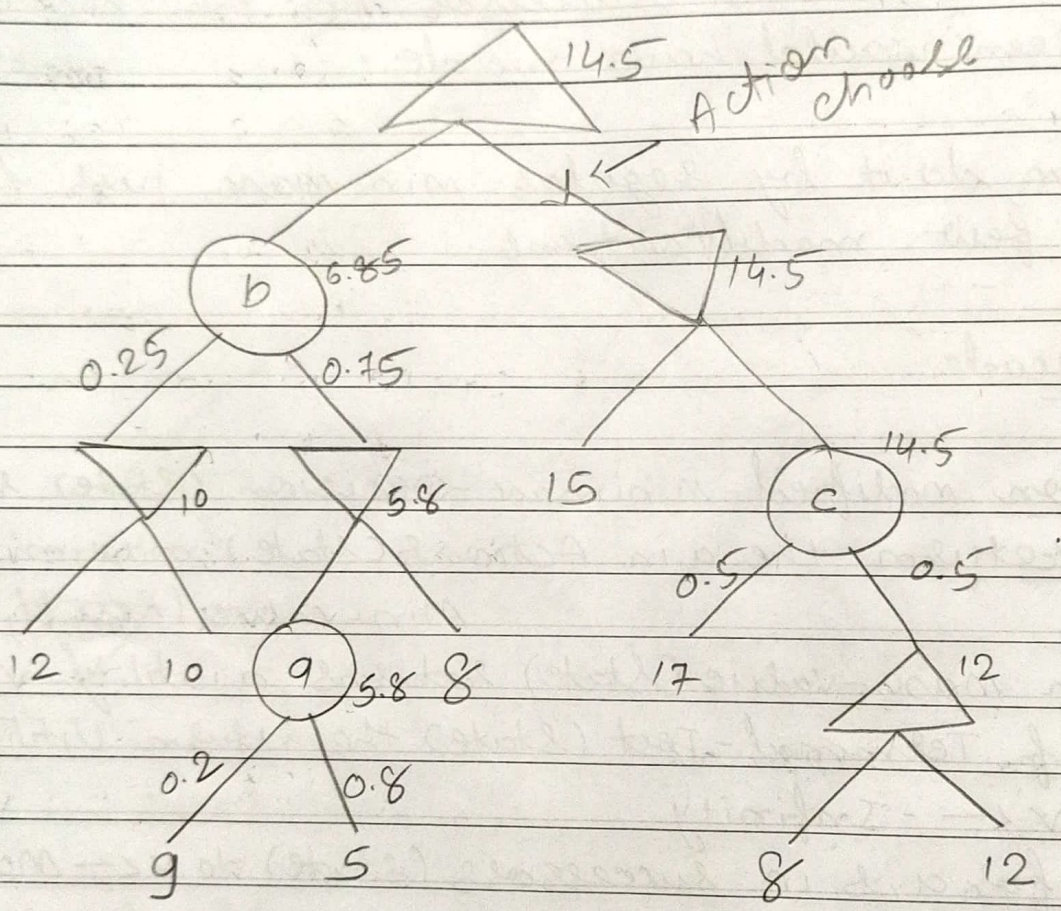
Optimality:-

⇒ If DeepGreen always select optimal move for any state, this will return the exact same move as regular MinMax else it will return move that will result in heighest payoff. This will explore fewer nodes as every MIN node will have only one successor

1001982478

⇒ So, the optimality can't be stated unless we know exact Depth & leaves.

* Task-3



$\therefore a = (0.2)(9) + (0.8)(5) = 5.8$

$\therefore b = (0.25)(10) + (0.75)(5.8) = 6.85$

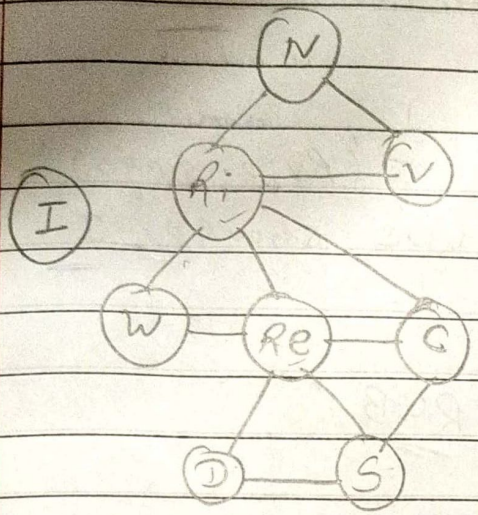
$\therefore c = (0.5)(17) + (0.5)(12) = 14.5$

1001982478

Miracle
Pg.No :
Date : 1 /201

* Task-4

(A). Constraint Graph



(B).

Step-1

N (MRV: 3 Deg 2)
Ri (MRV: 3 Deg 5) ✓
V (MRV: 3 Deg 2)
W (MRV: 3 Deg 2)
Re (MRV: 3 Deg 5)
C (MRV: 3 Deg 3)
D (MRV: 3 Deg 2)
S (MRV: 3 Deg 3)
I (MRV: 3 Deg 0)

⇒ we choose Ri because it has highest degree (5)

Step-2

N (MRV: 2 Deg 2)
V (MRV: 2 Deg 2)
W (MRV: 2 Deg 2)
Re (MRV: 2 Deg 5) ✓
C (MRV: 2 Deg 3)
D (MRV: 3 Deg 2)
S (MRV: 3 Deg 3)
I (MRV: 3 Deg 0)

⇒ Choose Re because highest degree (5)

Step-3

N (MRV: 2 Deg 2)
V (MRV: 2 Deg 2)
W (MRV: 1 Deg 2)
C (MRV: 1 Deg 3) ✓
D (MRV: 2 Deg 2)
S (MRV: 2 Deg 3)
I (MRV: 3 Deg 0)

⇒ we choose C because it has highest degree (3)

Step-4

N (MRV: 2 deg 2)
V (MRV: 2 Deg 2)
W (MRV: 1 Deg 2)
D (MRV: 2 Deg 2)
S (MRV: 1 Deg 3) ✓
I (MRV: 3 Deg 0)

⇒ we choose S because it has highest degree.

Step-5

N (MRV: 2 Deg 2)
V (MRV: 2 Deg 2)
W (MRV: 1 Deg 2) ✓
D (MRV: 1 Deg 2)
I (MRV: 3 Deg 0)

⇒ we choose W because it has highest degree (2)

1001982478

Step-6

N (MRV: 2 Deg 2)
V (MRV: 2 Deg 2)
D (MRV: 1 Deg 2) ✓
I (MRV: 3 Deg 0)
⇒ we choose D ~~because highest degree~~.

Step-7

N (MRV: 2 Deg 2) —
V (MRV: 2 Deg 2)
I (MRV: 3 Deg 0)
⇒ we choose N

Step-8

V (MRV: 1 Deg 2) ✓
I (MRV: 3 Deg 0)
⇒ we choose V

Step-9

I (RV: 3 Deg 0)
we choose I.

(c). Valid solution is :- RGB

Ri → Red
Re → Green
W → Blue
C → Blue
S → Red
D → Blue
N → Green
V → Blue
I → Red

(d). yes, we can use structure of problem to make solving more efficient consider Re & V. Removing this will convert the given problem into a tree structure constraint specific problem which can be solved via cut set conditioning By eliminating loop in a graph, it becomes a tree structure which is easier to solve.

⇒ "I" can be solved as it's own independent subproblem as it is not connected to other nodes.

(e). A - Solution.

	Ri - Green	S - Green
N - Red	V - Blue	D - Blue
I - Green	Re - Red	
W - Blue	C - Blue	

Miracle
Pg.No:
Date: / /201

* Task -5

⇒ Using TT-ENTAILS As described in slides,

CHECK - EQUIVALENCE (KB1, KB2) Returns TRUE or FALSE:
{ If ~~Return~~ TT-Entails?(KB1, KB2) and TT-ENTAILS?(KB2, KB1)
Return TRUE
Else
Return FALSE
}

(A). For all the rows where KB is True, S1 is True.
So, KB1 |= S1.

(B) For all rows where KB is false [not (KB) is true] check if S1 is false [not (S1) is true].
There is atleast one row where this is not the case.
So, not (KB) ⊭ not (S1).

* Task-6

⇒ Two cases where knowledge base is false:-
$\neg(A \wedge \neg B \wedge C \wedge D) \wedge \neg(\neg A \wedge \neg B \wedge C \wedge \neg D)$
converting to CNF ⇒
using Demorgan law to substitute '∧' inside paranthesis to '∨'
$(\neg A \vee \neg(\neg B) \vee \neg C \vee \neg D) \wedge (\neg(\neg A) \vee \neg(\neg B) \vee \neg C \vee \neg(\neg D))$

⇒ Applying double negation:-

final CNF:-

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

* Task-7 ✓

→ To show that this knowledge base entails G

$A \Rightarrow C$

$B \Leftrightarrow C = (B \rightarrow C) \wedge (C \rightarrow B)$

$D \Rightarrow A$

$(B \text{ AND } E) \Rightarrow G$

$B \Rightarrow F$

E

D

we convert to Horn Form (HF)

(i). Forward chaining:-

→ Apply modus ponens (MP) to $D \Rightarrow A$, D added A to KB
Apply MP to $A \Rightarrow C$, A added C to KB
Apply MP to $C \Rightarrow B$, C added B to KB
Apply MP to $B \text{ AND } E \Rightarrow G$, B, E added G to KB
- Thus we show $KB \models G$

Explore list is {D, A, C, B, E, G}

(ii). Backward chaining:-

→ To show G, we need B & E to be true. where C is already True.

⇒ we initialize Goal state as GS

GS | G | , we get by B & E ⇒ G.

GS | B | by C → B.
| E |
| G |
,

GS | C | , by A ⇒ C
| B |
| E |
| G |

GS | A | , by D ⇒ A
| C |
| B |
| E |
| G |

GS | D | where D is already True.
| A |
| C |
| B |
| E |
| G |

Thus, by using MP, D ⇒ A i.e. A is True.
by using MP, A ⇒ C i.e. C is True.
by using MP, C → B i.e. B is True.
by using MP, B & E ⇒ G i.e. G is True.

∴ KB ⊨ G where KB entails G.

Miracle
Pg.No :
Date : / /201

1001482478

(iii). Resolution:-

⇒ Resolution Algo. is $KB \wedge \neg\alpha$
It's unsatisfiable

$\therefore (A \Rightarrow C) \wedge (B \Leftrightarrow C) \wedge (D \Rightarrow A) \wedge [(B \wedge E) \Rightarrow G] \wedge (B \Rightarrow F)$
$\wedge E \wedge D \wedge \neg G$

convert to CNF:-

$\therefore (A \Rightarrow C) \wedge (B \Rightarrow C) \wedge (C \Rightarrow B) \wedge (D \Rightarrow A) \wedge [(B \wedge E) \Rightarrow G]$
$\wedge (B \Rightarrow F) \wedge E \wedge D \wedge \neg G$

$\therefore$ eliminate "$\Rightarrow$" $\alpha \Rightarrow B \therefore \neg\alpha \vee B$.

$\therefore (\neg A \vee C) \wedge (\neg B \vee C) \wedge (\neg C \vee B) \wedge (\neg D \vee A) \wedge [\neg(B \wedge E) \vee G]$
$\wedge (\neg B \vee F) \wedge E \wedge D \wedge \neg G$

move '$\neg$' inside by De Morgan's law.

$\therefore (\neg A \vee C) \wedge (\neg B \vee C) \wedge (\neg C \vee B) \wedge (\neg D \vee A) \wedge (\neg B \vee \neg E \vee G)$
$\wedge (\neg B \vee F) \wedge E \wedge D \wedge \neg G$

$\neg G \quad \neg B \vee \neg E \vee G$

$\neg B \vee \neg E$

$\neg B \vee \neg E \quad \neg C \vee B$

$\neg E \vee \neg C$

$\neg E \vee \neg C \quad \neg A \vee C$

$\neg E \vee \neg A$

Scanned with CamScanner

Miracle
Pg.No :
Date : / /201

$\neg E \cup \neg A \quad E$

$\neg A$

$\neg A \quad \neg D \cup A$

$\neg D$

$\neg D \quad D$

$\square \rightarrow$ empty clause.

So, $KB \models G$.

1001982478

* Task-8

(A). Constants:- John, Mary, Monday.

Predict:-

- Give100 (x, y) - x gives y a check for 100$ on Tuesday.
- Rain (x) - If rains on x
- mow (x) - x mows lawn on wednesday.

The contract:-

Rain (monday) ⇒ Give100 (John, mary)
Give100 (John, mary) ⇒ mow (mary)

(B). what truly happend:-

It did not rain on monday - ¬Rain(monday)
John gave mary a check for 100$ on Tuesday - Give100 (John, mary)
mary mowed the lawn on wednesday - mow (mary)

Logical statement:-

¬Rain (monday) ∧ Give100 (John, mary) ∧ mow (mary)

(C). The symbols are:-

S_1: Rain (John)
S_2: Rain (mary)
S_3: Rain (monday)
S_4: Give100 (John, John)
S_5: Give100 (John, mary)
S_6: Give100 (John, monday)

S_7: Give100(mary, John) S_{12}: Give100(monday, mary)
S_8: Give100(mary, mary) S_{13}: now(John)
S_9: Give100(mary, monday) S_{14}: now(mary)
S_{10}: Give100(monday, monday) S_{15}: now(monday)
S_{11}: Give100(monday, John)

The contract in propositional logic

$S_3 \Rightarrow S_5$

$S_5 \Rightarrow S_{14}$

$\Rightarrow$ So, truly happened events in propositional logic

$\neg S_3 \cap S_5 \cap S_{14}$

(d). As there is no scenario where the events are true and the contract is false, we can say that the contract is not violated. which means that, for all the scenarios where the events are true, the contract is also true. i.e. Events $\models$ Contract. Hence, the contract is not violated.