Camlin Page 11
Date / /

CD : 02

27/03/2023

ADVANCED SEARCH AND GAME PLAYING.

Advanced AI techniques:

- AND/OR graphs
- solution strategy for AND/OR graphs *
- Game playing
  - Game trees *
  - Minimax Algorithm * } (✓✓)
  - Alpha Beta Pruning * (Chess play strategy) 15m (✓✓)
- constraint satisfaction problem (CSP) *]
- solution strategy for CSP / case study. (One case study)
- Genetic Algorithms } Non-classical AI Algorithm.
- simulated Annealing } Nature inspired Algorithm

Russell & Norvig

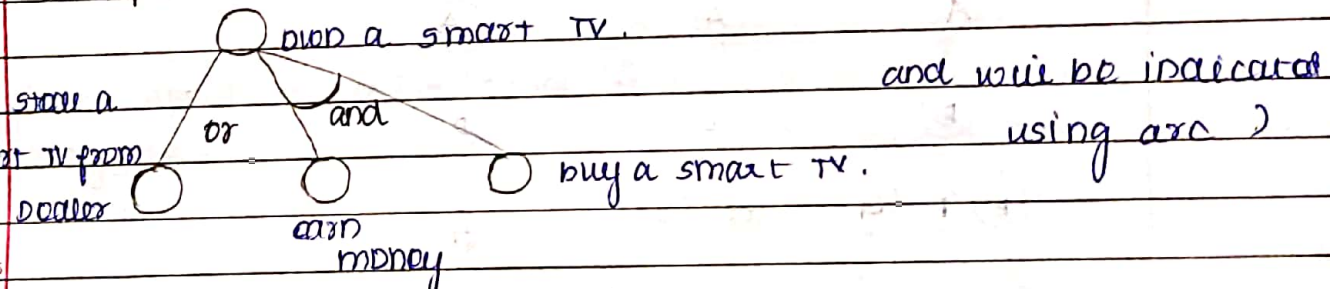
* - Problem oriented topics

AND/OR Graphs:

↳ graphical model for real world AI problems.

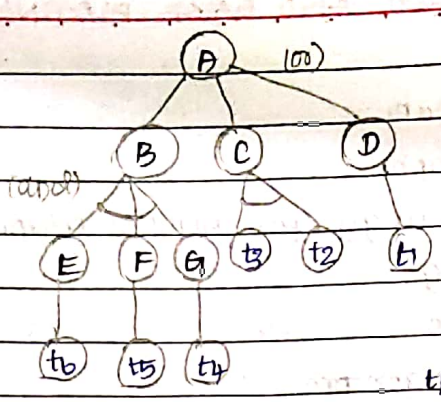
Directed graph / Digraph & undirected graph.

Example:



and will be indicated using arc)

AND/OR graph is a combination / mixture of all edges are of OR edges and AND edges



A, B, C, D, E, F, G } non terminal nodes

t1, t2, t3, t4, t5, t6 } terminal nodes

28/12/2023

⇒ All terminal nodes are solvable nodes

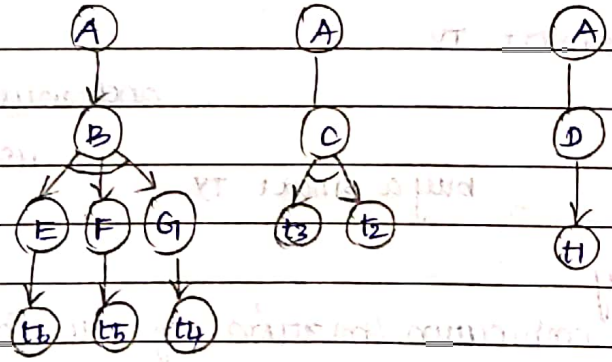
⇒ All non terminal nodes are looking for a solution.

unsolved nodes are the non terminal nodes, present at leaf node they cannot be solved.

Solution graph of an AND/OR graph:-

Extraction of an AND/OR graph

(subgraph of an AND/OR graph)



multiple solutions on the or edge

An AND/OR graph models a real world scenario using nodes and edges like a traditional graph. However, the edges can be an AND edge or OR edge depending on the real world scenario.

The nodes of the graph indicates events of the real world.

A solution graph is a subgraph or partial graph of the AND/OR graph in which all the terminal nodes are solved nodes. / solvable nodes

The rest of the nodes are called as non terminal nodes.

Characteristics of AND/OR graph :-

(1) Terminal nodes are solved nodes

(2) If the child nodes of any given node are AND nodes then that node is solved with when all those AND nodes are solved.

(3) If the child nodes of any given node are OR nodes, then that node is solved if any one of the OR nodes are solved.

(4) A solution graph is a part of the AND/OR graph all of whose nodes are solved and in which no node including the start node has more than one edge to an OR node.

(X) (X) (X)

Depth First search for extracting solution graph of an AND/OR graph :-

PROCEDURE DFS_AND/OR :
Begin
1. Put into OPEN list an unsolved graph consisting only of the start node.

2. LOOP : If OPEN = EMPTY then EXIT(FAIL)

3. P = FIRST(OPEN) /* Fetch the first element (unsolved graph) from the OPEN list */

4. If SOLVED (P) then EXIT(SUCCESS)
/* If P is a solved graph the search has succeeded and P is the solution */

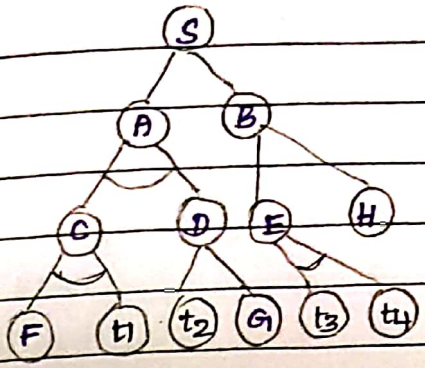
5. REMOVE (P, OPEN)

6. Expand P, Evaluate all the newly generated unsolved graphs and put any unsolved graphs whose start nodes are not unsolved at the head of OPEN list (top of stack)

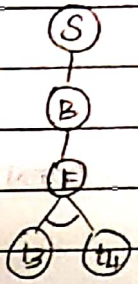
7. goto LOOP

END DFS_AND/OR

The following AND/OR graph models a real world problem. Trace DFS to obtain a solution graph to solve the real world problem. Observe that a AND/OR graph handles nodes which are not solvable due to uncertainity.



solution graph::



tracing of DFS_AND/OR ::

OPEN

1. S

OPEN ≠ EMPTY

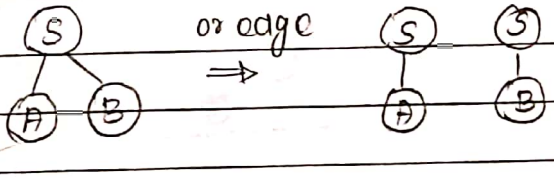
P = S

SOLVED (P) × false

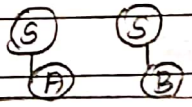
REMOVE (P, OPEN)

OPEN

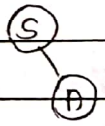
Expand P



OPEN



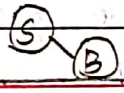
2. OPEN ≠ EMPTY

P = 

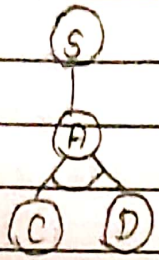
SOLVED (P) × false

OPEN

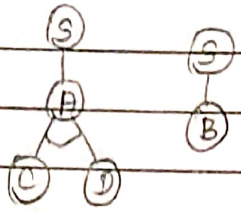
REMOVE (P, OPEN)



Expand P

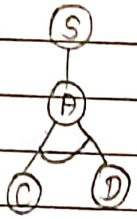


OPEN



8. OPEN ≠ EMPTY

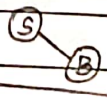
P:



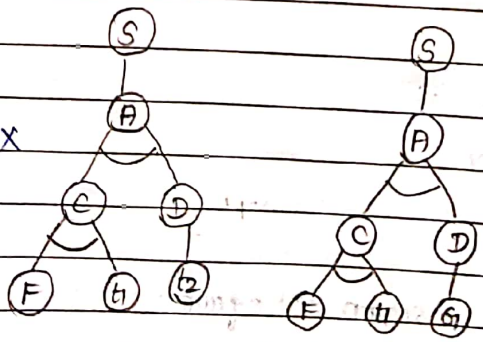
SOLVED(P) x false

REMOVE (P, OPEN)

OPEN



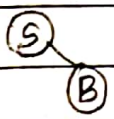
Expand P



since both the sub graph in P cannot be solved, they cannot be pushed into OPEN list

since terminal node present at the leaf that cannot be solved

OPEN



4. OPEN ≠ EMPTY

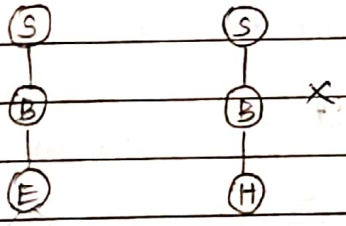
P: 

solved (P) ✗ false

OPEN

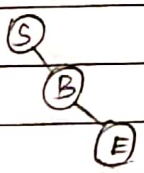
REMOVE (P, OPEN)

Expand P



sub graph is unsolved, thus it cannot be pushed into OPEN list.

OPEN



5. OPEN ≠ EMPTY

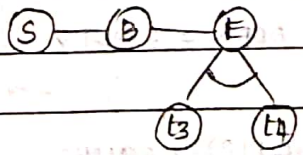
P: 

solved (P) ✗ false.

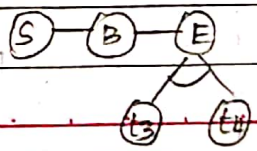
OPEN

REMOVE (P, OPEN)

Expand (P)

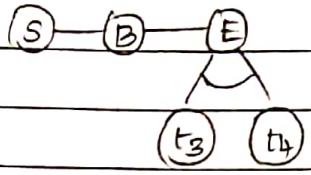


OPEN



b. OPEN ≠ EMPTY

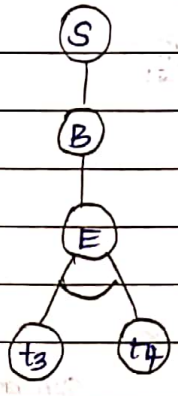
P:



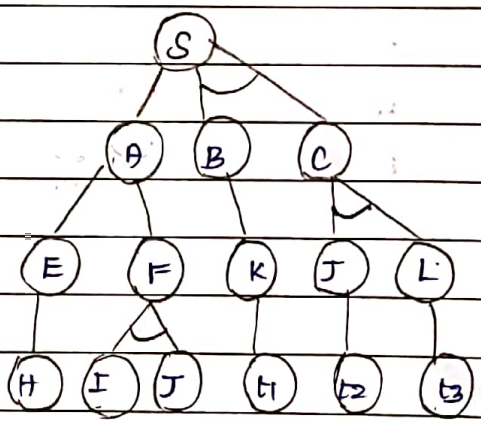
SOLVED (P) → EXIT (SUCCESS)

Algorithm terminates.

Solution graph for the given AND/OR graph :



29/03/2023 Obtain a solution graph for the following AND/OR graph



Tracing of OFS-AND/OR graph :

OPEN

1.

S

OPEN ≠ EMPTY

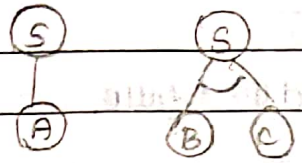
P: S

solved (P). × false

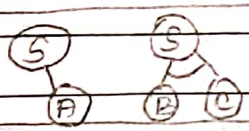
REMOVE (P, OPEN)

OPEN

Expand P



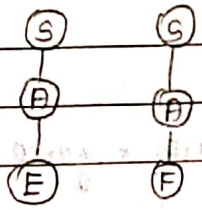
OPEN

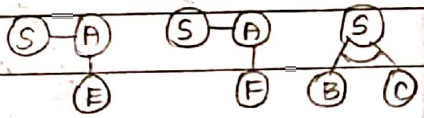


2. OPEN ≠ EMPTY

P:  Solved (P) ← false

REMOVE (P, OPEN)

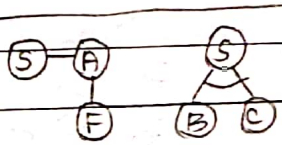
Expand (P) 

OPEN 

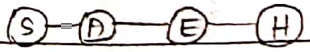
3. OPEN ≠ EMPTY

P:  Solved (P) ← false

REMOVE (P, OPEN)

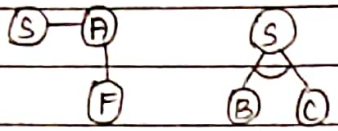


Expand P



Since the graph cannot be solved, it is not pushed inside OPEN.

4. OPEN



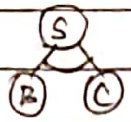
OPEN ≠ EMPTY.

P: 

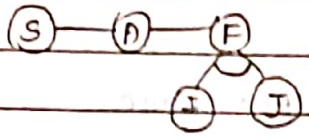
solved (P) = false

Remove (P, OPEN)

OPEN

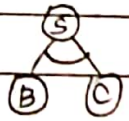


Expand P

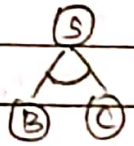


since the subgraph is cannot be solved, it is not pushed inside OPEN.

5. OPEN



OPEN ≠ EMPTY

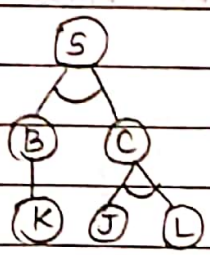
P: 

solved (P) = false

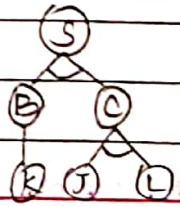
Remove (P, OPEN)

OPEN

Expand P

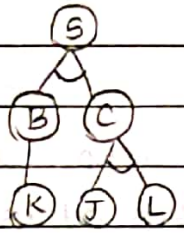


OPEN



6. OPEN ≠ EMPTY

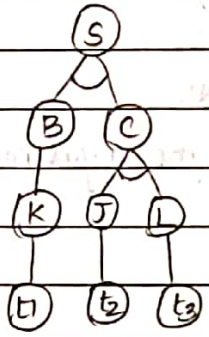
P :



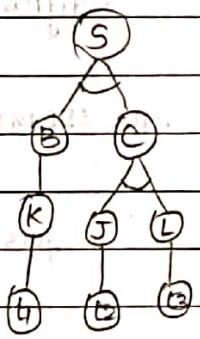
Solved (P) × false

Remove (P, OPEN) OPEN

Expand P

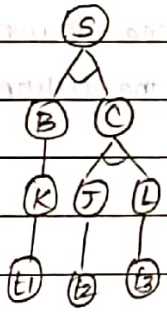


OPEN



7. OPEN ≠ EMPTY

P :

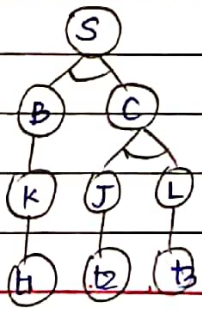


Solved (P) ✓ true

→ EXIT (SUCCESS)

Algorithm terminates.

Solution for the given AND/OR graph :



Russel & Norving.

Game Playing :-

Game types :- History of AI
↳ Game playing in AI
CHESS / Chinese checkers

Game tree is a data structure which is a tree, consisting of all possible moves of the game which allows the method to move from one state of the game to the next state.

A game tree comprises the following :-

(i) ~~Initial state~~ INITIAL STATE :-
This is the initial BOARD position of the game.

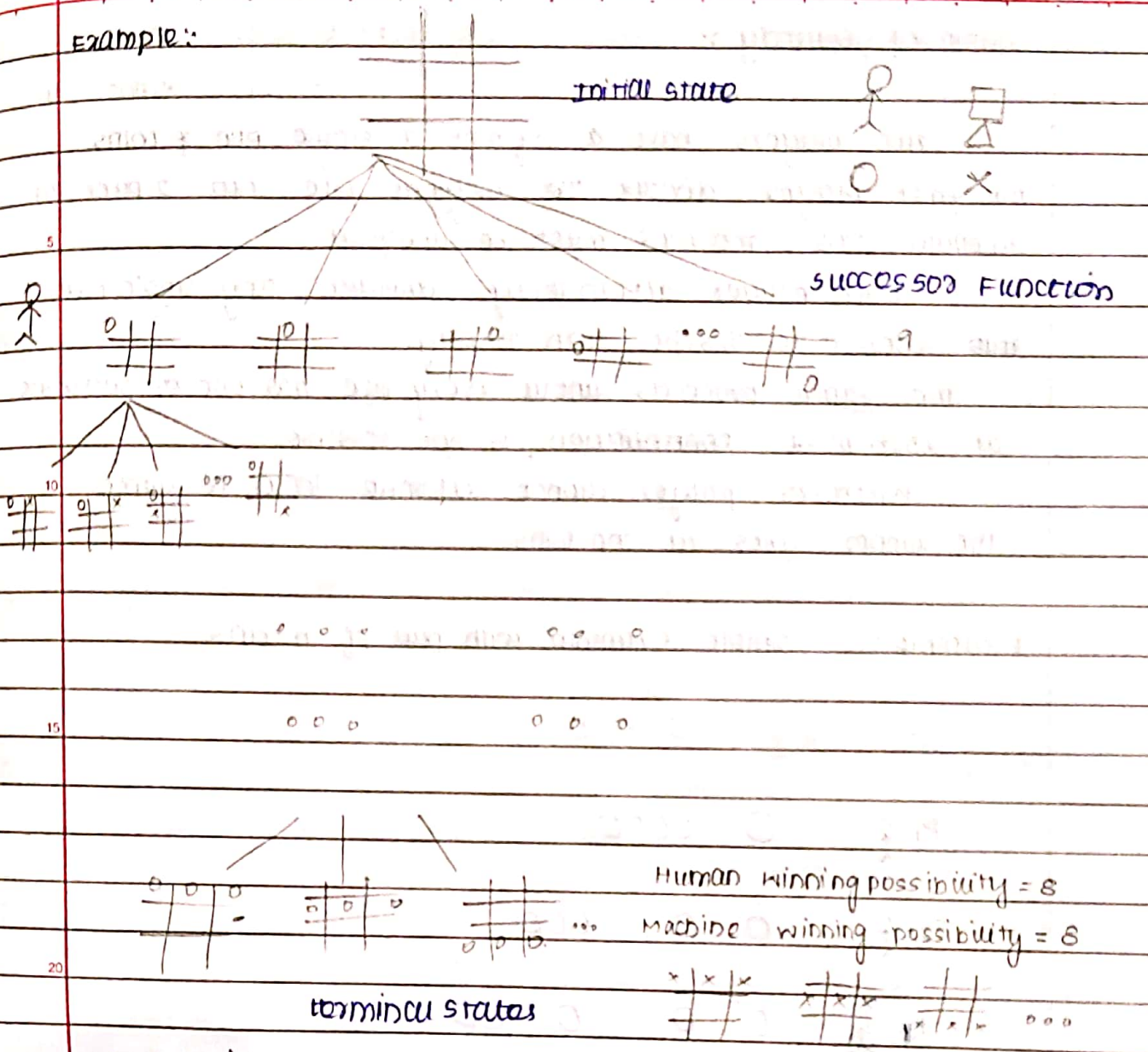
(ii) SUCCESSOR FUNCTION :-
This defines the legal moves of the game by the players

(iii) TERMINAL STATE :-
This is the board position when the game gets over. (WIN / LOSS / DRAW)

(iv) UTILITY FUNCTION / PAYOFF FUNCTION :-
This defines a numeric value for the outcome of the game. Utilities of the terminal nodes are used to determine the utilities of other nodes

31/03/2023

Example:



payoff / utility function } needed for machine.

Game of Grundy :-

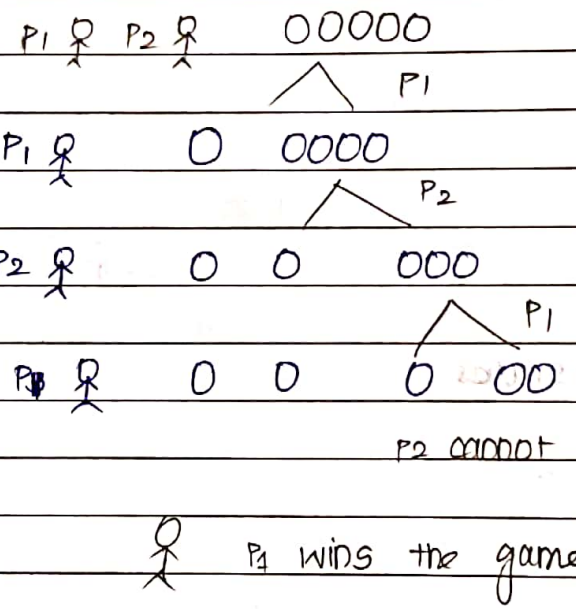
Two players have a in front a single pile of coins the first player divides the original pile into 2 piles of unequal coins. The piles must be unequal.

Each player alternately divides any single pile when it is his/her turn to play.

The game proceeds until every pile has one or two coins at which point continuation is not possible

Whichever player cannot continue looses the game. The winner gets all the coins.

Example :- simple example with pile of 5 coins



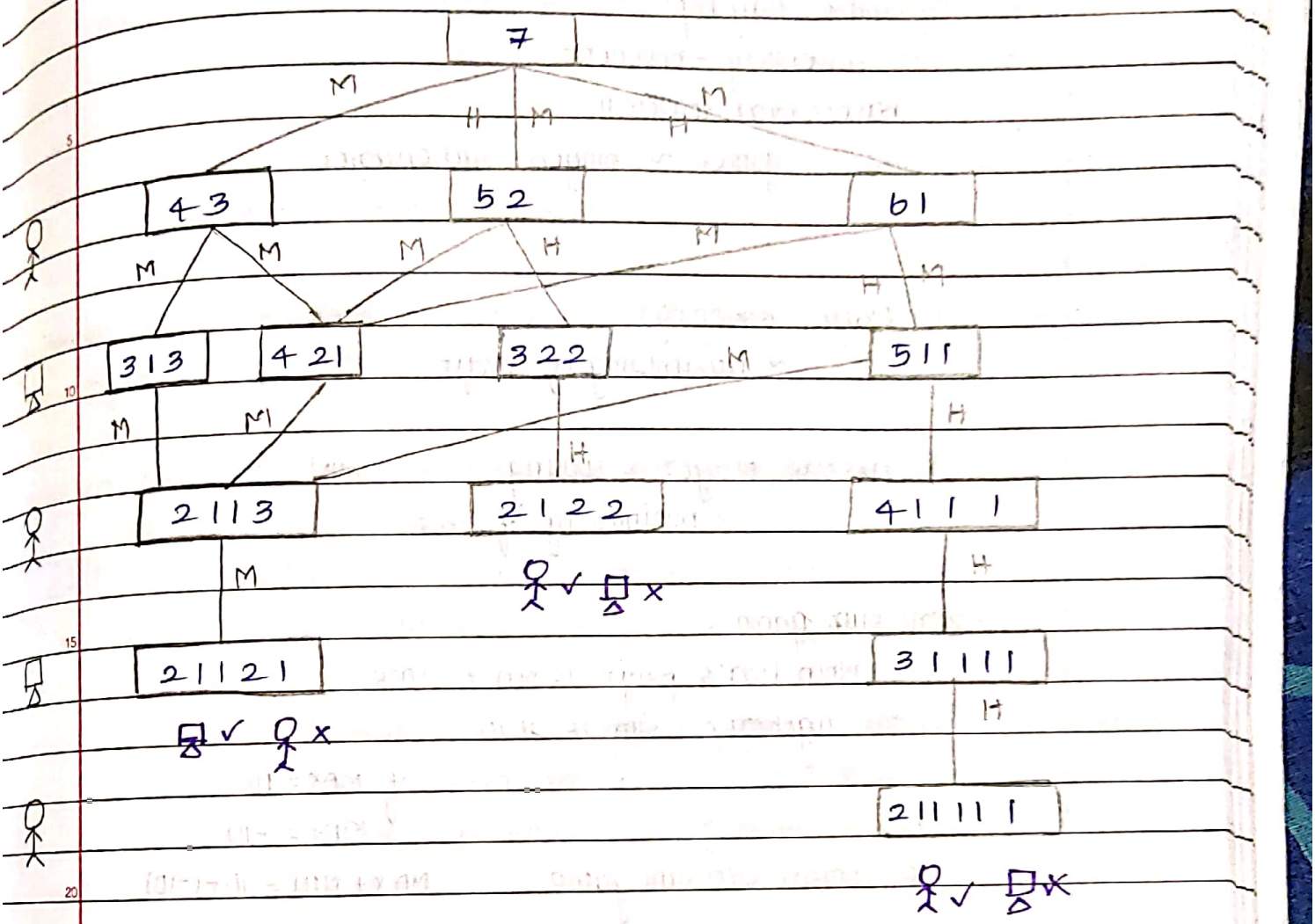
P2 cannot further play.

P1 wins the game

(:X:) Game Tree cannot have duplicate nodes (:X:)

Camlin Page 125
Date / /

Trace a game tree for the game of grundy for the pile of 7 coins



01/04/2023

robot player (soccer play) ⌒ robotics

Adversarial search :: (Game tree)

↳ Game playing
↳ adversary - opponent
↳ adversarial search in
games ~ players ~ adversaries
(players with opposite interest)

↳ Player ~ winning ~ maximizing of profit — MAX ← player maximizing profit

↳ opposite player ~ loosing ~ minimizing of loss — MIN ← player minimize loss

zero sum game :
When Max's profit is Min's loss,
the algebraic sum is zero

if MAX = 10
MIN = -10
MAX + MIN = 10 + (-10)
= 10 - 10 = 0.

Two person playing
Two-person zero sum game

Noble prize winning on Economics

Game theory :: Von Neumann
Economics /
operation research)

MINIMAX STRATEGY :

In this strategy two players MAX and MIN are involved where MAX always moves first (opening player) the minimax strategy has the following elements :

S_0 : Initial state which specifies how the game is setup and the start.

PLAYER(s) : Defines which player has the move in a state.

ACTION(s) : Returns the set of legal moves in a state

RESULT(s, a) : The transition model which defines the result of a move.

TERMINAL-TEST(s) : A terminal test which is true when the game is over and false otherwise. States where the game has ended are called terminal states

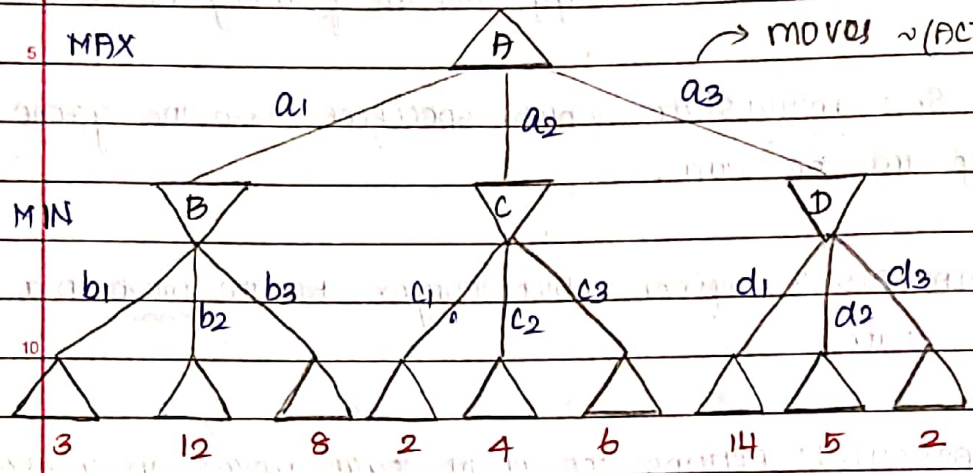
UTILITY(s, p) : A utility function (also called objective function / payoff function) defines the final numeric value for a game that ends in a terminal state s, for a player p.

NOTE :

The initial state, action function ACTION(s), result function RESULT(s, a) (S_0) define a game tree for the two person zero sum game. Thus, the game tree has game states as the nodes and moves are the edges

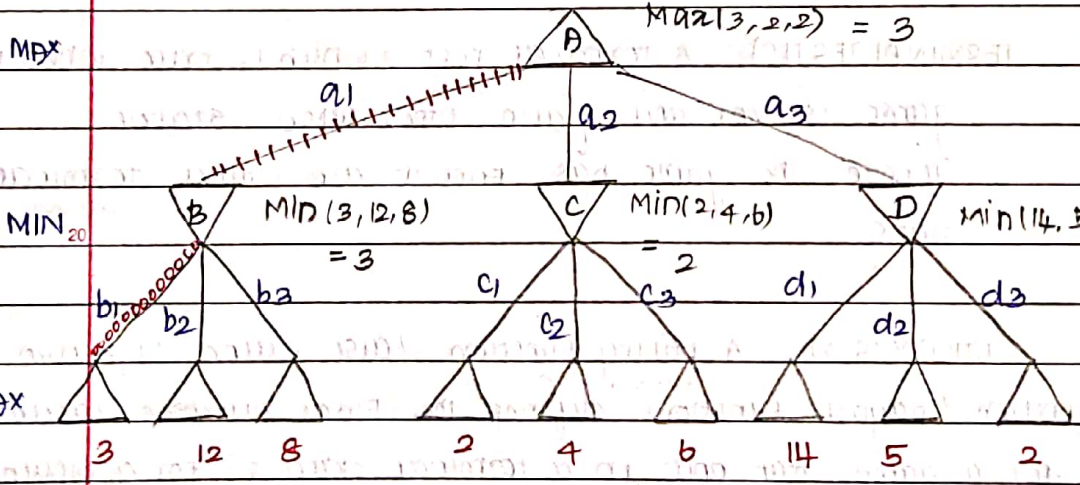
(X) MAX △ MIN ▽

The following is an example of a game tree with two players MAX and MIN, playing a zero sum game.



03/04/2022 Minimax strategy :

2-ply game tree



The MINIMAX strategy is controlled by the following equation:

$$MINIMAX(S) = \begin{cases} UTILITY(S) & \text{IF TERMINAL TEST(S) is true} \\ \max_{a \in ACTIONS(S)} (MINIMAX(RESULT(S,a))) & \text{IF PLAYER(S) = MAX} \\ \min_{a \in ACTIONS(S)} (MINIMAX(RESULT(S,a))) & \text{IF PLAYER(S) = MIN} \end{cases}$$

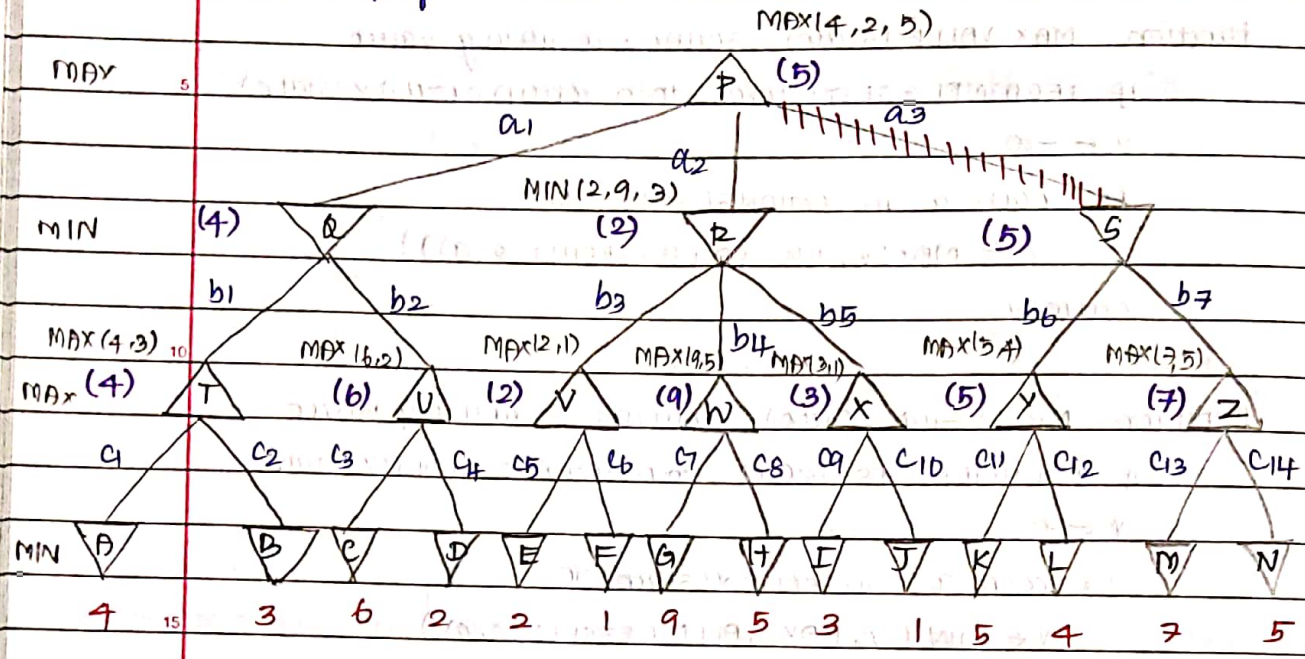
function MINIMAX-DECISION (state) returns an action
return $\arg\max_{a \in ACTIONS(s)}$ MIN-VALUE (RESULT(state, a))

function MAX-VALUE (state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY(state)
$v \leftarrow -\infty$
for each a in ACTIONS(state) do
$v \leftarrow MAX(v, MIN\text{-}VALUE(RESULT(s, a)))$
return v.

function MIN-VALUE (state) returns a utility value
if TERMINAL-TEST (state) then returns UTILITY(state)
$v \leftarrow \infty$
for each a in ACTION(state) do
$v \leftarrow MIN(v, MAX\text{-}VALUE(RESULT(s, a)))$
return v.

An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves to determine the backed-up values of a state. The notation $\arg\max_{a \in S} f(a)$ computes the element a of set S that has the maximum value of $f(a)$.

For the following game tree (3 ply) trace MINIMAX strategy and obtain the strategy played by MAX, to maximize his profit



Strategy played by MAX is a3 with minimum profit of 5

MIN
b1
MAX
3

If you want to draw the game tree for chess game parameters are :-

The average no of branches = 35

In a every series of game, the average no of moves of player = 50 [For 100 players - 100 moves]

No. of nodes $\Rightarrow 35^{100}$ nodes

approximately 10^{10} terminal nodes.

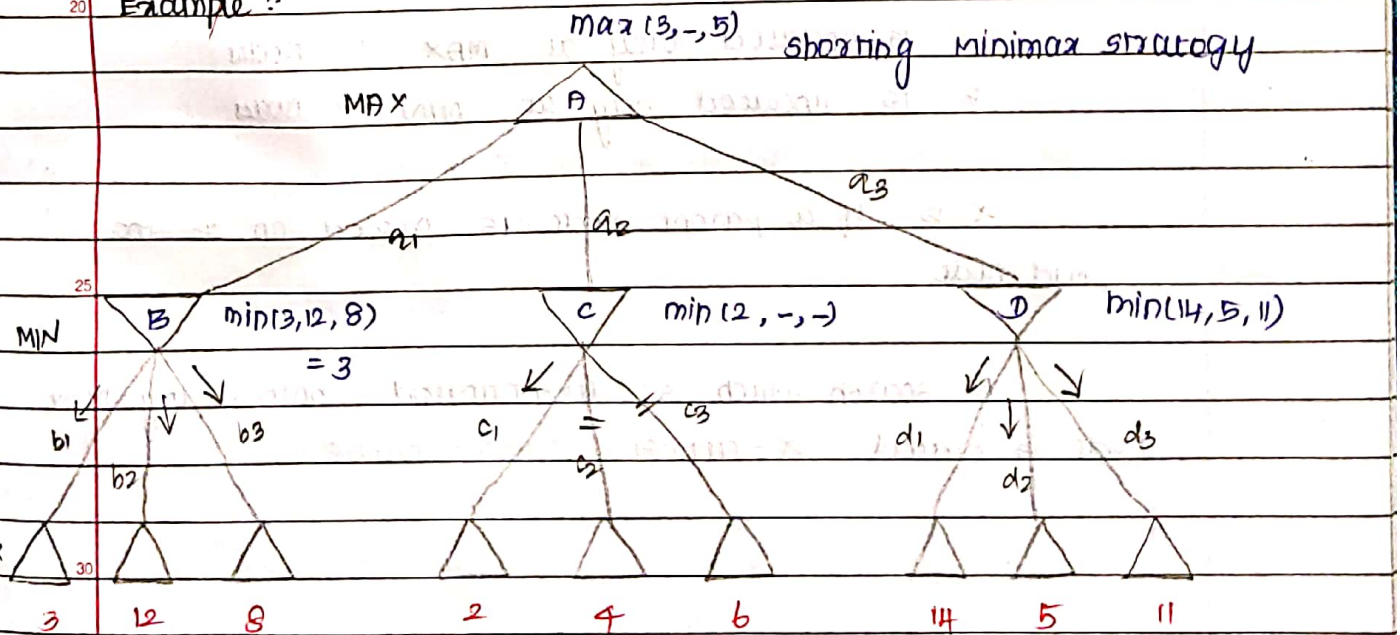
In general the game tree can be humongous.

conserving two persons zero sum game, the minimax strategy explores the complete game tree before it finds the strategy for MAX

Time constraint is very large

The Minimax strategy has the disadvantage that as it goes through all the nodes, and its an time consuming task.

Example :-



Showing Minimax strategy

Discard is what is called as pruning

The minimax strategy which incorporates pruning is called alpha-beta pruning.

Alpha and Beta are two parameters which helps the implement the logic behind pruning.

WORK RULE FOR ALPHA-BETA PRUNING:

Every node in the minimax tree is assigned with alpha beta value as ($[\alpha, \beta]$ value)

Initially, Alpha, Beta $[\alpha, \beta]$ is assigned as $[-\infty, \infty]$

Throughout the working, the value of α and β should be $\alpha < \beta$ (pruning strategy)

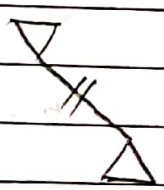
If $\alpha \geq \beta$ then pruning condition is executed

α is updated only at MAX △ nodes

β is updated only at MIN ▽ nodes

α, β of a parent node is passed on to the child node

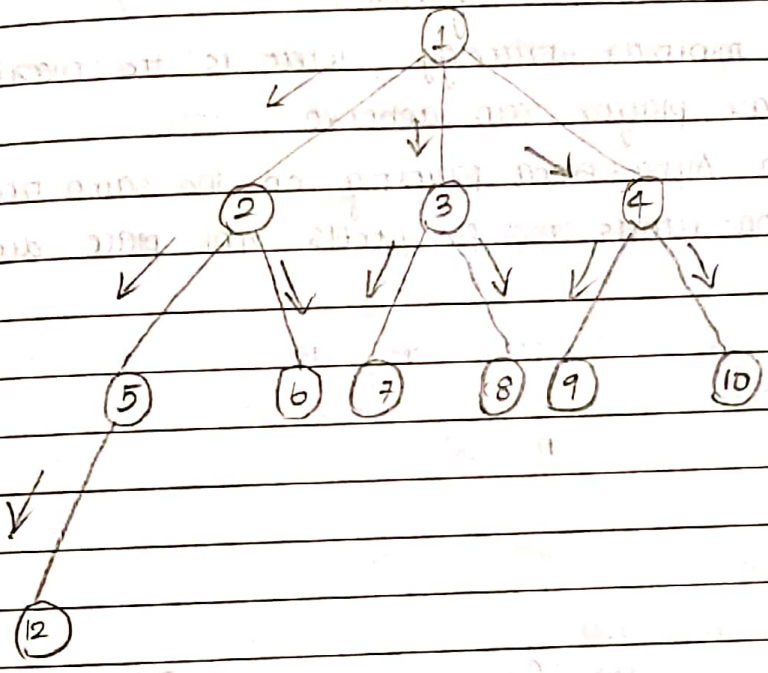
A search which is discontinued below any MIN node is called α-cutoff (Alpha-cutoff)



search discontinued below any max node is called

β - cut off (beta cutoff)

Alpha beta pruning just like Minimax strategy adopts depth first search.



1 2 5 12 6 3 7 8 4 9 10

2(c) *(15M)

10/04/2023

α-β pruning :- (problems)

↳ α, β

Rules :-

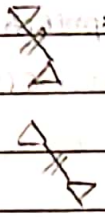
$[\alpha, \beta] = [-\infty, \infty]$

$\alpha < \beta$ throughout working

$\alpha \geq \beta$ pruning condition

$\alpha \Rightarrow \triangle$

$\beta \Rightarrow \triangledown$



⇒ Alpha - cutoff

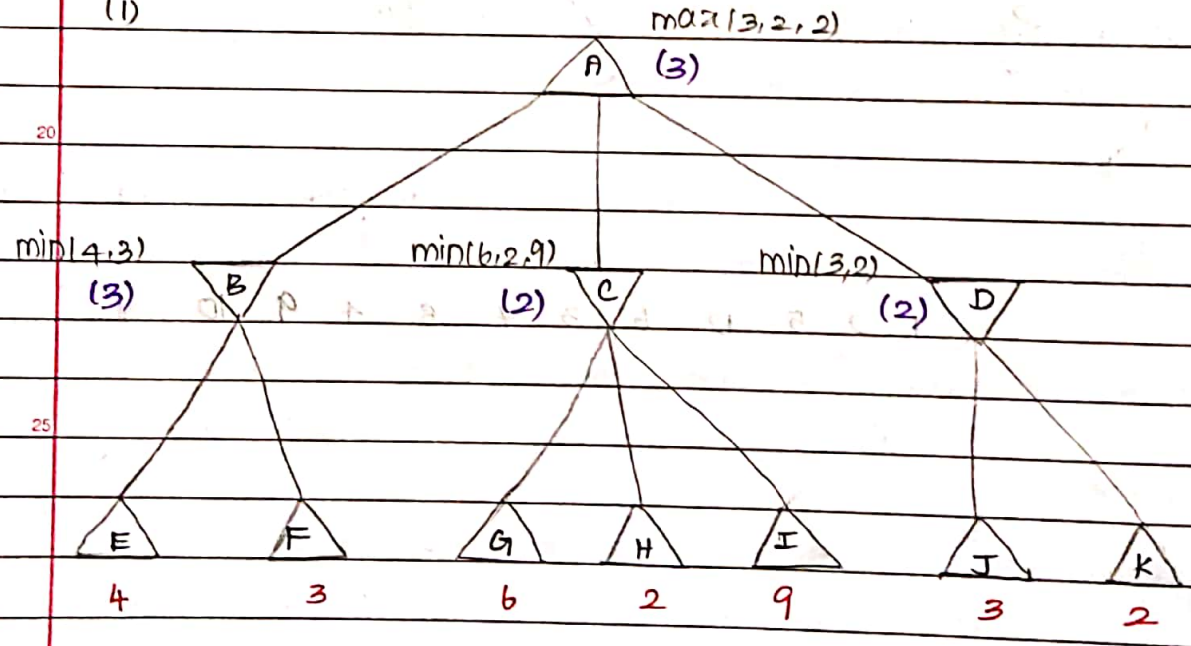
⇒ Beta - cutoff

A Minimax game tree is as follows

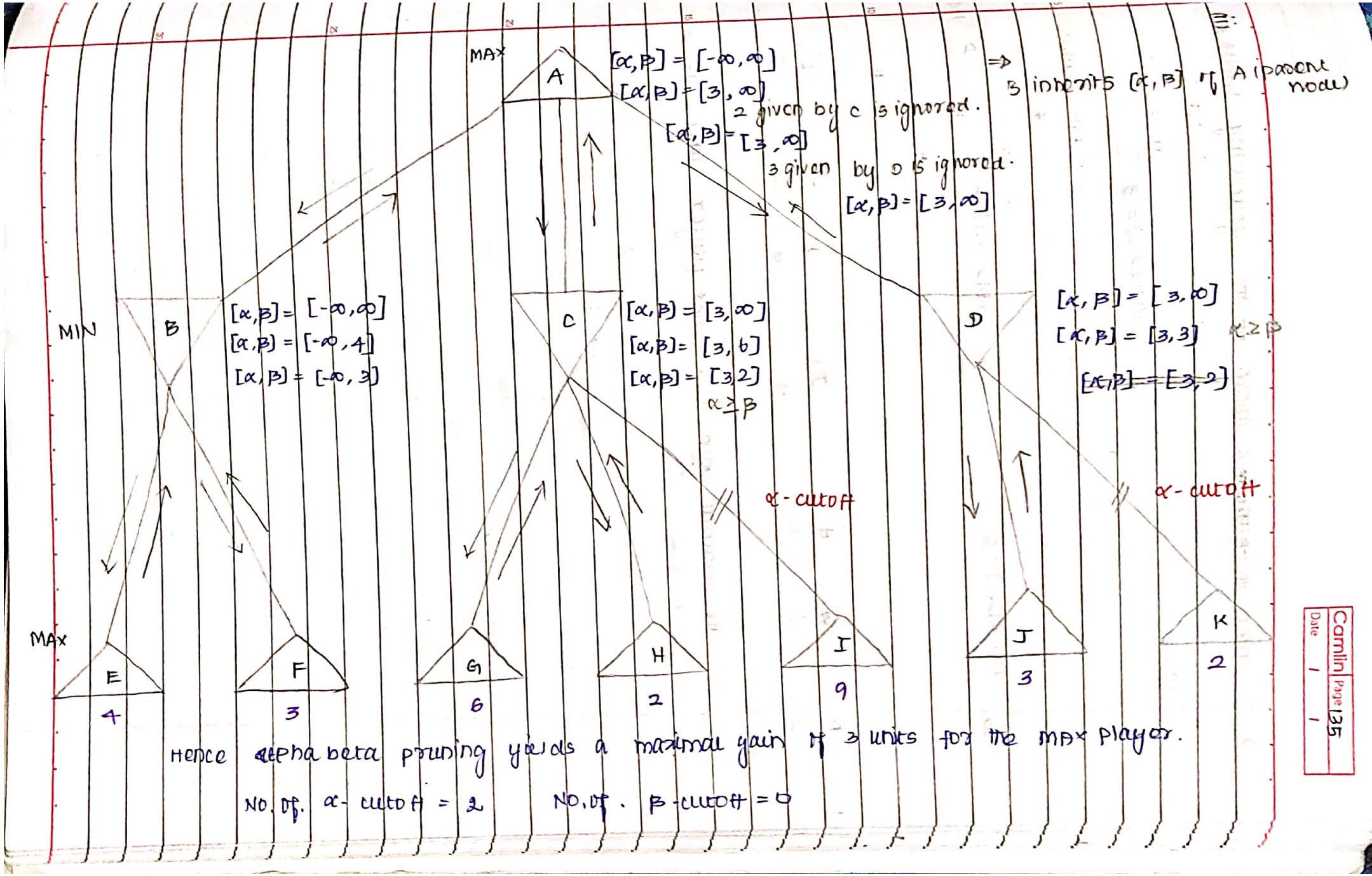
(i) Trace minimax strategy, what is the maximum gain that max player can achieve

(ii) Trace Alpha-Beta pruning on the same tree, how many alpha-cutoffs and β-cutoffs took place during the pruning.

(i)



The maximal gain that max player achieves is 3.



⇒ 3 inherits $[\alpha, \beta]$ of A (parent node)

2 given by C is ignored.

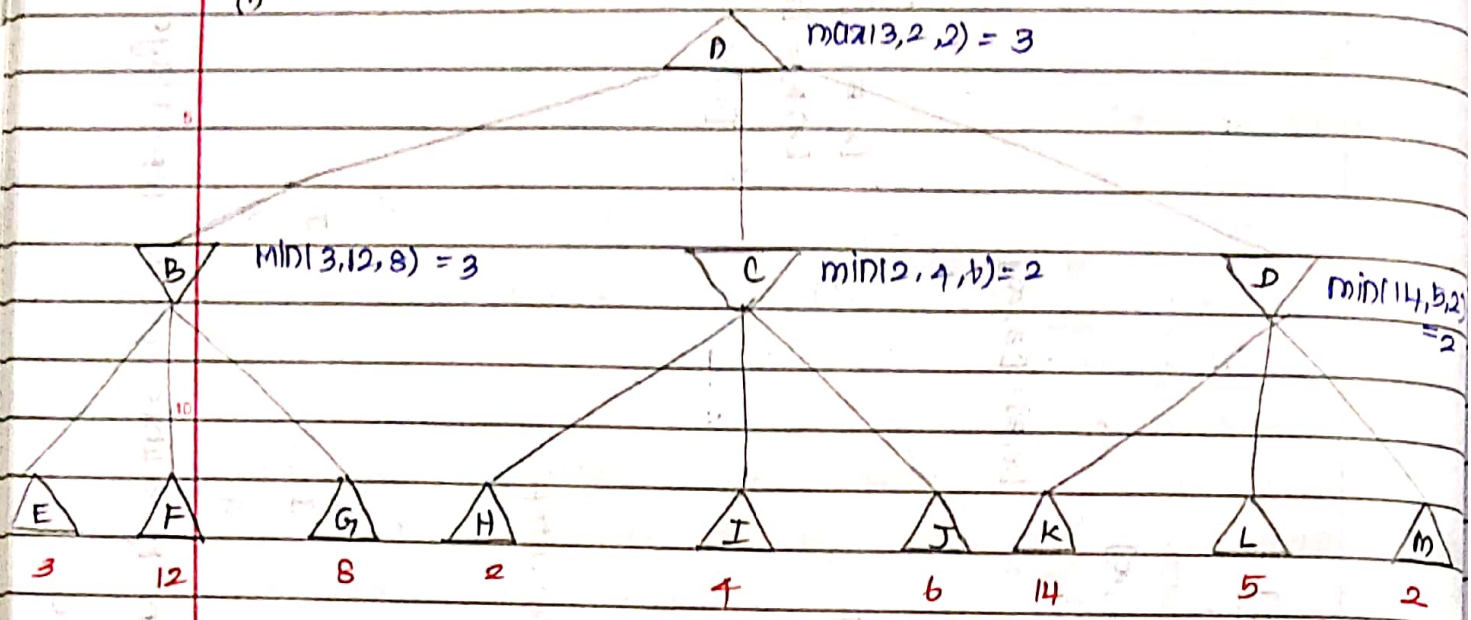
3 given by D is ignored.

Hence alpha beta pruning yields a maximal gain of 3 units for the MAX player.

No. of. α - cutoff = 2 No. of. β - cutoff = 0

2. trace the questions above on the following MINIMAX tree.

(i)



The maximal gain that MAX player achieves is 3.

(iii) $[\alpha, \beta] = [3, \infty)$

$\alpha \geq \beta$

A is given by C is ignored
$[\alpha, \beta] = [3, \infty]$
B is given by D is ignored
$[\alpha, \beta] = [3, \infty]$

MAX
MIN
MAX

A
B
C
D
E
F
G
H
I
J
K
L
M

$[\alpha, \beta] = [3, \infty]$
$[\alpha, \beta] = [3, 2]$
$\alpha \geq \beta$

$[\alpha, \beta] = [-\infty, \infty]$
$[\alpha, \beta] = [-\infty, 3]$

$[\alpha, \beta] = [3, \infty]$
$[\alpha, \beta] = [3, 4]$
$[\alpha, \beta] = [3, 5]$
$[\alpha, \beta] = [3, 2]$

$\alpha \geq \beta$

α-cutoff
β-cutoff

Hence alpha beta pruning yields a minimum gain of 3 units for the MAX player.

No. of α cutoff = 2

No. of β cutoff = 0

19/04/23

3. i) trace minimax strategy
ii) trace alpha beta pruning

The maximal gain that max player acheives is 5 units.

(i)

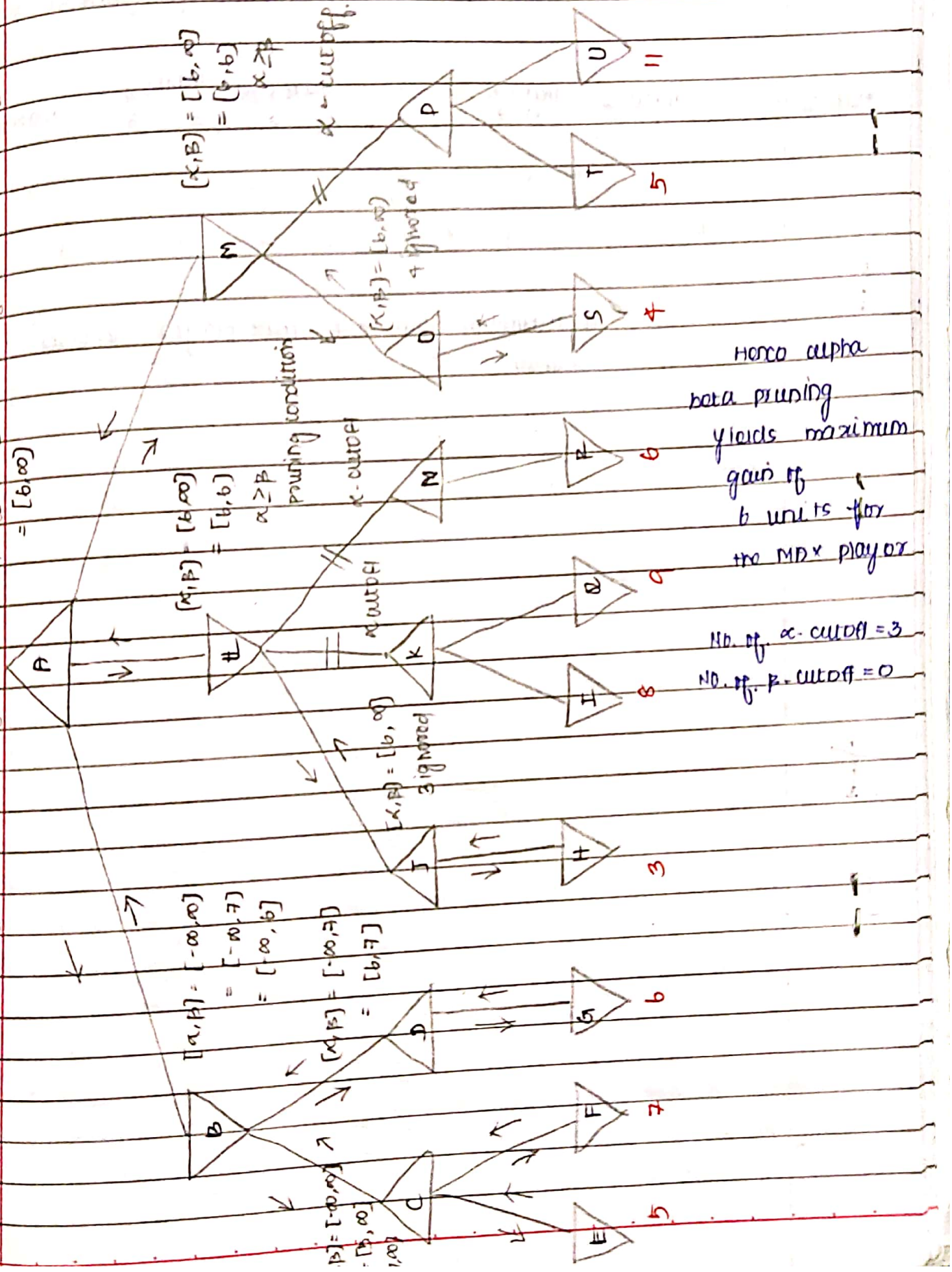
Camlin | Page 139
Date / /

```
function ALPHA-BETA-SEARCH(state) returns an action
    v ← MAX-VALUE(state, -∞, +∞)
    return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← -∞
    for each a in ACTIONS(state) do
        v ← MAX(v, MIN-VALUE(RESULT(s, a), α, β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v

function MIN-VALUE(state, α, β) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for each a in ACTIONS(state) do
        v ← MIN(v, MAX-VALUE(RESULT(s, a), α, β))
        if v ≤ α then return v
        β ← MIN(β, v)
    return v
```

The alpha-beta search algorithm. Notice that these routines are the same as the MINIMAX functions, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintains α and β (and the bookkeeping to pass these parameters along).

Camlin Page 141
Date / /

Trace alpha beta pruning on the following MINIMAX tree.



Hence alpha beta pruning yields maximum gain of 6 units for the MAX player

No. of α - cutoff = 3
No. of β - cutoff = 0

Constraint satisfaction problem (CSP):

Real world problem
(constraints)
problem is said to be solvable
only if conditions are
solved & technically called as
solution is constraints.
exist

Real world problem :
variables $x_1, x_2, \ldots, x_n$
values ~ domains
conditions ~ constraints.

Definition of CSP:
A constraint satisfaction problem (CSP) consists of three components X, D, and C. X is set of variables $[x_1, x_2, \ldots, x_n]$ D is a set of Domains $[D_1, D_2, \ldots, D_n]$ one for each variable. (i.e.), D_i for x_i. C is a set of constraints (conditions) that specify allowable combinations of values.

Each domain D_i consists of a set of allowable values $(v_1, v_2, \ldots, v_k)$ for variable x_i.

Each constraint C_i consists of a pair $\langle$ scope, rel $\rangle$ where scope is a tuple of variables that participate in the constraints and rel is the relation that defines the values that those variable can take on.

Advanced AI strategies
game playing
alpha beta
AND-OR graph
DFS → solution graph.
minimax (game theory)
→ minimax strategy
→ Alpha beta pruning

constraint satisfaction problem (CSP)

solution satisfies all conditions
↳ feasible solution.
optimal solution :
that feasible solution which satisfies the objective function
(Min() or Max() as the case may be)

Example :

If x_1 and x_2 both have domain {A, B} then the constraint saying that the two variables, x_1 and x_2 must have different value is written as

$$\langle (x_1, x_2), x_1 \neq x_2 \rangle$$

An alternate way of writing the constraint can be as follows

$$\langle (x_1, x_2), \{(A, B), (B, A)\} \rangle$$

19/04/23

Technical keywords of a CSP problem :-

To solve a CSP, we need to define a state space and the notion of the solution. Each state in a CSP is defined by an assignment of values to some or all of the variables.

i.e. $\{X_1 = v_1, X_2 = v_2, \ldots, X_n = v_n\}$

An assignment that doesn't violate any constraint is called consistent or legal assignment.

A complete assignment is one in which every variable is assigned and a solution to a CSP is a consistent, complete assignment.

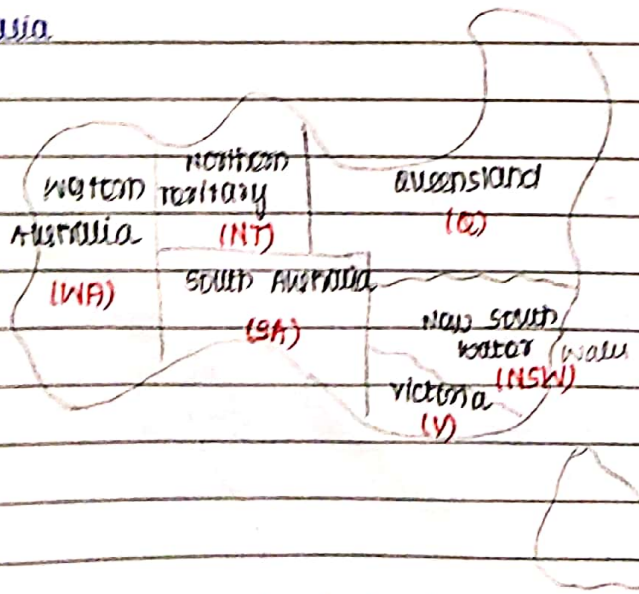
A partial assignment is one that assigns values to only some of the variables. → non deterministic polynomial

CASE STUDY :- MAP COLOURING :- (NP complete problem)

"exponential time complexity"

No. of colours - chromatic number.

Let us consider the principle states and the territories of Australia



The objective is to colour the map with three colours (Red, Blue, Green)

Human approach :

WA - Red NT - Blue SA - Green
Q - Red NSW - Blue V - Red T - Green.

AI approach :

AI regards this problem as a CSP, where the constraint is

(i) the no of colours given (Red, Blue, Green)
(ii) No two adjacent states or territories can have the same colour.

We define the CSP problem as follows

variables X (states & territory) = {WA, NT, SA, Q, NSW, V, T}

domain D (colour) = {Red (R), blue (B), Green (G)}

constraints C = {WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q, SA ≠ Q, SA ≠ NSW, SA ≠ V, NSW ≠ V, Q ≠ NSW}

solution is to be obtained by solving the CSP problem is a complete and consistent assignment of the values Red, Blue, Green to the 7 variables.

A possible solution is

{WA = R, NT = B, SA = G, Q = R, NSW = B, V = R, T = G}

21/04/2023

↱ NOP deterministic

Map colouring :: NP complete Problem.

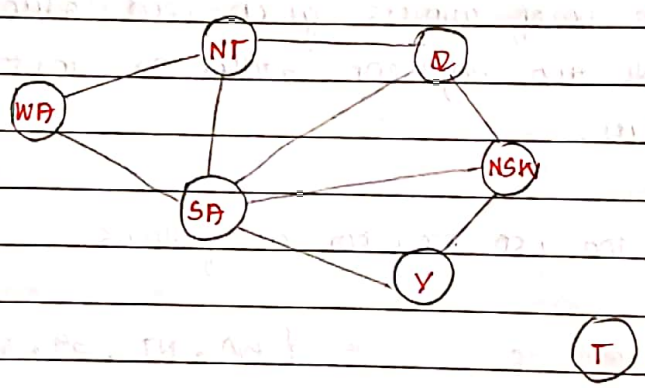
"exponential time complexity"

I/P
state space model ↷

Depth First search

In the next phase the map has to be transformed into a state space model which is a graph (undirected graph) such a graph is called a constraint graph.

The constraint graph has the variables (X) as the Nodes. And the adjacency as its Edges.



In the next stage we invoke Depth first search on the constraint graph and trace the order in which the nodes are visited. DFS - systemmatic search.

(i)

WA

OPEN ≠ EMPTY

N = WA goal (N) ? → X

WA

Expand N

(NT)
(WA)
(SA)

NT SA WA

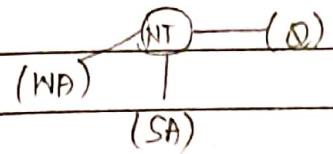
(iii)

OPEN ≠ EMPTY

N = NT goal(N) ? → ×

SA WA NT

Expand N



Q SA WA NT

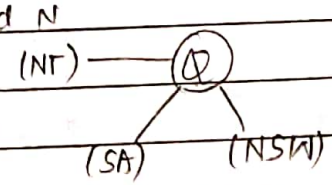
(iii)

OPEN ≠ EMPTY

N = Q goal(N) ? → ×

SA WA NT Q

Expand N



NSW SA WA NT Q

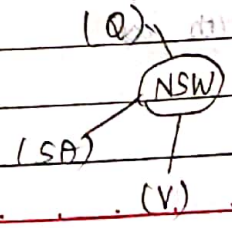
(iv)

OPEN ≠ EMPTY

N = NSW goal(N) ? → ×

SA WA NT Q NSW

Expand N



V SA WA NT Q NSW

(v) open ≠ Empty

N = V goal(N)? → False ✗

SA WA NT Q NSW V

Expand N

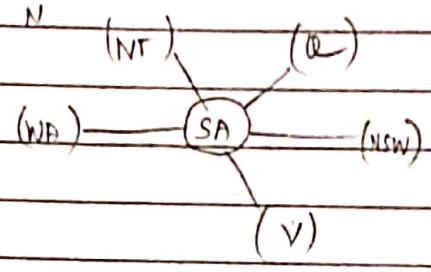
SA WA NT Q NSW V

(vi) open ≠ Empty

N = SA goal(N)? → False ✗

WA NT Q NSW V SA

Expand N



WA NT Q NSW V SA

(vii) since all the variables have not been included in the closed, we include T into Final output

=> WA NT Q NSW V SA T

Thus the Depth First Traversal of the given Coloring graph produces the sequence path as

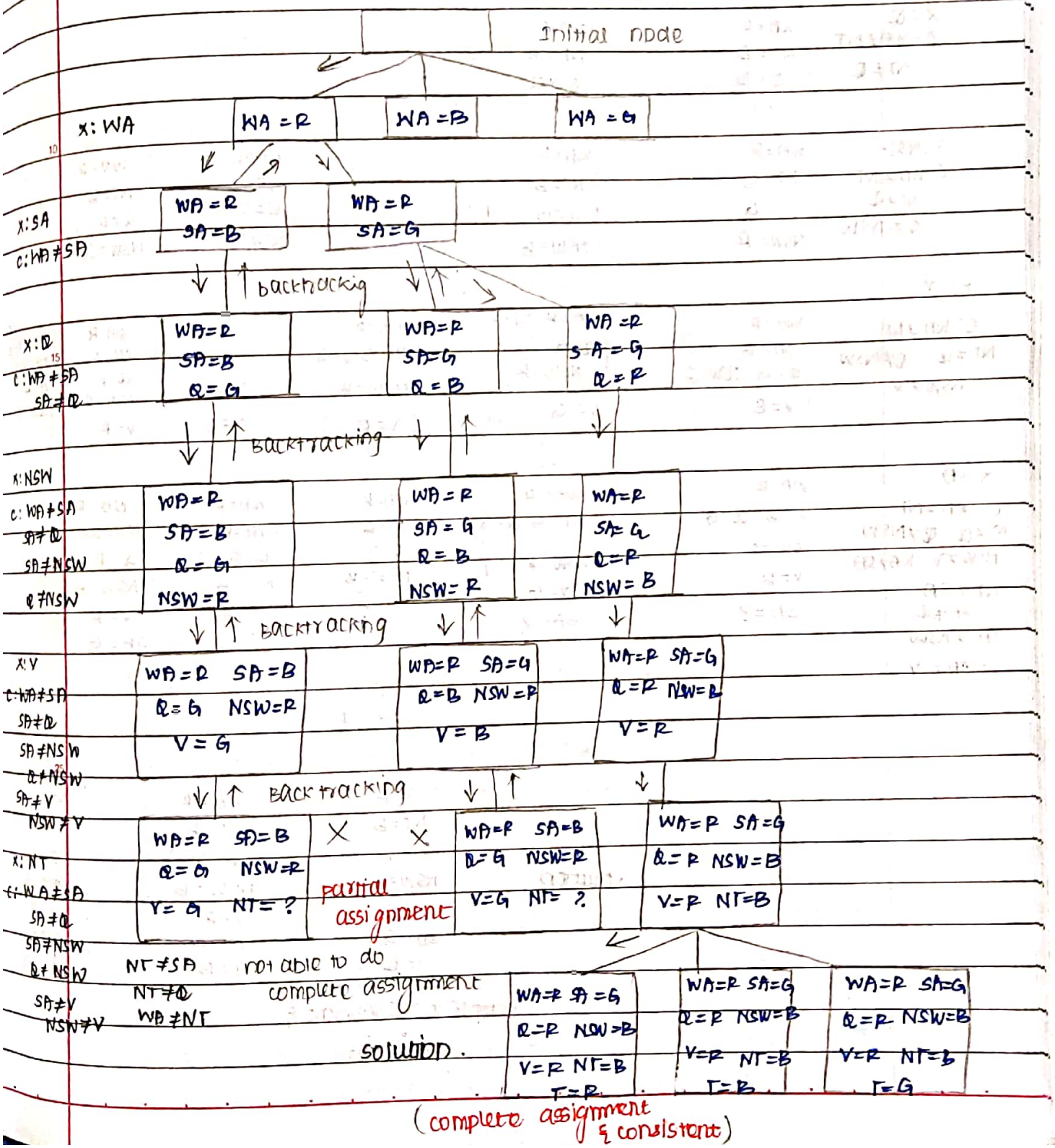
WA → NT → Q → NSW → V → SA → T

28/04/2023

Let us suppose, the DFS sequence was as follows

WA → SA → Q → NSW → V → NT → T

the DFS strategy proceeds to solve the Map colouring problem by combining itself with simple backtracking algorithm. the search tree is as follows



Path that have been got from pg:no:150

WA → NT → Q → NSW → V → SA → T