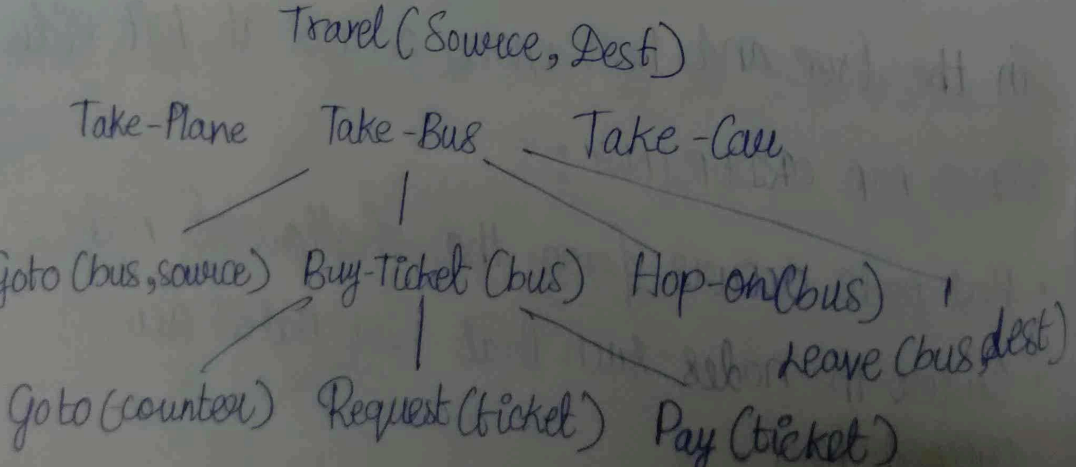
HIERARCHIAL PLANNING :

- It is a method based on Hierarchical Task Network or HTN planning.
- It combines ideas from Partial order & HTN planning.
- In HTL planning, the initial plan is viewed as a very high level description of what to be done.
- This plan is refined by applying decomposition actions.
- Each decomposition action reduces higher level action to lower set actions.
- Decomposition continues until only the primitive actions remain in the plan.

Eg: Travelling from a source to destination



- In the hie
from 'che
- Then use
by Plane
Bus;
- Then "Ta
Bus Stop"
& "Leave
- The "By t
decomposed
Pay for ti

⇒ Here we
Order Plann

Advantages :

- Computationa
- Create large
- Easy to fix

Disadvantages

- Require dete
- Unable to ha

- In the hierarchical diagram, suppose we travelling from 'chennai' to 'Goa'
- Then we have to plan how to travel, whether by Plane, Bus or Car. Suppose if we choose Bus;
- Then "Take-Bus" further into "go to Chennai-Bus Stop", "Buy ticket for Bus", "Hop on Bus" & "Leave for Goa".
- The "By ticket for Bus" can be further decomposed into: Go to Counter, Request tick & Pay for Ticket.

⇒ Here we use the concept of "One level Partial Order Planner"

Advantages:

- Computational cost is small.
- Create large plans
- Easy to fix problems & more efficient

Disadvantages:

- Require deterministic environment
- Unable to handle uncertain outcomes.

ALPHA-BETA PRUNING:

- It is a modified version of MINIMAX Algorithm
- It is an optimization for MINIMAX algorithm.
- The technique by which without checking each node of the game tree, computing the correct MINIMAX decision is called pruning.
- Using two-threshold parameter α & β is called α & β pruning.
- It can be applied at DFS.
- Two parameters can be defined as:

* Alpha: Highest value along the path of Maximizer. Initial value $= -\infty$

* Beta: Lowest value along the path of Minimizer. Initial value $= \infty$

- Hence pruning the nodes, makes the algorithm fast.

CONDITION FOR α-β PRUNING:

$$\alpha >= \beta$$

KEY POINTS:

- Max player will update only the value of α
- Min player will update only the value of β.
- While backtracking, the nodes will be passed to upper nodes instead of α-β values.
- We only pass α, β values to the child nodes.

MOVE ORDERING:

It is an important aspect of α-β pruning. It can be of two types;

- Worst ordering: It consumes more time because of α-β factors, such move is called worst ordering.
- Ideal ordering:

It occurs when lot of pruning happens in the tree and best move occurs at left side.

RULES FOR ORDERING:

- Best node occurs from the shallowest node.
- Order the nodes such that best nodes are checked first.

1] Agents:

Anything that can be perceived in its environment through SENSORS acting through ACTUATORS is called an agent.

Agent function:

Percept Sequence	Action
[A, clean]	Right
[A, Dirty]	Suck
[B, clean]	Left
[B, Dirty]	Suck
[A, clean] [A, clean]	Right
[A, clean] [A, Dirty]	Suck
⋮	⋮

Agent program:

function Reflex-VACUUM-AGENT ([location, status]) returns an action

if status = Dirty then return Suck
else if location = A then return Right
else location = B then return Left

Drawbacks:

- To big to generate and store
- No knowledge of non-perceptual parts
- Looping
- Takes a long time to build.

Types:

- Simple reflex agents:

→ It is a simple agent.
→ It selects actions based on current percepts
→ Large reduction in possible situations.

- Model-based reflex agents:

→ The most efficient way to handle partial observability is for the agent to keep track of outer world.
→ The agent should maintain an internal state that depends on percept history.
→ Updating the internal state information has two kinds
* information given by us.
* Information learned from its actions.

- Goal-based agents:

⇒ The agent needs some sort of goal information that describes desirable decisions.

Eg: Passengers destination

⇒ The agent can combine its information with its actions in order to achieve the goal.

- Utility-based agents:

⇒ It uses a model of the world, along with utility functions that measures preferences.

⇒ Then it chooses the action for expected best utility.

⇒ Certain goals can be reached in different ways, some are better, have a high utility.

CONCEPT OF RATIONALITY:

- A rational agent is one that does the right thing - conceptual speaking.
- Every entry of the agent function is filled out correctly.
- Right action cause the agent to be most successful.

Rationality : (depends on)

- Performance measures defines the criterion of success.
- Agent's prior knowledge
- Actions that the agent can perform
- Agent's percept sequence to date.

These gives the definition of a rational agent.

⇒ An important part of rationality is information gathering.

⇒ But collecting information only doesn't matter, it should include learning what it perceives

⇒ An rational agent should be autonomous.

Task environments :

They are the problems to which the rational agents are the solutions.

⇒ Rationality needs the specification of ;

- performance measure
- environment
- Actuators
- Sensors

7] Kriegspiel :

= A partially observable variant of chess in which pieces can move but are completely invisible to opponent.

Belief State :

= Keeping track of it is exactly the problem of state estimation.

Strategy :

= We need a move for every possible percept sequence that might be received.

Winning Strategy :

= For each possible percept sequence, leads to an actual checkmate. AND-OR search algorithm can be applied to find checkmates.

Probabilistic Checkmate :

= Checkmates are probabilistic with respect to randomization of winning player's move.

UNIT - 3

6] CSP as a search Problem :-

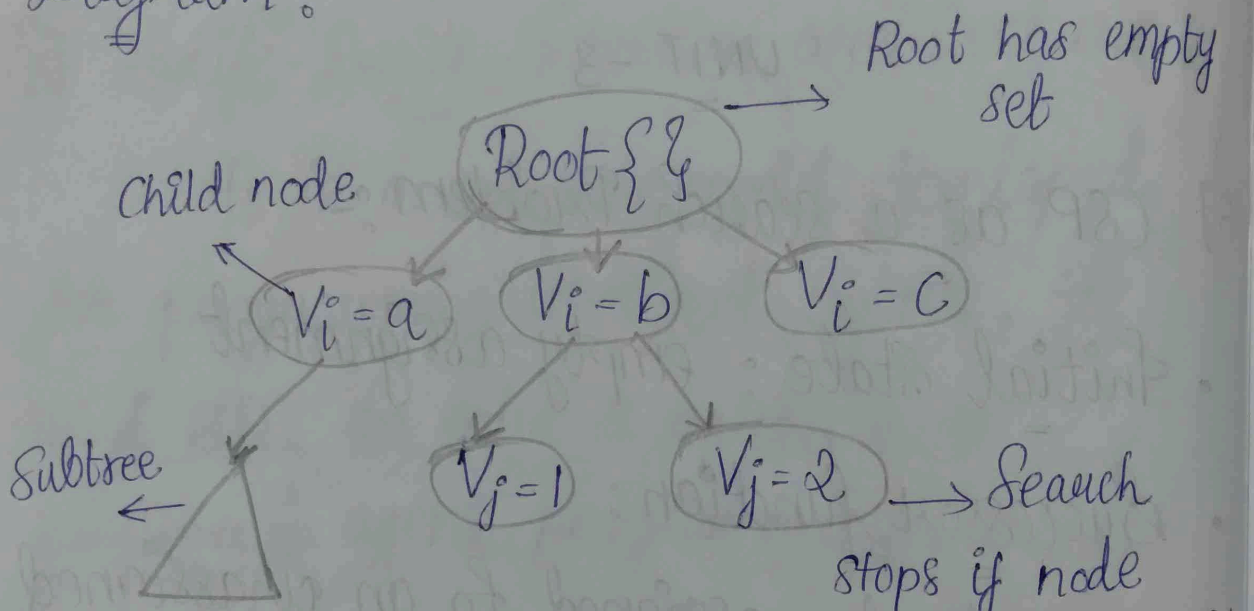
- Initial State : empty assignment
- Sucressor function :

  A value assigned to an unassigned variable, which doesn't conflict with the currently assigned variables
- Goal test : Complete assignment
- Path cost : Irrelevant

Backtracking Search :-

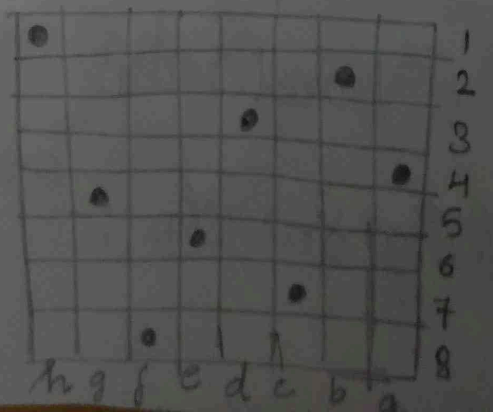
- DFS with single-variable assignment
- Basic uninformed search.
- Significantly reduces search spaces.
- Time complexity: From $O(d^{n!})$ to $O(d^n)$

Diagram:



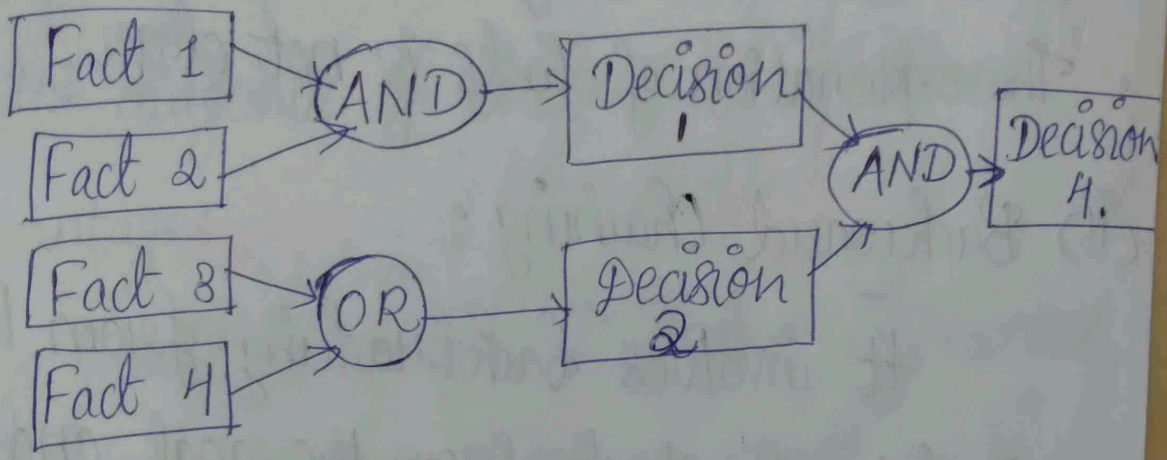
Eg: N-Queens

Placed N Queens on N×N chess board so that no queen can attack any other Queen.



5] (i) Forward Chaining:

In this methods rules of inference are applied to an existing data to extract additional data until goal is achieved.



Properties:

- It uses down-up approach
- It is data-driven

Eg: A - Tom is running

A→B - If tom is running, he will sweat

B - ∴ Tom is sweating

A DENDRAL expert system is a good example for FC. It is used in the prediction of molecular structure of substances.

Advantages:

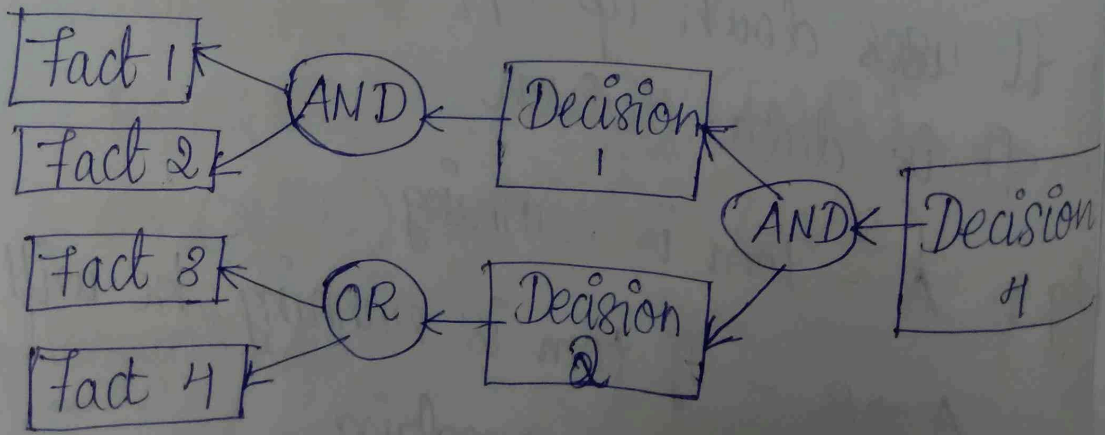
- It can be used to draw multiple results.
- It provide a good basis

Disadvantages:

- It is time consuming
- The explanation of facts is not clear.

(ii) Backward Chaining:

It involves backtracking from the endpoint. It starts from the goal and moves backward to initial point.



Properties:

- It uses up-down approach
- It is goal-driven

Eg: B - Tom is sweating

A → B - If tom is running, he will sweat

A - Tom is running

The MYCIN expert system is a real life example of backward chaining. It is used in the diagonis of bacterial infections.

Advantages:

- As result is known, it deduce inferences.
- It's a quicker method as result is available.

Disadvantages:

- It can be started if endpoint is known.
- It doesn't deduce multiple solutions.

UNIT - IV

1] Reasoning :

It plays a great role in the process of AI. Thus reasoning can be defined as the logical process of drawing conclusions, making predictions towards a particular thought with the help of existing knowledge.

Methods :

* Deductive Reasoning:

It is a strategic approach that uses available to draw valid conclusions. It believes in the data before drawing any result. It uses top-down approach.

Eg: In a class, ratio of boys more than girls.

* Inductive Reasoning:

It is associated with the hypothesis-generating approach. It helps in making generalization from specific facts. It uses bottom-up process.

Eg: All students from a class in London

* Common sense reasoning:

It comes from experience. It uses its previous experiences to draw a conclusion to do situation.

Eg: While overtaking someone on the road.

* Monotonic Reasoning:

It follows the different approach towards the thinking process. it uses facts to draw conclusion. It is used in conventional reasoning and logical-based systems.

Eg: Longest river in the world is Nile.

* Abductive Reasoning:

It uses incomplete facts and draws the condusion with the incomplete fact. It play an important role in daily life decision-making process.

Eg: Doctor explaining health reports

2] Propositional Logic:

It is the simplest form of logic where all the statements are made by propositions. It is a declarative statement which is either true or false.

Eg: It is Sunday.

Facts:

- Propositional logic is called Boolean logic as it works on 0 & 1
- It can be either true or false, but it cannot be both.
- It consists of an object, relations or function and logical connectives.

Syntax:

* Atomic Propositions:

It is simple & consists of a single proposition symbol. Gives either T or F

Eg: 2+2=4, it is true fact

* Compound Proposition:

It is constructed by combining atomic propositions, using parenthesis & logical connectives.

Eg: It is raining and the street is wet.

Logical Connectives:

Symbol	Word	Term	Example
$\wedge$	AND	Conjunction	$A \wedge B$
$\vee$	OR	Disjunction	$A \vee B$
$\rightarrow$	Implies	Implication	$A \rightarrow B$
$\Leftrightarrow$	If and only if	Biconditional	$A \Leftrightarrow B$
$\neg$ or $\sim$	Not	Negation	$\neg A$

Truth Table:

The representation of all combinations with logical connectives in a tabular format is called Truth table.

Eg

A	$\neg A$
T	F
F	T

Precedence:

I	II	III	IV	V	VI
()	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\Leftrightarrow$

Properties:

- Commutativity:

$$P \wedge Q = Q \wedge P$$
$$P \vee Q = Q \vee P$$

- Associativity:

$$P \vee (Q \vee R) = (P \vee Q) \vee R$$
$$P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$$

- Identity element:

$$P \wedge T = P$$
$$P \vee T = T$$

- Distributive:

$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$$
$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$$

- De Morgan's Law:

$$\neg(P \vee Q) = \neg P \wedge \neg Q$$
$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

- Double-negation:

$$\neg(\neg P) = P$$

PART-B UNIT-III

2] Game theory in AI :

It is basically a branch of mathematics used to model the strategic interaction between different players in a context with predefined rules and outcomes.

It can be applied in AI :

- Multi-agent AI-systems
- Imitation and Reinforcement Learning
- GAN's training

It can be divided into 5 types :

* Cooperative vs Non cooperative Games :

In cooperative games, participants can form alliances, but in NC games, they cannot form alliances (Eg: war)

* Symmetric vs Assymmetric Games :

In symmetric games, all participants have same goals & strategies but in

Asymmetric games, participants have different goals.

* Perfect vs Imperfect Information Games:

- In PI games, all players can see other players moves but in II games players can't see other players moves.

* Simultaneous vs Sequential games:

In simultaneous games, different players can take action concurrently, but in sequential games, each player is aware of other player's previous action

* Zero-Sum vs Non-Zero Sum Games:

In ZS games, player get profit from other players loss but in NS games, players get profit from other players profit.

Nash Equilibrium:

It is considered the essence of GT. It states that a point of equilibrium of collaboration of multiple players in a games. It guarantees maximum profit to each player.

Generative Adversarial Networks (GANs):

It is the combination of Discriminator and Generator. It is a 2-player competitive game where both players are continuously optimizing themselves to find Nash Equilibrium.

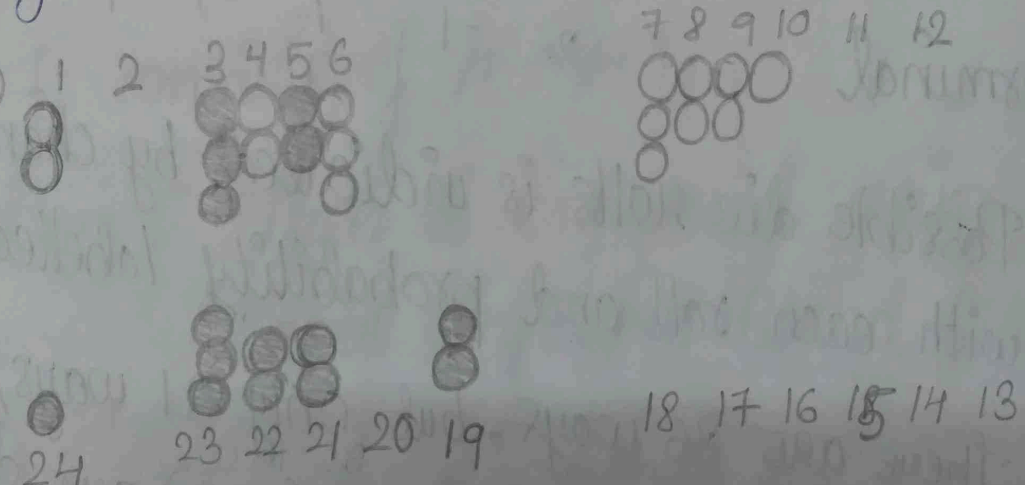
How to know game has reached Nash Equ?

In any game, one of the agents is required to disclose their strategy in front of other agents, if it doesn't happens, the games has reached Nash Eq.

3] Schematic Diagram (Game tree) for a backgammon position:

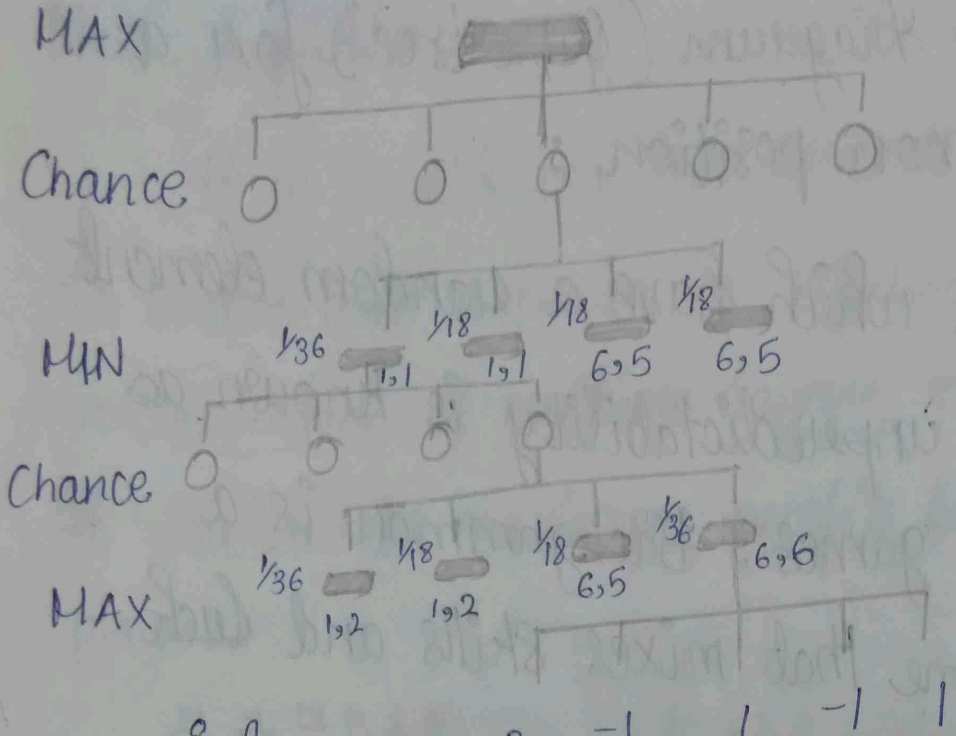
A game which have a random element to reflect unpredictability is known as stochastic games. Backgammon is a classic game that mixes skills and luck.

Eg:



Stochastic game tree for gammon position:

- He/she legal moves with white.
- He/she has no idea how black rolls/moves.
- So white can't build a normal game tree
- In backgammon, in addition to MAX & MIN nodes, a game tree must include chance nodes.



- Possible die rolls is indicated by chance nodes, with each roll and probability labelled.
- There are 36 ways, but only 21 ways/rolls are there because (6-5) & (5-6) are same. (1-1) to (6-6) has $P = \frac{1}{36}$. So other 15 rolls have $\frac{1}{18}$ of chance of happening.

As a result, we can generalise the deterministic minmax value to an expected minimax value with chance nodes. We compute expected value which is the sum of all outcomes, weighted by P of each action.

5] Monte Carlo Tree Search (MCTS):

It is a seach technique in AI. It is a probabilistic & heuristic driven search algorithm, which combines tree search implementation.

MCTS algorithm:

It consists of 4 steps:

- Selection: Tree is traversed using Upper Confidence Bound (UCB).

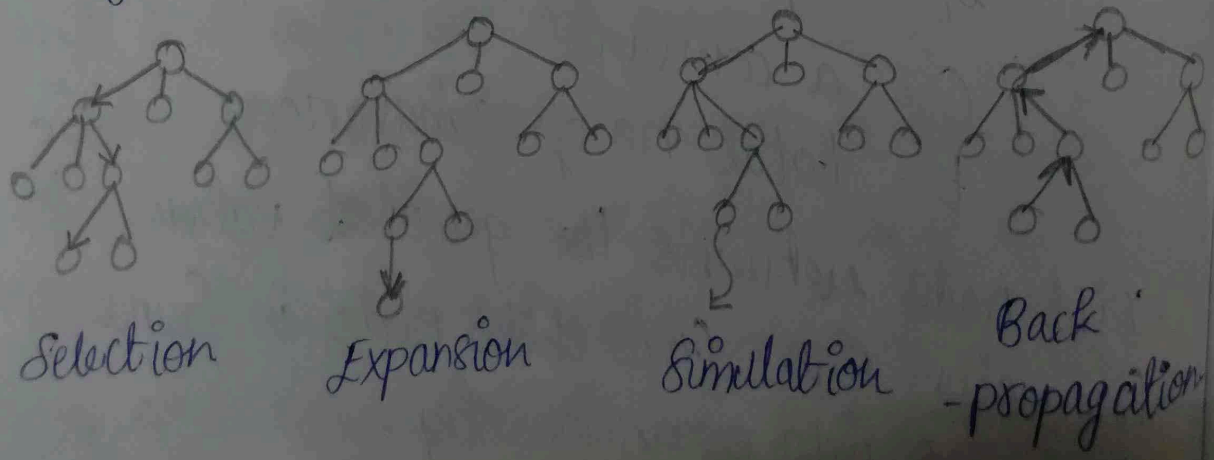
Formula: $S_i = x_i + c\sqrt{\frac{\ln(t)}{n_i}}$

where S_i = value of a node i
x_i = empirical mean of node i
C = a constant
t = total no of simulations

Child node returns the greatest value from above eqn. Once child node is found it jumps into expansion step.

- Expansion : A new child is added to tree which was optimally reached during the selection process.
- Simulation : It is performed by choosing moves or strategies until a result is achieved.
- Back Propagation : After adding a new node, tree must be updated. It back-propagates from new node to root node. Here no. of simulations is incremented. Also if new node's simulation results a win, then no of wins is also incremented.

Diagram :



Advantages :

* Simple algorithm to implement
* It is heuristic algorithm
* It can be saved in any state
* It supports asymmetric expansion.

Disadvantages :

* It requires large amount of memory
* There is a bit of speed issue.
* Reliability issue.

PART - B

19] State :

It contains all of the information necessary to predict the effects of an action and to determine if it is a goal state.

State Space :

It is a set of all possible states of a system. It considers a state of an instance with the intention of finding a goal state with desired property.

Search tree :

It is a tree data structure used for locating specific keys from within a set. It is the process of visiting each node in a tree data structure

Search Node:

= The root node is the start state and the set of children for each node consists of states reachable by taking any action while nodes in the tree.

Goal Action:

= It is an AI system for agents that allows them to plan a sequence of actions to satisfy a particular goal. It haves an individual willingness to achieve a goal set.

Transition Model:

= It describes what each action does.

Goal Test: It determines given state a goal state

Path Cost: It assigns a cost to each path that follows goal.

Branching Factor:

= Branches are influenced by many factors If the out degree is not uniform, average branching factor can be calculated.