(35) Unit 01

Unit I
Introduction.

MBD WRITEWELL
Date .....
Page .....

- Nowdays, we are trying to make system (machines) to be intelligent.
- Intelligence is the ability to think and understand instead of doing things by instinct or automatically.
- Intelligence involves solving problems, learning, building analogies, creativity, reasoning, planning, perceiving & then acting.
- Basically, Artificial intelligence is defined by John McCarthy in 1956 as "the science and engineering of making intelligent systems (machines").
- AI is the study of - How to make computers do things which at the moment people can do better.
- A system or a machine is said to be behave intelligently if it passes the following characteristics.

1. Perceive (understand) one's own environment.
2. Act in complex environment.
3. Learn and understand from experience.
4. Reason to solve problems and discover hidden knowledge.
5. Able to apply knowledge successfully in new situations.
6. Think abstractly using analogies
7. Communicate with others.

Imp ★ Goals / Approaches of AI :-

- Definitions of AI leads to 4 possible approaches for AI :-

⟹

MBD WRITEWELL
Date .....
Page .....

1. Cognitive Science Approach — Systems that think like human.

2. Law of ~~Four~~ Thought Approach - Systems that think rationally.

3. Turing Test Approach - Systems that act like human.

4. Rational Agent Approach - Systems that act rationally.

	Human-like	Rationally
Think	1) Cognitive Science Approach.	2) Law of Thought Approa
Act	3) Turing Test Approach	4) Rational Agent Approach.

1) Cognitive Science : Think like human.
- Cognitive science aims to develop, explore and evaluate theories of how mind works through the use of computational models.

MBD WRITEWELL
Date .....
Page .....

- The goal is to make machines think like human.

- Goal is not just to produce human-like behaviour but to see if machine follows the reasoning process i.e. to verify that machine follows the steps which are similar to the steps followed by the human while solving a problem.

2. Law of Thought: Think Rationally
It is the study of mental faculties through use of computational models.

- It is basically the study of computations that make it possible to perceive, reason, act.

- Focus is on inference mechanism which are provably correct and provide optimal solution.

- Inference is the process of deriving a conclusion on what one already knows.

- For eg:- "Socrates is a man. All men are mortal. Therefore Socrates is mortal".

- The goal is to formalize the reasoning process as a set of rules and procedures for inference.

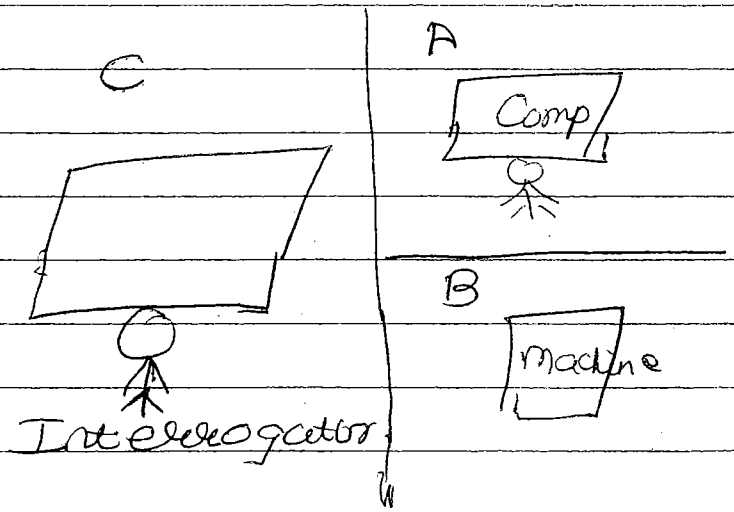
- But the issue is not all problems can be solved by just reasoning and inferences.

3) Turing Test:- Act Human Like
- It is the study of how to make computers do things which at the moment people do better.

- Focus here is on action, and not on intelligent behaviour.

- Goal is to develop systems that are like human.

- In order to conclude if a machine is like human Following test is called as Turing Test is performed:
- Basically this test involves 3 rooms which consists of a person, a computer and an interrogator



- The interrogator can communicate with other 2 by teletype.
- The interrogator tries to determine which is the person and which is the machine.

- The m/c tries to fool the interrogator to believe that it is human, and the person also tries to convince the interrogator that it is the human.

- If the machine succeeds in fooling the interrogator, then conclude that machine is intelligent.

4) Rational Agent : Act Rationally
- Rational means to do the things rightly.

- Focus here is on systems that act sufficiently if not optimally in all situations.

- Goal is to develop systems that are rational and sufficient.

MBD WRITEWELL
Date .....
Page .....

* History of AI:-

1. 1923 —
First use of the word "Robot" in English from Karel Kapek's play named "Rossem's Universal Robots" (RUR).

2. 1943 —
Foundations for neural networks laid.

3. 1950 —
Alan Turing introduced Turing Test.

4. 1956 —
- John McCarthy introduced term Artificial Intelligence.
- Demonstration of first running AI program at Carnegie Mellon University.

5. 1958 —
John McCarthy invents LISP programming lang. for AI.

MBD WRITEWELL
Date .....
Page .....

6. 1964 -
Danny Bobrow's dissertation at MIT showed that computers can understand natural Eng to solve algebra word problems.

7. 1965 -
Joseph W. at MIT built ELIZA an interactive problem that carries on dialogue (or chats) in English.

8. 1968 -
Scientists at Stanford Research developed Shakey, a robot, equipped with locomotion, perception and problem solving.

9. 1973 -
Robbot called as Freddy was built by at Edinburgh Univers where this robot was capable of using vision to locate and assemble models.

MBD WRITEWELL
Date .....
Page .....

10. 1979 –
The first computer controlled autonomouus vehicle, Stanford Cart, was built.

11. 1990 –
Major advances in all areas of AI –
1. Games.
2. Vision, Virtual reality.
3. Scheduling.
4. Case-based reasoning.
5. Multi-agent planning.

12. 1997 –
The Deep Blue Chess program beats world chess champion, Gaarey Kasparov.

13. 2000 –
MIT displays, Kismet a robot with a face that express emotions.
Robot Nomad explores remote regions of Antarctica.

MBD WRITEWELL
Date .....
Page .....

Foundations of AI :-

1. Philosophy –
Logic, reasoning, learning.

2. Mathematics –
Formal representat^n and proof, algos, computation.

3. Probability / Statistics –
Modeling uncertainty, learning from data.

4. Economics –
Utility, decision theory.

5. Neuroscience –
Neurons as info. processing units.

6. Psychology / Cognitive Science –
How people behave, perceive, represent knowledge.

7. Computer Engg –
Building fast computers.

8. Control theory –
Design systems that maximize an objective fun over time.

MBD WRITEWELL
Date .....
Page .....

9. Linguistics -
Knowledge representation,
grammars.

* AI models:-
- One important aspect of building AI solutions is modelling the problem.
- Foll. models are used in AI:-
1. Semiotic Models:-
- These models are based on sign processes or signification and communication.
- The processes of carrying meaning depends on codes.
- Semioticans classify signs or signs system in relation to the problem.
- This meaning assignment and mapping process depends on the use of codes based on individual sounds or letters that human use to form words or movements

2. Statistical Models:-

- It refer to representation and formalisation of relationships through statistical technique.
- Most of the AI problems can be represented as statistical or pattern matching problems.
- Various learning models in AI are based on statistics.

* Intelligent Agent:-

Def^n – An agent is an entity that can perceive (understand) the information and act on that information to achieve the desired outcome.

- Intelligent agent is an agent that is capable of making decisions and act most logically in the scenario.
- The foll. fig. depicts the general structure and working of an agent.

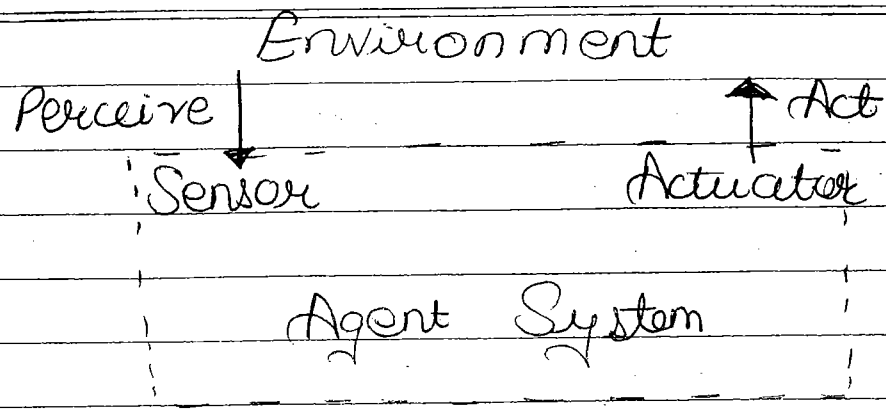


Fig:- Basic Structure of an Agent.

- An agent interacts with the environment through sensors and actuators.

- A sensor allows the agent to sense environment and perceive the present state.

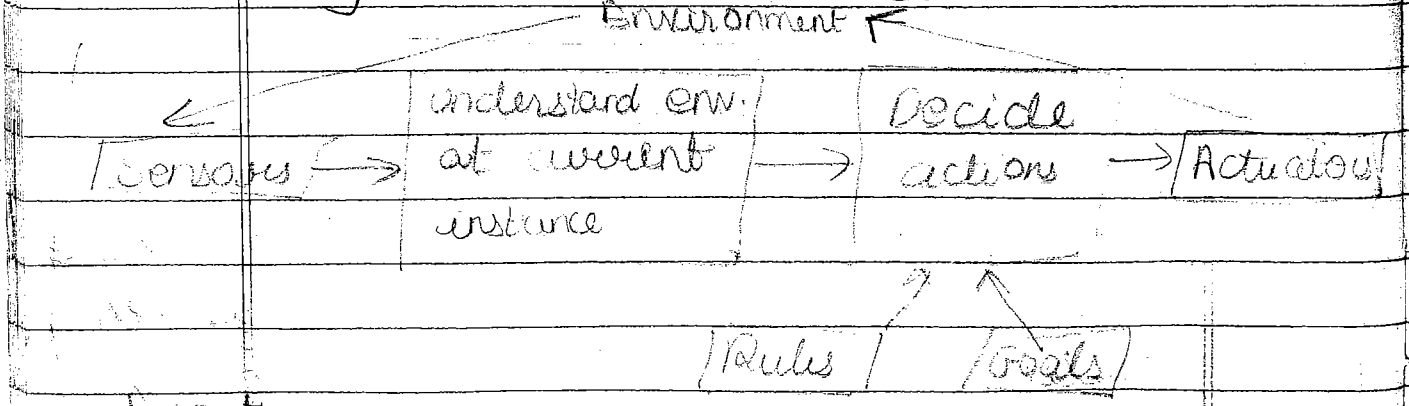
- An actuator allows the agent to take actions with reference to the environment perceived.

- Considering human being as an agent his/her ears, eyes, nose, tongue acts as a sensor while her/his hands, legs acts as an actuator.

MBD WRITEWELL
Date .....
Page .....

Defⁿ. Intelligent agent is an entity that works without assistance, interprets inputs, senses the environment, makes choices and ultimately acts to achieve goal.

- IA have to learn and use knowledge to achieve their goals.
- Agents may be very simple or very complex.
- Eqs:- A reflex m/c such as thermostat, a human being, etc
- Fig. shows relationship between agent and environment.



MBD WRITEWELL
Date .....
Page .....

- It is a set of rules, (if-then) that drives the decisions of the agent towards the goal fulfillment

Eg.- 1) Automatic Attendance S/m -
- Here, the fingerprint sensors sense the fingerprints.
- The actuator gives the message and attendance is marked for employee over to sensed fingerprint

2) Auto-door open and close system
Camera - Sensors
Actuator - Motors that close/open door

a) Percept -
- Percept is the mental result or product of perceiving.
- Basically its an impression or sensation of something perceived.

- An agent while observing has some percept at any moment of time.
- A percept is a sequence i the complete history of everything the agent has ever captured.
- Agents choice of action depends on the percept sequence.

b) Agent Function –

- Agent behaviour is mathematically represented by agent function.
- This agent function is generally implemented by an internal program called agent program.
- Agent pgm uses an agent function to achieve the desired goal.
- The agent function maps from percept histories to actions.

$f : P \rightarrow A$.

* Rational Agent :-

- A rational agent is an agent that behave logically and does right things.
- A rational agent is an agent that chooses to perform an action which leads to an expected optimal result.

- A rational agent provides or makes rational rather logical decisions.

- Typical examples of rational agent can be a person, governing body, decision authority, firm, m/c or S/W.

- Rationality determines how an intelligent agent needs to behave/act with respect to the environment to get the expected outcomes.

Ideal rational agent ⇒
- A rational agent is expected to select an action that could maximize the performance, having the evidence on the basis of percept sequence as well as built-in knowledge.

- So, agent takes actions and decisions on the available percept sequence.
- The parameters that play a role in it are as follows:-

1. Priorities and preferences of the agent.
2. Info available with the agent about its environment.
3. Possible actions an agent can perform.
4. Estimated on the actual benefits and the chances of the success of the actions.

* Task Environment and its Types.- Imp.

- Intelligence is always defined in association with environment.
- The task environment is the environment in which the task takes place.
- The task environment along with desired behaviour and task is necessary for

classmate
Date
Page

Unit II (contd..).
Problem Solving and Building Smart Systems.

* General Search:-
- Search is an algorithm that discovers or locates a path to the solution.
- Search is an algo. that takes problem as an input and returns with a solution from search space.
- Search space is a set of all possible solutions.
- Solution is a state where all requirements are fulfilled. Such a state is called as Goal state.
- A state space [S, A, I, G] is a collection of states, arcs between them and a non-empty set of intial states and goal states.

* Control Strategies:-
- They guide how to reach the goal state or what way has to be followed in order to find a solution.
- Foll. are the search control strategies:-
1. Forward Search
- The search proceeds from intial to goal state.
- Data directed methods.
- Eg:- Searching a city on map.

2. Backward Search:-
- Search proceeds from goal state to initial state.
- Goal Directed

3. Systematic Seach -
- Here no info abt domain is available.
- It can only distinguish betn goal and non-goal state.
- Used when search space is small.
- Eg:- BFS, DFS.

4. Heuristic Search:-
- Search depends on knowledge of problem domain.
- Basically they guide the search and hence called as heuristics.

* Parameters of Search Evaluation:-
1. Completness - Algo is said to be complete if it is guranteed to find a solution.
2. Optimality/Admissibility - A search solution is said to be optimal if it gives best soln.
3. Time Complexity - Max time reqd to execute the algo.
4. Space Complexity :- Max memory space reqd to compute the algo.

Time and space complexity are measured in terms of:-
b = maximum branching factor of tree.
d = depth
m = max length of any path

* Types of Search Method:-
1. Uninformed Search-
- Uses no info abt problem to guide the search and ∴ efficient.
- Also called as blind, exhaustive or brute force search.

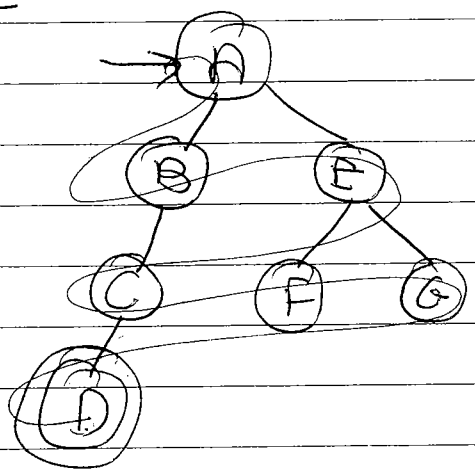
2. Informed Search-
- Uses info abt the problem to guide the search, usually guesses the distance to goal state ∴ efficient.
- But search may not be always possible.
- Also called as heuristic / intelligent search.

* Uniformed Search:-
- No knowledge about problem domain.
- Foll. are uninformed search methods
1. Breadth First Search.
2. Uniform Cost Search
3. Depth First Search
4. Depth Limited Search.
5. Iterative Deepening Depth First Search.
6. Bi-directional Search.

classmate
Date
Page

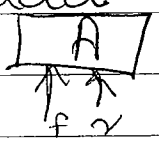
1] Breadth First Search:-
- A search strategy in which highest layer of decision tree is searched completely before proceeding next layer is called BFS.
- Guarantees optimal (best) solution.
- Uses FIFO (Queue) data structure for implementation.

• Example:-

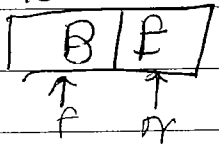


Soln:-

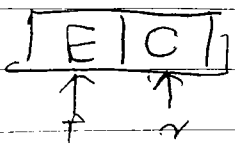
1] Initially add root node A to empty queue



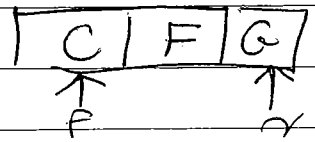
2] Remove A and add its successors.



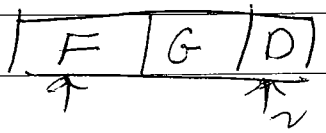
3] Remove B, add its succ. C.



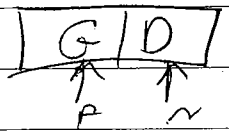
d) Remove E, add its succ?



e) Remove C. Is it a goal? No. then add successor of C to queue



f. Remove F.



g. Remove G.

h. Remove D which is a goal node and Stop.

• Properties of BFS -

1. Completness — Yes.
Solution is reached if it exists.

2. Optimality — Yes.
Finds the shortest path

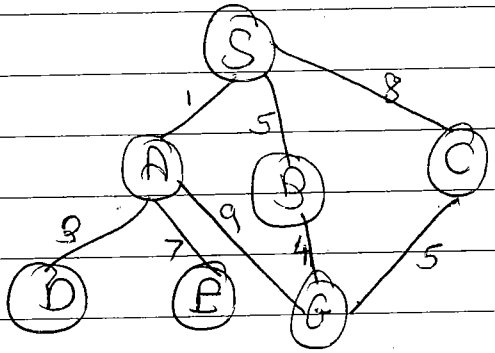
3. Time Complexity : $O(b^d)$

4. Space Complexity :- $O(b^d)$.

2) Uniform Cost Search:-

- Extension to BFS, it expands the next node which has lowest path cost.
- The main purpose of this method is to reduce the total cost of path traversal.
- Priority Queue is used.
- Variant of Dijkstra's Algo.

- Example:-



Expanded Node	Node list [S]
S	A(1), B(5), C(8)
A	D(4), B(5), C(8), E(8), G(10)
D	B(5), C(8), E(8), G(10)
B	C(8), E(8), G(9), G(10)
C	E(8), G(9), G(10), G(13)
E	G(9), G(10), G(13)
G	[ ]

G(9) is removed first. So answer is S - B - G.

classmate
Date
Page

- Properties:-

1. Completness:- Yes.
Sol^n is reached if its exists.

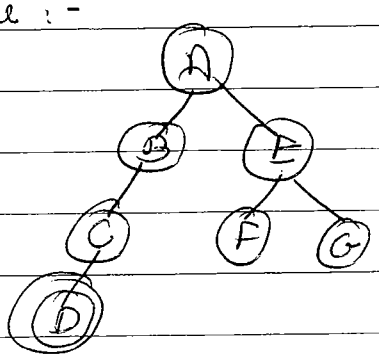
2. Optimality:- yes
Returns the least cost path.

3. Time Complexity:- $O(b^d)$

4. Space Complexity:- $O(b^d)$

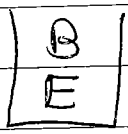
3. Depth First Search Tree.
- Explores one path to the deepest level and then backtracks until it find a goal state.
- Implemted using Stack.
- Does not gurantee to provide optimal Sol^n.
- Used when search space is large.

- Example:-

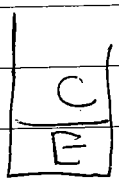


Soln:-

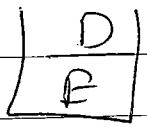
1. Push root node 'A' on stack. Is it a goal node? No. then pop A and add its successors.

2. 

3. Pop B. Add its successors.



4. Pop C. Add its successor D.



5. Pop D. It is a goal node so stop. And pop all other contents.

- Properties of DFS:-

1 Completness :- No.
If the depth of tree is infinite.

2 Optimality :- No.
It stops at the first goal state it finds, no matter if there is another goal state better than that.

- Time Complexity :- $O(b^d)$
- Space Complexity :- $O(bm)$

* Advantages of DFS:-
1. Simple to implement.
2. Needs relatively small memory space.

* Disadvantages:-
1. Not complete (will not find soln in all cases.
2. Not optimal.

* Advantages of BFS:-
1. Complete.
2. Optimal because longer paths are never explored until a shorter ones have already been enumerated.

* Disadvantages of BFS:-
- Requires more memory as each traversed nodes is stored.

4) * Depth Limited Search(DLS):-
- It is an extension to DFS technique.
- The drawback of DFS is that sometimes it can lead to infinite path ~~node~~ ode to the dead ends.

- In DLS, a limit on depth of DFS can be set.
- The basic idea is to not allow expansion of tree after the given depth.

• Properties:

1. Not Complete:-
- Since solution may exist but not found due to limit depth on search.
- It is complete when depth limit is greater than of Sol^n depth.

2. Not optimal

3. Time Complexity :- $O(b^l)$

4. Space Complexity :- $O(bl)$

5. * Iterative Deepening Search (IDS)
- Enhanced version of DLS.
- Sometimes depth limit restricts DLS from finding Sol^n, since Sol^n may exist beyond prescribed depth limit.
- IDS is used to address such problem.
- If the Sol^n is not found till the given depth limit then depth limit is incremented by 1.
- Sol^n is searched at this level.

- If soln is still not found the depth limit is again incremented by 1 and process goes on.

- Properties:-

1. Complete:- Yes
2. Optimal :- Yes
3. Time Complexity :- $O(b^d)$
4. Space Complexity :- $O(b^d)$

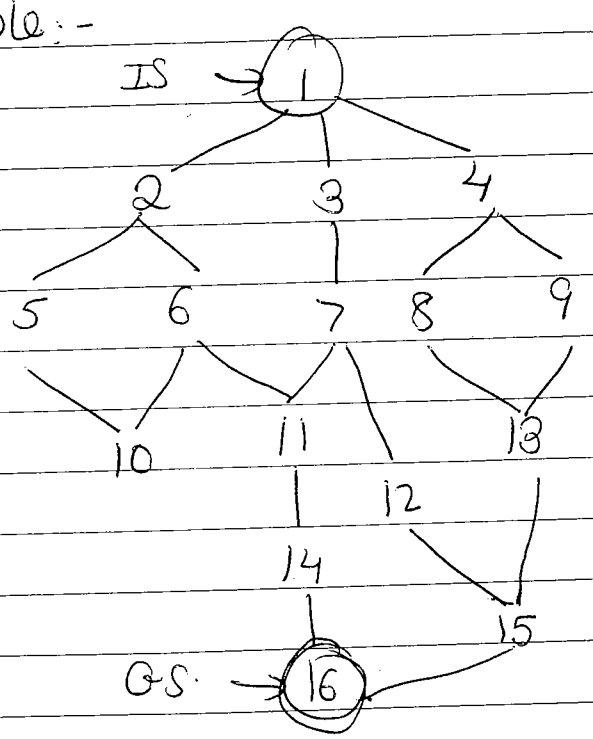
- limitation:-
  - Regeneration of tree after reaching the depth limit
  - All previous search results are discarded and search start from scratch
  - IDS method is preferred when search space is large and depth factor is known.

6. Bi-directional Search:-
   - Search is carried from both ways i.e from forward as well as backward direct
   - Forward search is started from initial state and backward search is started from goal state.

- At some point they intersect. If they intersect Solut" exist or else no solution is present.
- It is like BFS carried from both ends.

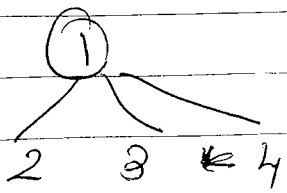
- Properties:-
1. Complete and Optimal
2. Time and Space Complexity :- $O(b^{d/2})$

- Example:-

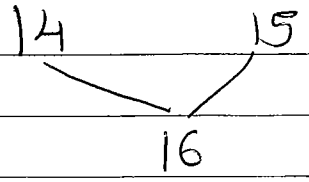


Soln:-

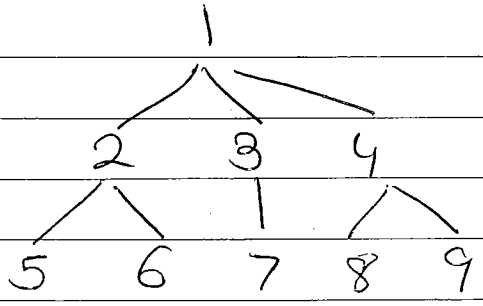
1. Start (1)
2. Goal (16)
3. Start



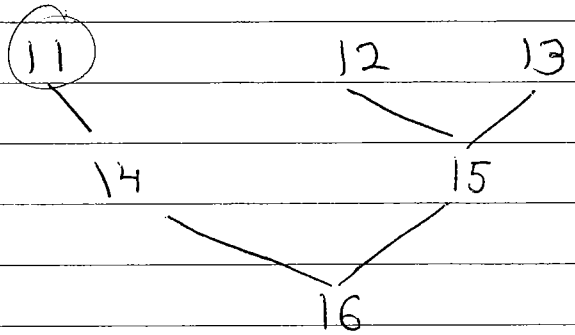
4 Goal:-



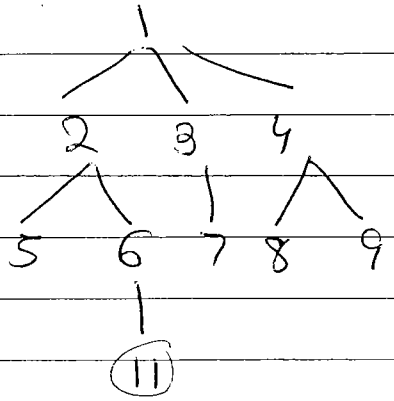
5 Start:-



6. Goal



7.



Intersect - Goal ~~for~~ Search successful.

classmate
Date
Page

* Heuristic / Informed Search

- Was developed to overcome the drawbacks of uninformed search i.e. high time and space complexity.
- Informed search uses information about the domain or knowledge abt the problem to move towards goal state.
- These methods do not always find the best solution but they guarantee to find good solution in reasonable amount of time. (Not optimal).
- These methods are used to solve problems that require longer time.

- Following are the Informed Search Methods-
1. Best First Search
2. Greedy Methods
3. A* Search
4. Iterative Deepening A*.
5. Heuristics.
6. AO* Search

* Heuristic Function :- $h(n)$

- It is a fun. that guides the decision of selecting a path while aiming to reach a goal node.
- A heuristic function $h(n)$ provides an estimate of the cost of path from

the given node (n) to reach the closest goal node.

- Informed search algos use heuristic fun which guides them to reach goal states in less amount of time

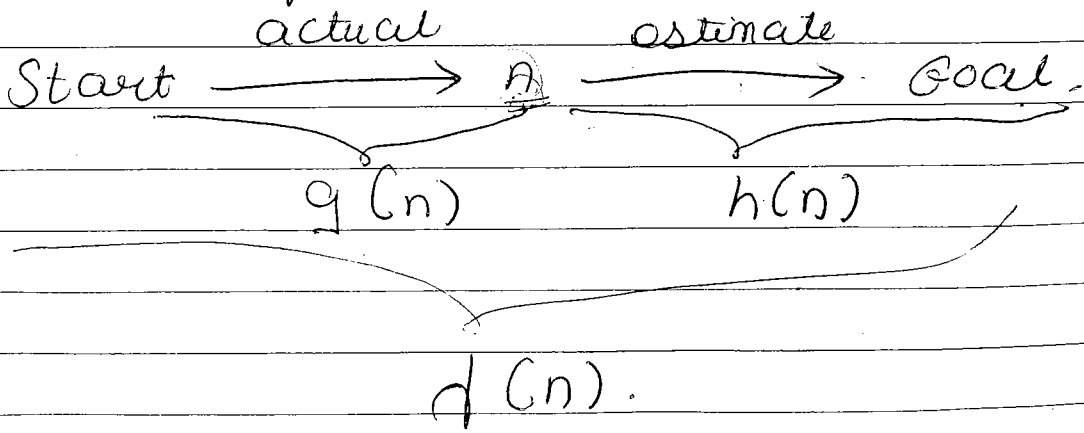
* Search Notations:-

1. $f(n)$ is a evaluation fun^{n} that estimates least cost solution through node n.

2. $h(n)$ is heuristic function that estimates least cost path from node n to goal node.

3. $g(n)$ is cost fun^{n} that estimates least cost path from start node to node n.

$$f(n) = g(n) + h(n)$$



classmate
Date
Page

1) Best First Search:-
- Here nodes are expanded and explored one by one.
- An evaluation funct^n $f(n)$ is used to decide which node has to be expanded next.
- The node which has lowest value of evaluat^n is selected for expansion.

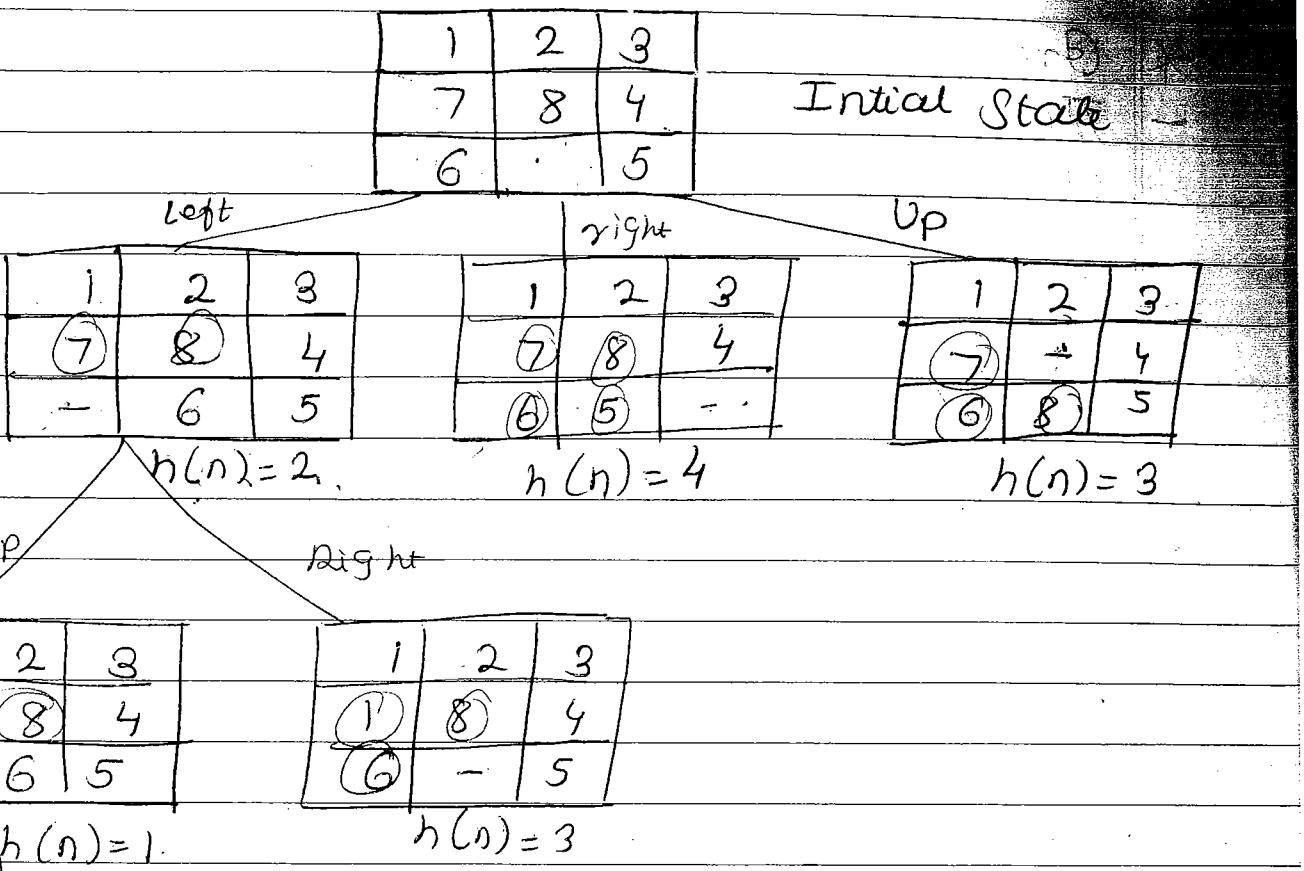
Procedure:-
1. Start with root node.
2. The node having lowest value for $f(n)$ is selected.
3. Repeat the process till you get the goal node or till all nodes are explored.

* Example:-
1) 8 puzzle problem:-
Given - Three possible moves → left, up, right.
- Heuristic here is "No. of tiles not in correct position".
- Here smaller value of heuristic leads closer towards the goal.

1	2	3
8		4
7	6	5

- Goal State.

$f(n) = h(n)$



- Properties of Best First Search:-

1. Not complete - Can lead to infinite path.
2. Not optimal -
Reason is that select^n of minimum value of h
The search could also lead to dead end.
3. Time and Space Complexities:- $O(b^m)$.

Note - Heuristics consider best soln at moment.
(i.e. they consider current best^n soln and not future best^n soln).

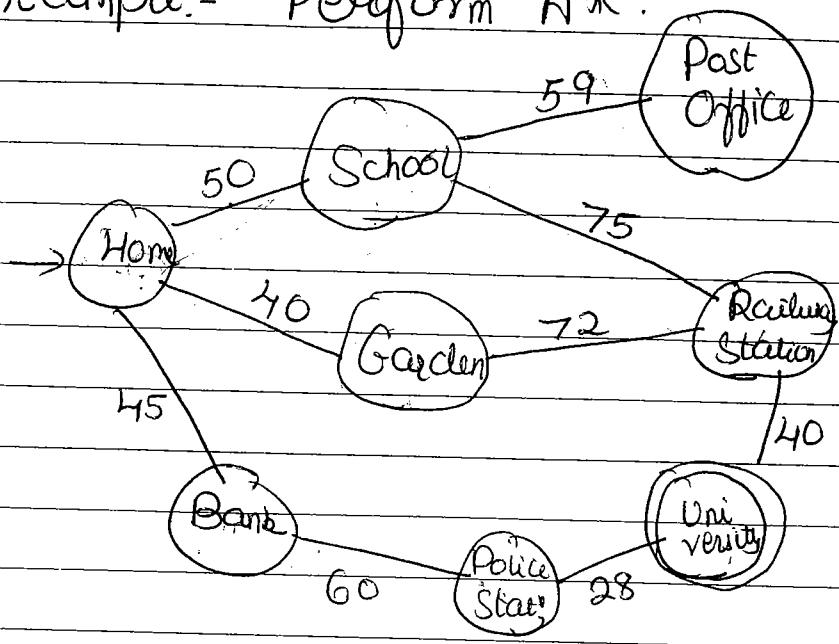
* Greedy Search.
- They try to expand the node that is closest to the goal in order to lead to a solution quickly.
- Thus, it evaluates nodes by using heuristic function i.e. $f(n) = h(n)$.
where $h(n)$ is estimate cost of cheapest path from node 'n' to goal node.
- It minimizes estimated cost to reach the goal. The node that is closest to the goal node is expanded first.
- Best first search is greedy best first search.

• Properties:-
- It is not complete (infinite path)
- It is not optimal.

* A* Search.-
- It is a combinat^n of Uniform Cost Search and greedy search techniques.-
- Uniform Cost search considers $g(n)$ i.e exact cost from start state to actual node n.
- Whereas Greedy Search considers $h(n)$ i.e heuristic path cost from node n to a goal state.-

- ~~Heuristic~~ Evaluat^n function for A* is
$f(n) = g(n) + h(n)$.

* Example:- Perform A*.



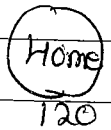
IS = Home, GS = University.

- Following are the given heuristic values:-

Home = 120
Bank = 80
Garden = 100
School = 70
Railway Statn = 20
Post Office = 110
Police Station = 26

Soln:-

Step 1:-

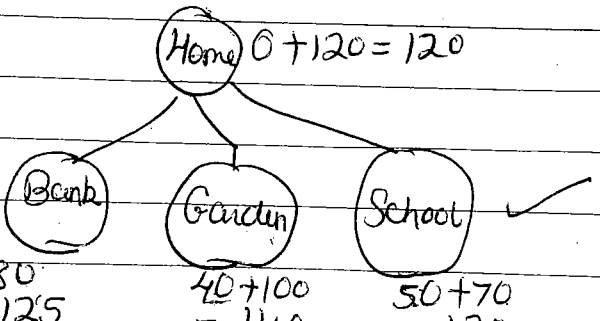


$f(n) = g(n) + h(n)$ — distance from start node to this node

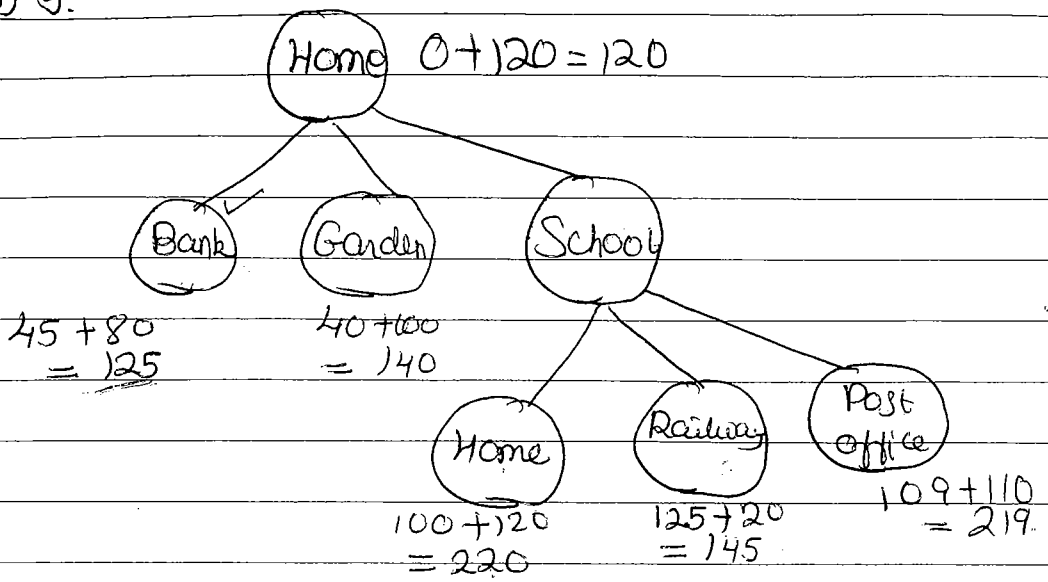
$= 0 + 120$

$f(n) = 120$

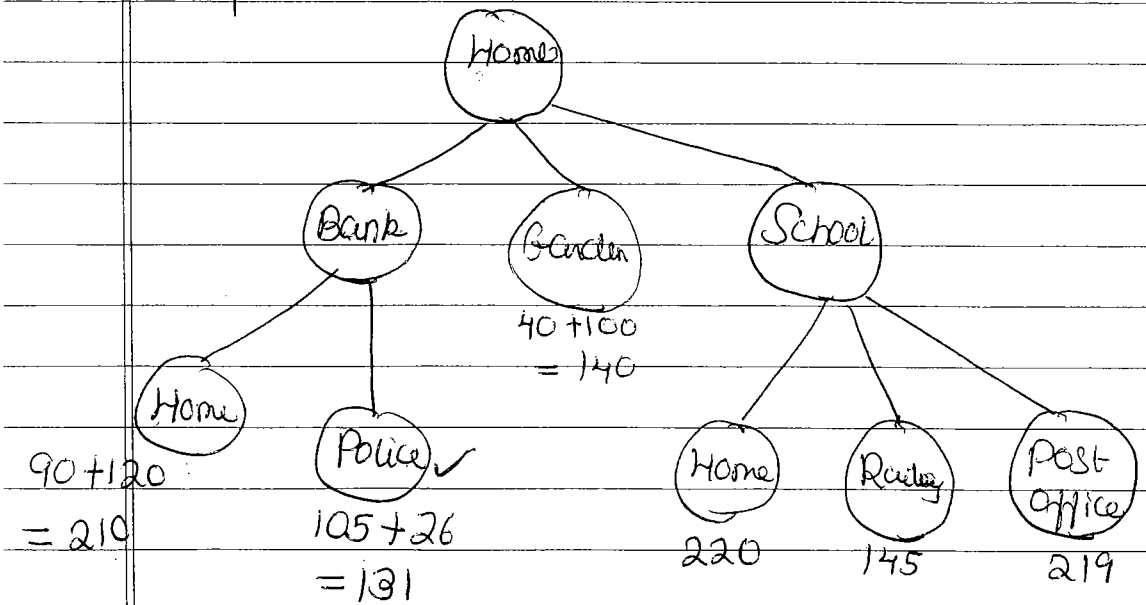
2)

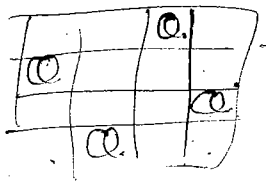


Step 3:-


Home 0+120=120
Bank
Garden
School
45+80 = 125
40+100 = 140
Home
Railway
Post office
100+120 = 220
125+20 = 145
109+110 = 219

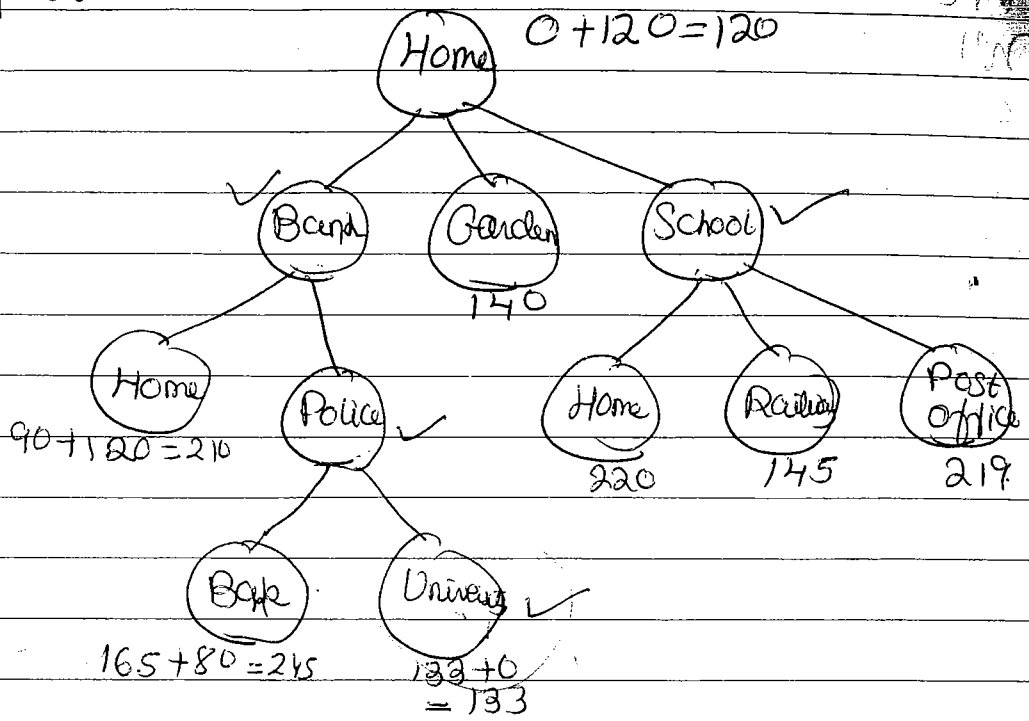
- Step 4:-


Home
Bank
Garden
School
40+100 = 140
Home
Police
Home
Railway
Post office
90+120 = 210
105+26 = 131
220
145
219



→ 4 Queen's.

• Step 5:-



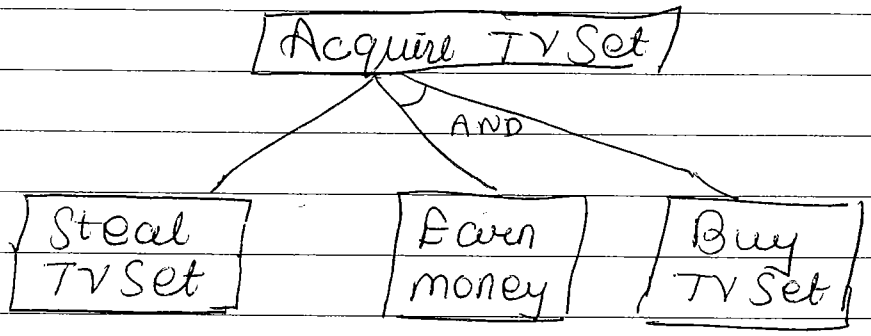
* Properties of A* Search:-

1. Completness :- Yes.

2. Optimal - A* is optimal if h(n) is admissible heuristic.
- Admissible heuristic means h(n) never over estimates the cost to reach the goal node.

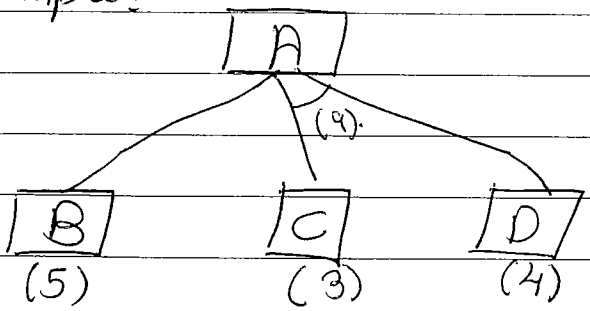
- Time complexity depends on heuristic func
- Space complexity is the main issue here. Since all nodes that are generated are kept in memory
- So, A* is not recommended to be used for large-scale problem.

* AO* Search - Problem Reduction:-
- When a problem can be divided into a set of problems, where each sub problem can be solved seperately and combination of these solutions will lead to a main solution ~~At.~~
- ~~that~~ AND - OR graphs / trees are used for representing the soln



Example of AND - OR graph

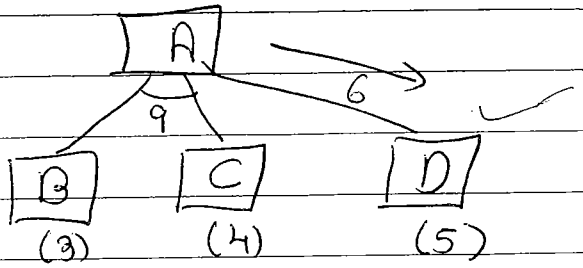
* Example:-



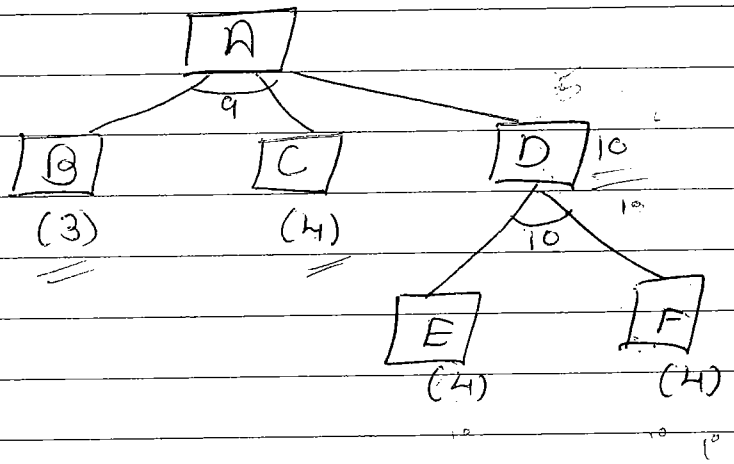
- Consider $h(n) = 1$ for all
- We will select node C because its having least value among all other B nodes.
- But choosing B will be a better because node C is ANDed with node D.
- So cost will be $(3+1+4+1) = 9$.
- And node B will be $(5+1) = 6$. So select node B.

Step 1:- A

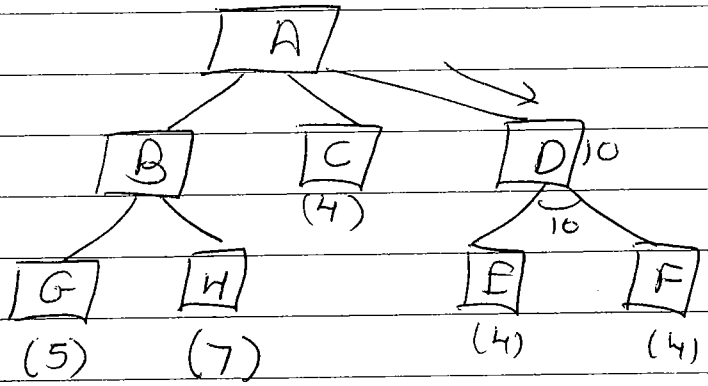
Step 2:-



Step 3:-



Step 4:-



* Generally A* cannot work efficiently with AO* trees/graphs.
- AO* always finds optimal solution.

Local Search Algorithms and Optimization Problems

- Informed and uninformed search concentrate on path through which goal is reached.
- But often if the problem does not demand the path of sol^n and it expects only the final config of sol^n then we have diff. types of problem to solve.

For eg:- 8 queens problem, IC design, job shop scheduling etc are some of the problems that do not concentrate on path.

- Local search algos operate on single state rather than multiple paths.
- They generally move to the neighbours of current state.
- No req of maintaing path in memory.
- They are used for solving optimizat^n problems.

- Advantages:-

1. They use very little or constant amount of memory.
2. They have ability to find reasonable sol^n for infinite state spaces.

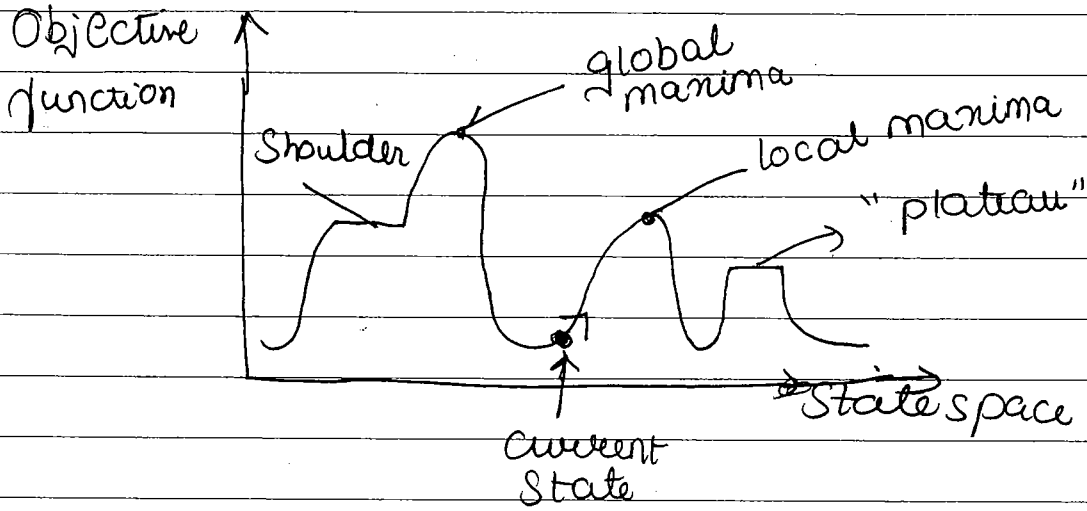
- Following are some of the local search optimizat^n problems:-

1. Hill Climbing Search
2. Simulated Annealing.
3. Local Beam Search
4. Genetic Algo.

1) Hill Climbing Search:-

- This algorithm generally moves up in the direction of increasing value, that is, up hill.
- The basic idea is to always select a state that is better than the current state i.e. it always move to a neighbor which has a better score.
- It terminates when it reaches a "peak" where no neighbor has a higher value/score.
- This algo. only looks out for immediate neighbors of current state.
- It is similar to greedy local search which means it only considers immediate neighbors of current state.
- It does not maintain a search tree rather stores only the current node data structure.
- Since it keeps no history, it cannot recover from failures of its strategy.
- This method works in small setting of specific environment.

- This strategy works very well but sometimes it may not be appropriate to be used in real life scenarios due to shape of entire space.
- Basically heuristic helps in deciding the direction of search.



- Hill Climbing procedure:-

1. Start from initial node.
2. Consider all neighbors of current state.
3. Choose neighbour with best quality and move to the state.
4. Repeat step 2 to 4 until all neighbouring states are of lower quality.
5. Return current state as solution state.

- This method is a heuristic based search method.

- Problems with Hill Climbing :-

1. Local Maxima :-
- This is a state better than the local region or neighboring states but not global maximum.
- This occurs since a better solution exists but is not present in vicinity (or near) the current state.
- Solution :- Backtrack to some earlier node and try to move in some other direction.

2. Plateau -
- It is a flat area of search space where all neighboring states has same value.
- Algorithm fails to determine best direction to move on.
- Solution :- Big jump has to be taken in some direction.

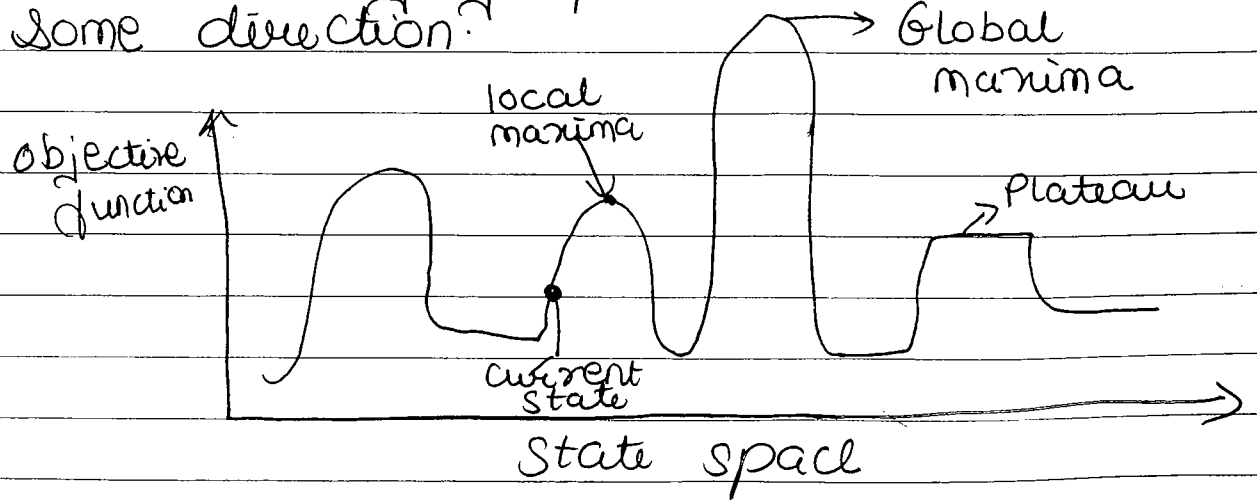


Fig :- Hill Climbing : Problems.

8. Ridge :-
- It is the search space at higher altitude than surroundings.
- It cannot be traversed by a single move.
- It is a special kind of local maxima
- Sol^n :- Move in multiple directions at once.

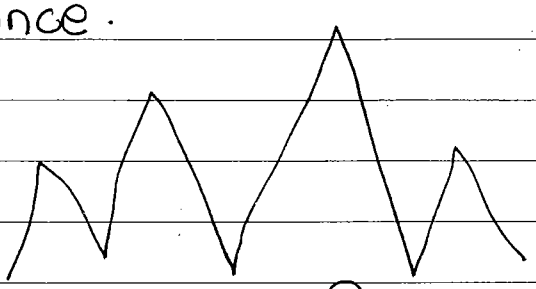


Fig :- Ridge.

• Applications :- Travelling Salesman problem, 8 Queen prob^n.

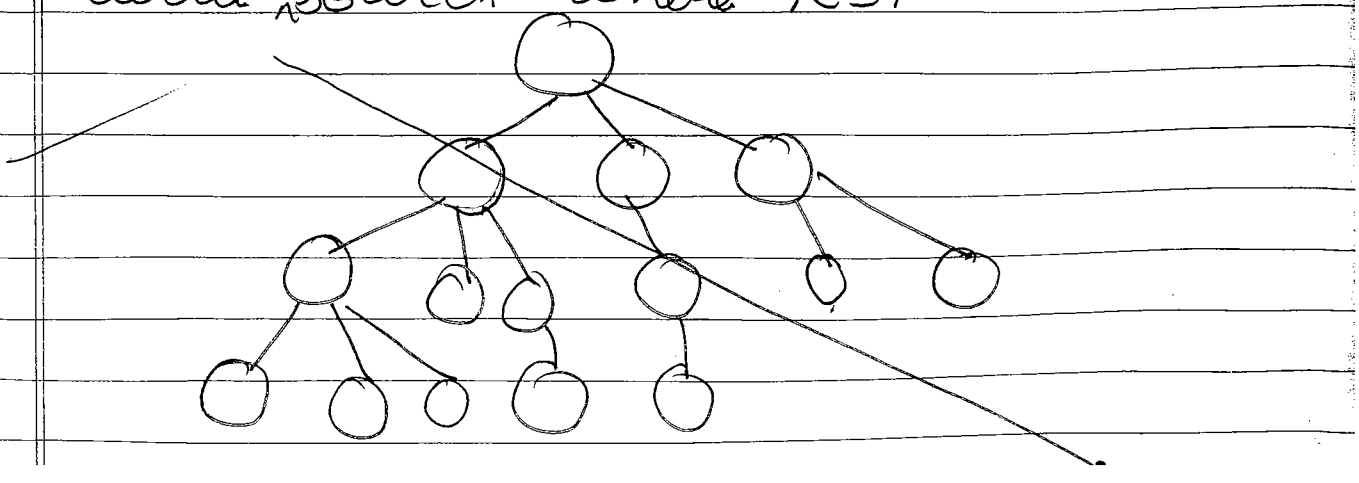
2) Local Beam Search :-
- It is a heuristic search algorithm.
- This algo. expand the most promising (good) nodes from the limited set.
- There can be more than one good node selected for expansion, say K nodes.

• Algorithm :-
1. Maintain K states instead of single state.
2. The search begins with K randomly generated states.
3. At each iteration, find all

Successors of K nodes. &
4. Check if a goal state is found. If found then halt or else select K best of successors.

- Example:-
Assume that we want to select best engineering students from the country, the steps are as follows:-
1. First select K states from country with best engg. results.
2. Select K cities with best results.
3. Select K colleges with best results.
4. Select K courses with best results.
5. Select K divisions
6. Select K students
7. Select a student among them with max. marks.

- K is the no. of best nodes expanded at each level. So K is the width of beam.
- Hill climbing is a special case of local beam search where K=1.



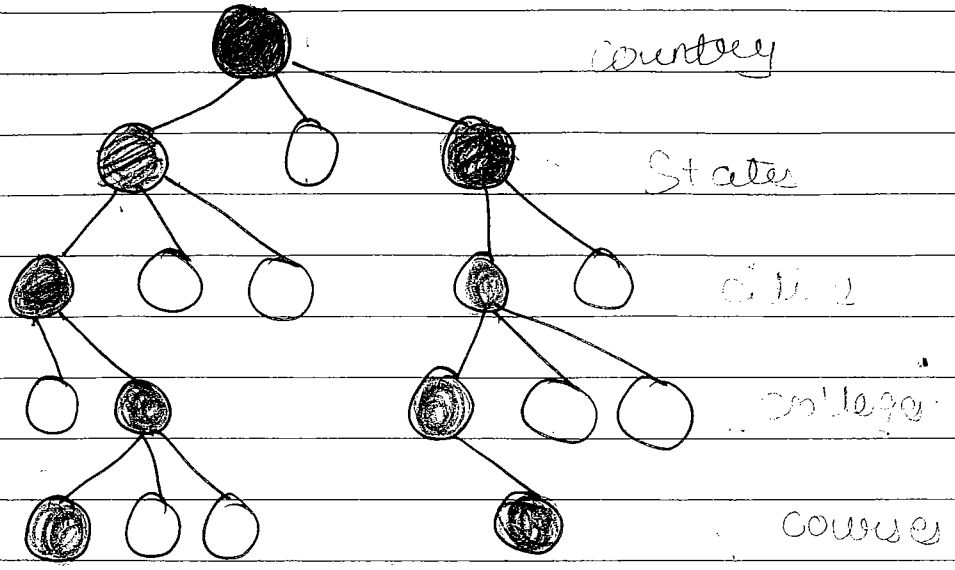


Fig:- Local Beam Search

- Fig. shows that at every level best k nodes are selected.

* Adversarial Search Problems:-
- Searches in game playing are different.
- Game can provide either perfect or imperfect information.
- They generally have multi-agent environment.
- The environment is co-operative and competitive.
- Foll. are the adversarial search methods.-

1. Miniman Algorithm
2. Alpha-Beta Pruning.

* Game Terminologies:-

1. Game -
A game means sort of conflict in which n individuals or groups participate.

2. Rules:-
Rules are the condition under which game is played.

3. Strategy -
List of optimal choices for each player at every stage of game.

4. Move -
The way in which game progresses from one stage to another.

5. Payoff / Outcome / Utility:-
- A payoff is a value associated with each player's final ~~sitt~~ situation.

OR

- It refers to what happens at the end of the game.

* Types of Games:-
Games are classified as follows:-

1) Based on Information:-

Games of Perfect Information

Games of Imperfect

a) Games of Perfect Info.-
- Here all moves are not known to everyone.
Eg:- Chess, tic-tac-toe.

b) Games of Imperfect Information.-
Here all moves are not known to everyone.

2] N person games:-
- It involves n ~~per~~ players in the game.

3] General zero-sum games:-
- Games whose sum of payoff of all players result in zero.
Eg:- In chess one person wins (payoff +1) and other loose. So will have payoff value -1. Hence the sum of both players score/payoff i.e (-1+1=0).

4] Non-zero Sum Game:-
- These are the games who outcome are non-zero value.
It has 2 type:-
a) Negative Sum Games (Competitive Games)
- Here nobody really win, rather everybody loose.

Unit I
Introduction

- Heuristic Search :-

* Variable Neighbourhood Descent :-

- Generally, we can define different neighbourhood functions for a given problem.
- Every candidate sol^n has more than one neighbour sol^n; the choice of which one to move to is taken using only info about the solutions in the neighbourhood of the current one, hence the name local search.

- Neighbourhood functions that are sparse lead to quicker movement during search, since the algo. will have to inspect few neighbors.

	searching movement	stuck
sparse	quick	↑
dense	search slows	↓

- But, there is a greater probability of getting stuck on local optimum.

- This probability of getting stuck becomes lower as neighbourhood $funct^n$ become denser but then search progress also slows down because the algorithm has to inspect more neighbours before each move.

- Variable Neighbourhood Descent (VNDN) tries to get best of the both.

Date / /
Page

- It starts searching with sparse neighbourhood funct^n.
- When it reaches an optimum, it switches to a denser function.
- The hope is that most of the movement would be done in earlier rounds, & that time reqd. for searching will also be less.

Algo:-

1. node ← start
2. for i ← 1 to n
3. do moveGen ← Move Gen(i)
4. node ← Hill Climbing (node, moveGen)
5. return node.

* Tabu Search:-

- The basic idea here is when there are no better choices, instead of terminating as in local search, it tries to continue searching.
- Tabu search modifies the termination criteria.
- The algo. does not terminate on reaching a ~~loc~~ maximum, instead it continues to search until some other criterion is not

- Local search method has a tendency to stuck in suboptimal regions or on plateaus where many solut^{n}s are equally fit.

- Tabu search enhances the performance of local search.
- Firstly, at each step worsening moves can also be accepted if no improving move is available (local minimum)

- In addt^{n}, prohibitions (tabu) are introduced to discourage the search from coming back to previously visited sol^{n}. (should not be visit again).

i.e. if a potential sol^{n} has been previously visited within short period of time, it is marked as "tabu", so that algo. does not consider it repeatedly.

- When a tabu (forbidden) move has a sufficiently attractive evaluat^{n} where it would result in a sol^{n} better than any visited so far, then its tabu classificat^{n} may be overridden. A condt^{n} that allows such an override to occur is called an aspiration criterion.

- Tabu search flow is depicted as shown in below:

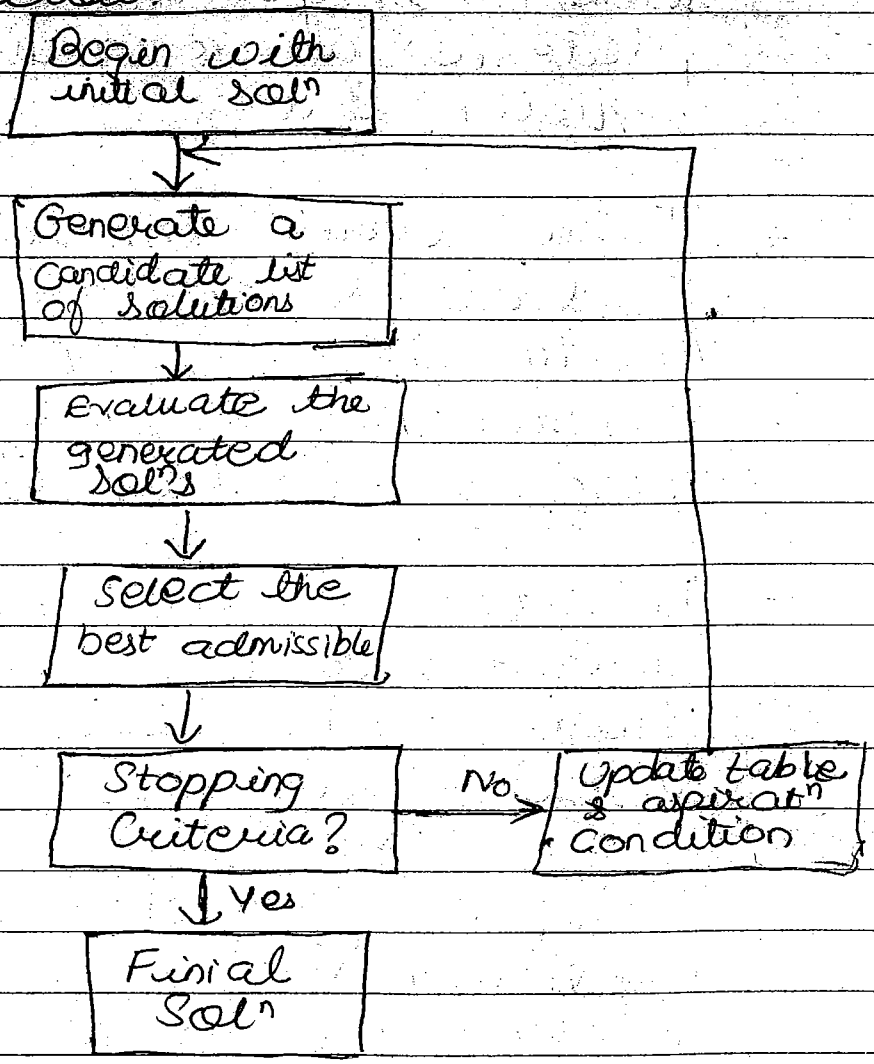


Fig:- Tabu Search: Flow.

Adv:-

- It allows to exit from sub-optimal regions by making non-improving soln to be accepted.
- Use of tabu list improves efficiency.

- Disadvantage:-

- Cannot find global optimum in some cases.

Date 1 1
Page

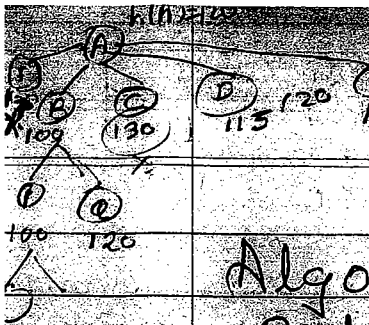
* Optimal Search:-
• Memory - Bound Heuristic Search:-
- Generally, we tend to run out of memory before time!
- This occurs when the entire search space paths are stored & the only soln to overcome this is to remember partial solutions.

- In order to overcome space requirements, memory-bounded heuristic search is used
- Foll. algos. fall under this category:-
1. Iterative Deepening A* (IDA*).
2. Recursive Best First Search (RBFS).

1] IDA*:-
- IDA* resolves the problem of A* where the (less) memory problem is overcome, and at the same time, optimality is also maintained.

- In IDA*, at each iteratn DFS is applied.
- A track is maintained of the costs i.e. $f(n) = g(n) + h(n)$ of each & every node that is generated.

- "Whenever a node is generated whose cost is more than the threshold of that iteratn, the path is discarded."
- Then backtracking is applied.
- Initial cut-off is set to be equal to



Date / /
Page

Algo:-

1. Set limit = h(root). This is f-limit.
2. Prune if f(node) > f-limit.
3. Set f-limit to be equal to the min. cost of any node that is pruned.

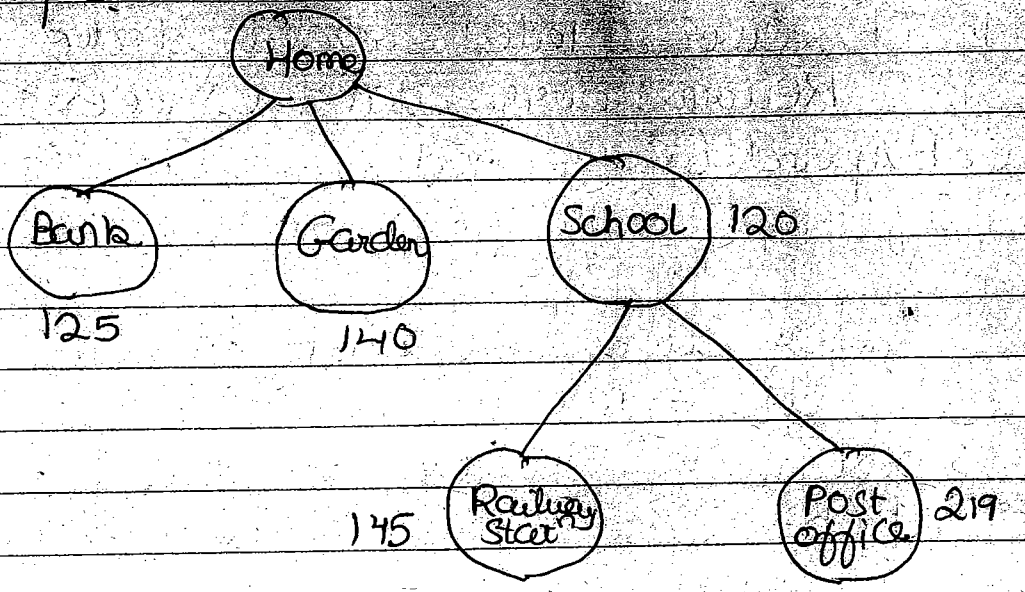
- IDA* performs series of depth first searches.
- If admissible heuristic exists, an optimal sol^n is guaranteed.
- Hence, IDA* is optimal in terms of time & space.
- Moreover, it reduces the overhead of managing open & close list.

- Disadv:-
  Excessive node generation.

* Recursive Best First Search (RBFS):-

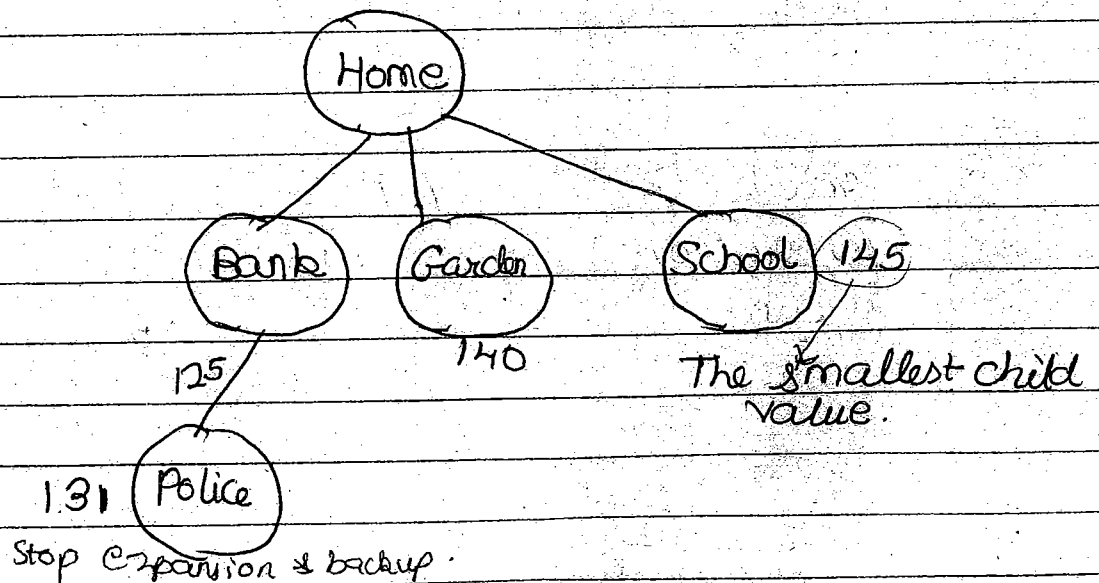
- The basic idea here is to remember the best path or the best alternative & backtrack if the best alternative (i.e lowest f(n)) node. And
  Backtrack if the best first gets very expensive i.e. when the cost exceeds that of previously expanded node.
- As the decision to backtrack is taken the f-cost of current node (i.e. expanded)

Example:-



Stop expansion & backup since $f(n)$ of Railway stn & P.O. is more than $f(n)$ of School.

Hence, we get,



Example:- RBFS Snapshot.

AU

[illegible] state = goal state
Return curr_state (success!)
2. Expand (curr_node) → children.
3. If empty (children) return failure
4. Do
for each c (child of curr_node).
f[c] = 219 f[c] = maximum (g(c) + h'(c), f[curr_node])
145 219 120
best_cost = 145 best cost = lowest of f-cost i.e., f[c].
145 > 120 if (best cost > prev f-cost found) return failure.
Consider alternative f-cost.
145
goto step1 with (best_cost node,
219 125
min (f-cost, alternative f-cost)).

- Space Complexity :- O(bd).

- Drawback :- of RBFS & IDA* is generation of same states again & again.

Practical

b) Positive Sum Games (Co-operative)
- Here all players have one goal that they Contribute together.

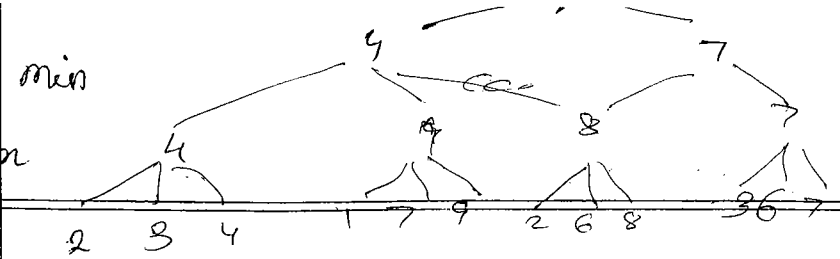
Eg:- Educational games, building block, etc.

(1)

1) Miniman (MM) Algorithm:-
- It can be used for games having 2 players.

Eg:- Chess, tic-tac-toe, etc.
- These are basically Logic Games.
- The algo. is effetive for games having few logical states transitions from the current state.
- Miniman is exhaustive or uninformed search.
- Basically here two players are involved i.e. MIN and MAX.
- Starting with current game position, a search tree is generated.
- Basically there are 2 views:- MAX view and MIN view.
- The end position (or the end result) is evaluated from the MAX's point of view.
- The values are assigned based on evaluation.

- The nodes that belong to MAX are given maximum value of its children.
- The nodes that belong to minimum value MIN are given minimum value of its children.



(2)

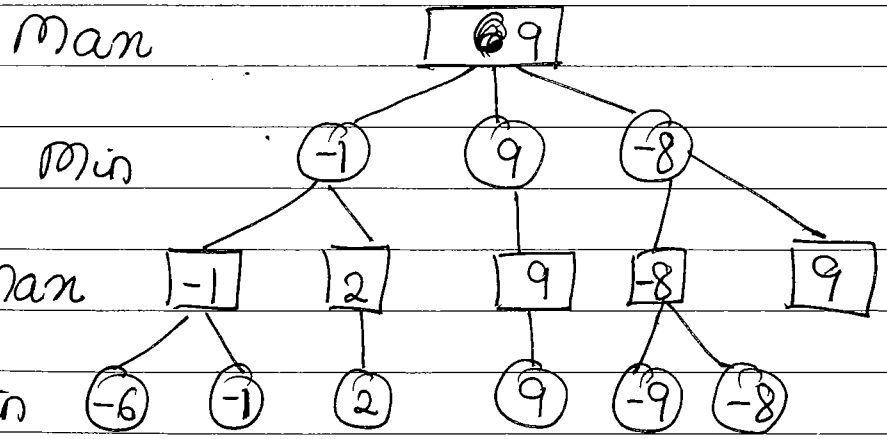
- In short, the player MAX tries to move to a state of maximum value, while player MIN tries to move to a state of minimum value.

- MAX is represented by ◎ or △ or +.
- MIN is represented by ○ or ▽ or −.

MAX — ◎ □, △, +.
MIN — ○, ▽, −.

- Generally, game starts with MAX (assumption)

Example:-



Explanation:-

Step 1:- Start from leaf nodes (from far left).

For max:- Compare -6 and -1. Choose the maximum value i.e. -1. So assign it to parent node.

Similarly, compare -9 and -8. Choose max value i.e. -8 assign it to its parent.

Step 2:- For MIN.
Compare -1 and 2. Choose minimum as the parent node is of MIN. So select -1.
- Similarly compare -8 and 9. Choose minimum value i.e. -8.

Step 3: For max
Choose maximum value out of -1, 9, -8 i.e. 9. So assign 9 to its parent node.

- Properties of Miniman Algorithm:-

1. Complete - Yes.
The element is found if present in the search tree.

2. Optimal - Yes.

3. Time Complexity :- $O(b^m)$
where b is branching factor i.e. no. of children to a node.
m - maximum depth of the tree.

4. Space complexity :- $O(bm)$.

- Miniman (MM) algo. is inefficient for games with huge search space since it will take longer time to compute.

classmate
Date
Page

2] ~~Alpha-Beta Pruning~~:-

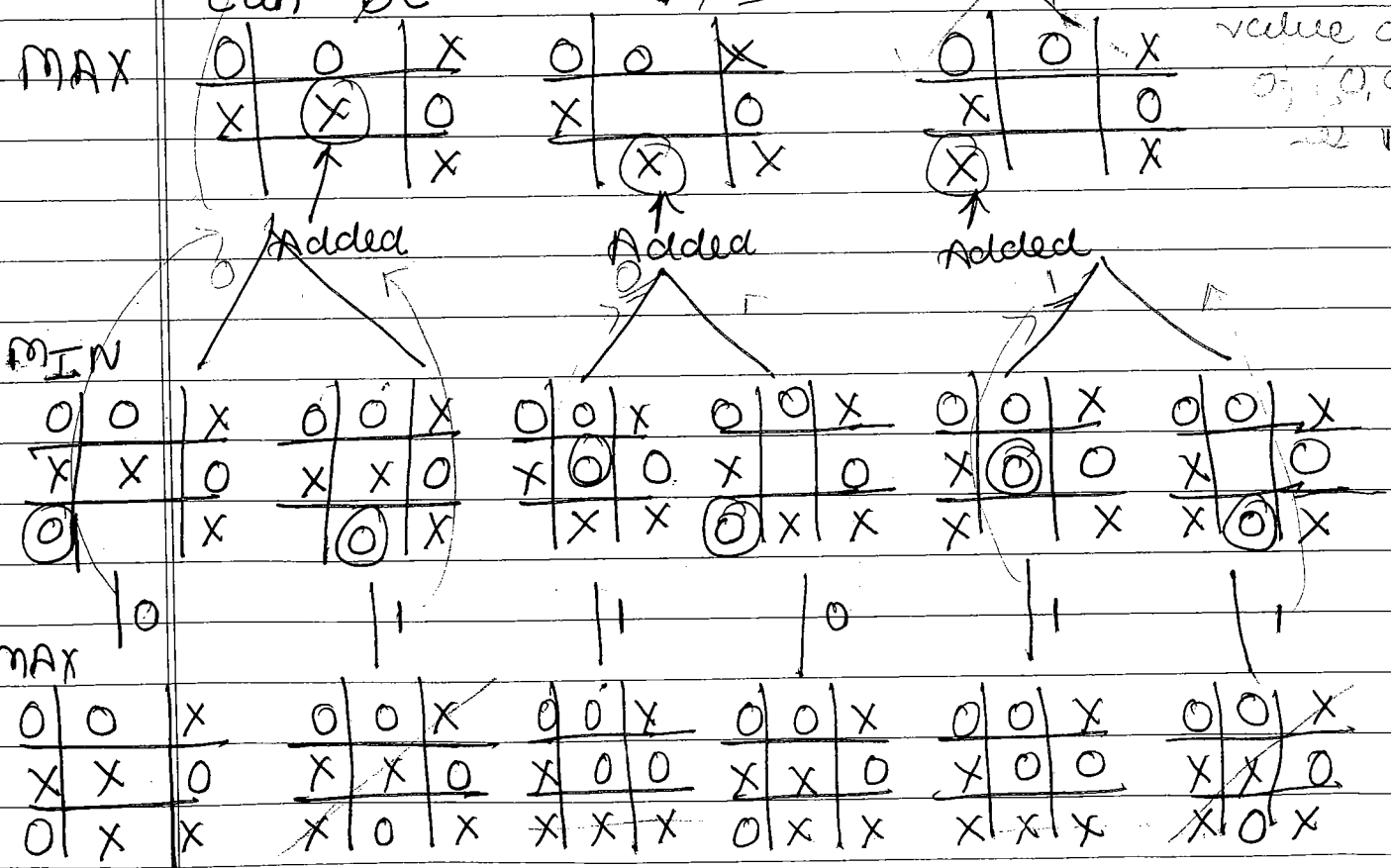
* Example:-

Q- Find the best move using Miniman (MM) algo for tic-tac-toe problem where intial state is given as:-

O	O	X
X		O
		X

Here the best move.

Sol:- First MAX (Player1) starts playing by inserting X. Now there are 3 blank spaces. So 3 possible states can be



- Give point as 1 if MAX wins or if MAX looses then payoff will be 0.

2006
2009

* Alpha - Beta Pruning:-
- The miniman is a exhaustive or uninformed search algo.
- So the main problem is that it has to examine large number of moves.
- So, in order to overcome this limitation alpha - beta pruning proposes to provide a solution without looking at every node in the game tree.
- Pruning refers to the elimination of nodes found to be while searching and evaluation.

• Alpha Cut off :- (For MAX node):- Done
- α ($\geq$) is the value of maximum/best choice found so far. (i.e. max. value found so far).
- If any value is worse (or less than) α, MAX will avoid it.

For eg:-

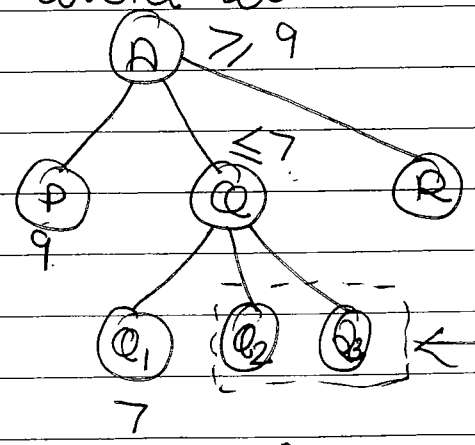
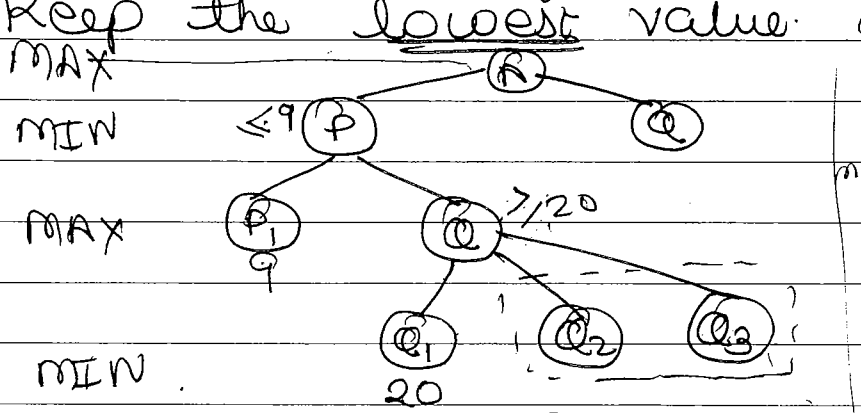


Fig:- Alpha - Cut.

- Assume value of P = 9. So A node (MAX) will have value ≥ 9. i.e. $\alpha = 9$ (for A).
- Node Q1 = 7 (Given). ∴ Q will be ≤ 7 (since Q is at the MIN level).

- Node A will have value $A \geq 9$. So exploring the nodes of branch Q ~~further~~ i.e. node ~~Q~~ Q_2, Q_3 is not necessary. (Since $7 \leq 9$). This is called as alpha-cut.
- It means Keep the highest value. Ignore the smaller one.

- **Beta Cut-Off:-**
- β ($\leq$) is the minimum/worst (i.e. lower value) or choice found so far.
- Keep the lowest value. OR



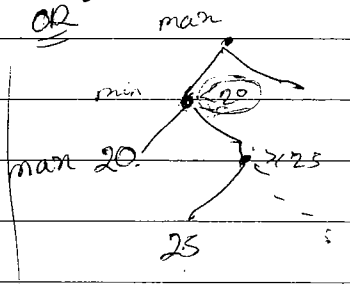


Fig:- Beta Cut.

Since parent node $P_1 \leq 9$ ~~and~~ and $Q \geq 20$. and their parent node P will have minimum value 9. So P will ≤ 9. Hence discard branch Q.

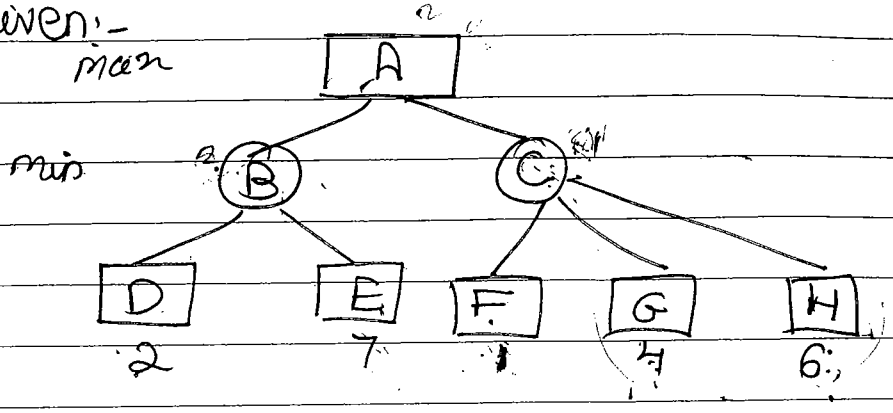
- **Example:-**
Consider the foll. tree. Perform alpha-beta pruning on it

1,05,976.

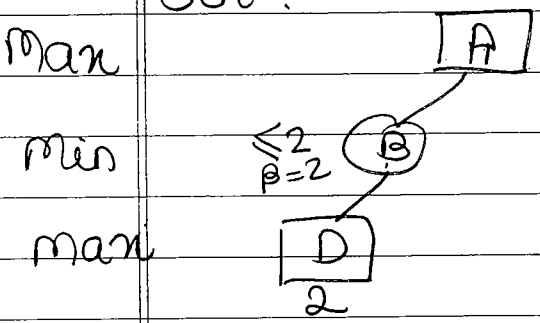
$\leqslant = \beta$

$\geqslant = \alpha$

Given:-

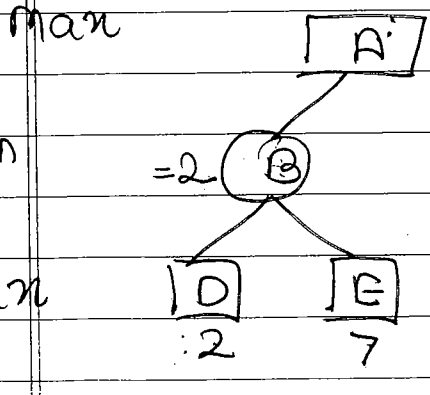


Soln:-



As D=2 so min value of B will be ≤ 2 i.e. $\beta = 2$.

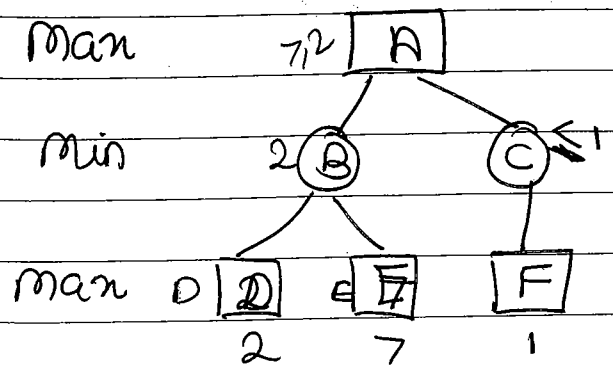
- Now backtrack and find successor of B which is E. E=7 (Given).



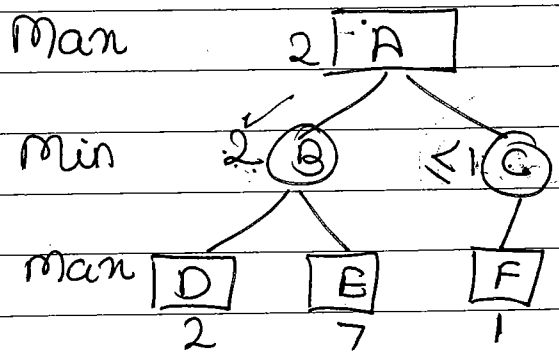
Select minimum value out of 2 and 7 i.e. 2. Hence B=2.

- Now backtrack to node A and find its successor i.e. node C. Go till you get leaf node from path C i.e. node F.

- Node F = 1 (Given).



C will be ≤1 since it is a MIN node. Now out of B and C choose maximum i.e. 2. Hence A = 2.



- Out of value 2 and ≤1 we choose value as MAX node.
- So there is no need to further process node of C since its value will not be selected. So prune the remaining 2 children of node C.

* Solve one more example from TM or in Slides.