⊛ Artificial Intelligence ⊛

⊛ Chapters :-

(1) Introduction
(2) Problems, State Space Search & Heuristic Search Techniques
(3) Knowledge Representation
(4) Symbolic Reasoning Under Uncertainty
(5) Probabilistic Reasoning
(6) Game playing
(7) Planning
(8) Natural Language Processing
(9) Connectionist Models
(10) Expert Systems
(11) Generic Algorithms
(12) Introduction to prolog.

⊛ Chapter : (1) : Introduction ⊛

Q.(1) What is Artificial Intelligence ?

- Artificial Intelligence is the study of how to make computers do things which, at moment people do better.

- Artificial Intelligence can be viewed from a variety of perspectives. like,

  - perspective of Intelligence
  - perspective of Business
  - perspective of programming.

- From the perspective of intelligence, artificial intelligence is making machines "Intelligent" -- acting as we would expect people to act.

  - The inability to distinguish computer responses from human responses is called the "Turing Test".

  - Intelligence requires knowledge.

- From a Business perspective AI is a set of very powerful tools, and methodologies for using these tools to solve business problems.

- From a programming perspective, AI includes the study of symbolic programming, problem solving and search.

  - Typically AI programs focus on symbols rather than numeric processing.

  - problem solving i.e to achieve a specific goal.

  - Search - rarely access a solution directly. Search may include a variety of techniques

- It is the science and engineering of making inteligent machines, especially intelligent computer programs.

Q.② What is Turing Test?

- The Turing Test was developed by Alan Turing in 1950. He proposed that the "Turing test is used to determine whether or not a computer (machines) can think intelligently like humans"?

- The basic idea of the Turing Test is simple: a human judge engages in a text-based conversation with both a human and a machine, and then decides which of the two they belive to be a human.

- if the judge is unable to distinguish between the human and the machine based on the convesation, then the machine is said to have passed the "TT".

Q.③ What are the AI Problems?

- much of the early work in the field of AI focused on formal task, ~~as~~ such as game playing and theorem solving.

- Game playing and theorem proving shows the property that people who do

them well are considered to be displaying intelligence.

- Initially computers could perform well at those tasks simply by being fast at exploring a large number of solution paths and then selecting the best one.

- Humans learn mundane (ordinary) task since thier birth. They learn by perception, speaking, using language, and training. They learn from formal tasks and expert tasks later.

- Another early foray into AI focused on commonsense reasoning, which includes reasoning about physical objects and thier relationship to each other, as well as reasoning about actions and thier consequences.

- As AI research progressed, techniques for handling large amount of world knowledge were developed.

- New tasks reasonably attempted such as perception, Natural language understanding and problem solving in specialized domains.

⊛ Task Domains of Artificial Intelligence ⊛

Mundane Tasks

- Perception
  - Computer vision
  - Speech, voice
- Natural Language Processing
  - Understanding
  - Language Generation
  - Language Translation
- Planning
- Common Sense Reasoning
- Robot Control.

Formal Tasks

- Games - Go | chess | ckeckers
- Mathematics
  - Geometry
  - Logic
  - Integration & Differentiation
- Theorem proving

Expert Tasks

- Engineering
  - Design
  - Fault finding
  - Manufacturing
  - Monitoring
- Scientific Analysis
- Financial Analysis
- Medical Diagnosis

Q.④ Explain AI Techniques and its Types.

- Artificial intelligence problems span a very broad spectrum. They appear to have very little in common except that they are hard.

- AI Research of earlier decades results into the fact that intelligence requires knowledge.

- knowledge posses following properties:

  - it is voluminous.
  - it is not well-organized or well formatted.
  - it is constantly changing.
  - it differ from data. And it is organized in a way that corresponds to its usage.

- AI Techniques is a method that exploits knowledge that should be represented in such way that:

  - knowledge captures generalization.
  - it can be understood by people who must provide it.
  - it can be easily modified to correct errors and to reflect changes in the world.

- it can be used in many situations even though it may not be ~~actually~~ totally accurate or complete.
- it can be used to reduce its own volume by narrowing range of possibilities.

(4) There are Three important AI Techniques:

[1] Search :-

- Provides a way of solving problems for which no direct approach is available.
- It also provides a framework into which any direct techniques that are available can be embedded.

[2] Use of knowledge :-

- Provides a way of solving complex problems by exploiting the structure of the objects that are involved.

[3] Abstraction :-

- Provides a way of seperating important features and variations from many unimportant ones that would otherwise overwhelm any process.

Q.⑤ Classification of AI.

OR

Types of AI.

[1] Weak AI :- The study & design of machines that perform intelligent tasks.

- Not concerned with how tasks are performed, mostly concerned with performance & efficiency. Such as solutions that are reasonable for NP-complete problems. E.g., to make a flying machine use logic & physics, don't mimic a bird.

[2] Strong AI :- The study & design of machines that simulate the human mind to perform intelligent tasks.

- Borrow many ideas from psychology, neuroscience. Goal is to perform tasks the way a human might do them - which makes sense, since we do have models of human thought & problem solving.

- Includes ~~AI~~. psychological ideas in STM, LTM, Forgetting, Language, genetics, etc. Assumes that the

by physical Symbol hypothesis holds.

[3] Evolutionary AI :-

- the study & design of machines that simulate simple creatures, and attempt to evolve and have higher level emergent behaviour.

ex. ants, bees, etc.

Q.⑥ What are the Applications of AI ?

- AI has been dominant in various fields such as -

[1] Gaming :

- AI plays vital role in strategic games such as chess, poker, tic-tac-toe, etc. where machine can think of large number of possible positions based on heuristic knowledge.

[2] Natural Language Processing :

- It is possible to interact with the computer that understands natural language spoken by humans.

[3] Expert Systems :

- There are some applications which integreate machine, software, & special information to impair de reasoning and advising. They provi explanation and advice to the users.

[4] Computer Vision Systems:

- These systems understand, inter pret and comprehend visual input on the computer.

[5] Speech Recognition:

- Some intelligent systems are capable of hearing and compre hending the language in terms of sentences and their ~~tongue~~ ges meanings while a human talk to it. It can handle diffe - rent accents, slang words, noise in the background, change in human's noise, etc.

[6] Hand Written Recognition:

- The handwriting recognition Soft ware reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

[7] Intelligent Robots :

- Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump and pressure. They have different efficient ~~sensors~~ processors, multiple sensors & huge memory, to exhibit intelligence.

- In Addition, they are capable of learning from their mistakes and they can adapt to the new environment.

⊛ Chapter: ② : Nature Language Processing ⊛

2.① Introduction About NLP:

- Natural Language Processing (NLP) is a field of Artificial Intelligence and in which computers analyze, understand, and derive meaning from human language.

- The field focuses on communication between computers & humans in natural language and is about making computers understand & generate human language.

- Natural language Processing (NLP) refers to communicating with an intelligent systems using a natural language such as English.

- Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to ~~dear~~ hear decision from dialogue based ~~on~~ clinical expert system, etc.

- By utilizing NLP, developers can organize and structure knowledge to

perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, topic segmentation, etc.

⊛ There are the following two components of NLP:-

1. Natural Language Understanding (NLU):

- Natural language understanding helps the machine to understand & analyze human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

- NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

2. Natural Language Generation (NLG):

- Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation.

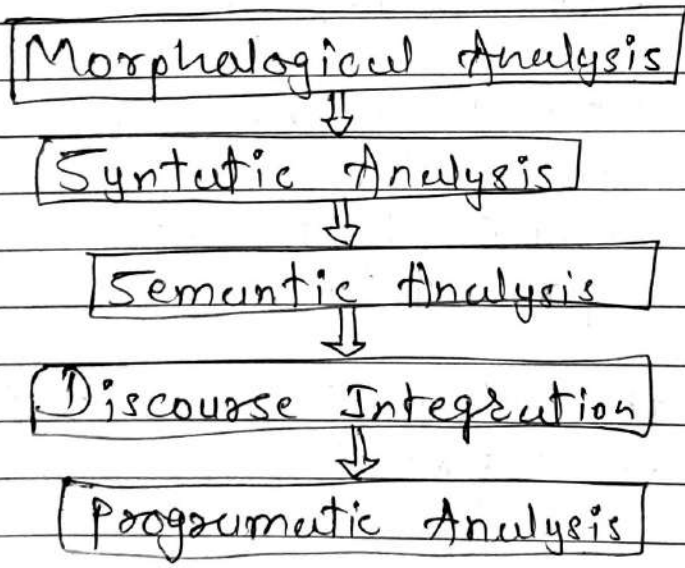
- It mainly involves Text planning, Sentence planning and Text Realization.

Ans ① Natural Language Processing (NLP) problem can be devided into two tasks:

1. Processing written text, using lexical, syntatic & semantic knowledge of the language as well as the required real world information.

2. Processing Spoken language, using all the information needed above plus additional knowledge about phonology as well as enough added information to handle the further ambiguities that arise in speech.

Q.② Steps of/in Natural Language Processing:



[7] Morphological Analysis:

- The morphological level of linguistic processing deals with the study of word structures and word formation, focusing on the analysis of the individual components of words.

- Lexicon of a language means the collection of words and phrases in a language. It involves identifying and analyzing the structure of words

- Morphological analysis is dividing the whole chunk of text into paragraphs, sentences and words.

- Suppose there is a sentences "I went to print Bill's .init file".

- Morphological analysis must do the following things:

  - pull apart the word "Bill's" into proper noun "Bill" and the possessive suffix "'s"

  - Recognize the sequence ".init" as a file extension that is functioning as an - adjective in the sentence.

- This process will usually assign syntactic categories to all the words in the sentences.

Syntactic Analysis:

- Syntactic analysis must exploit the results of morphological analysis to build a structural description of the sentences.

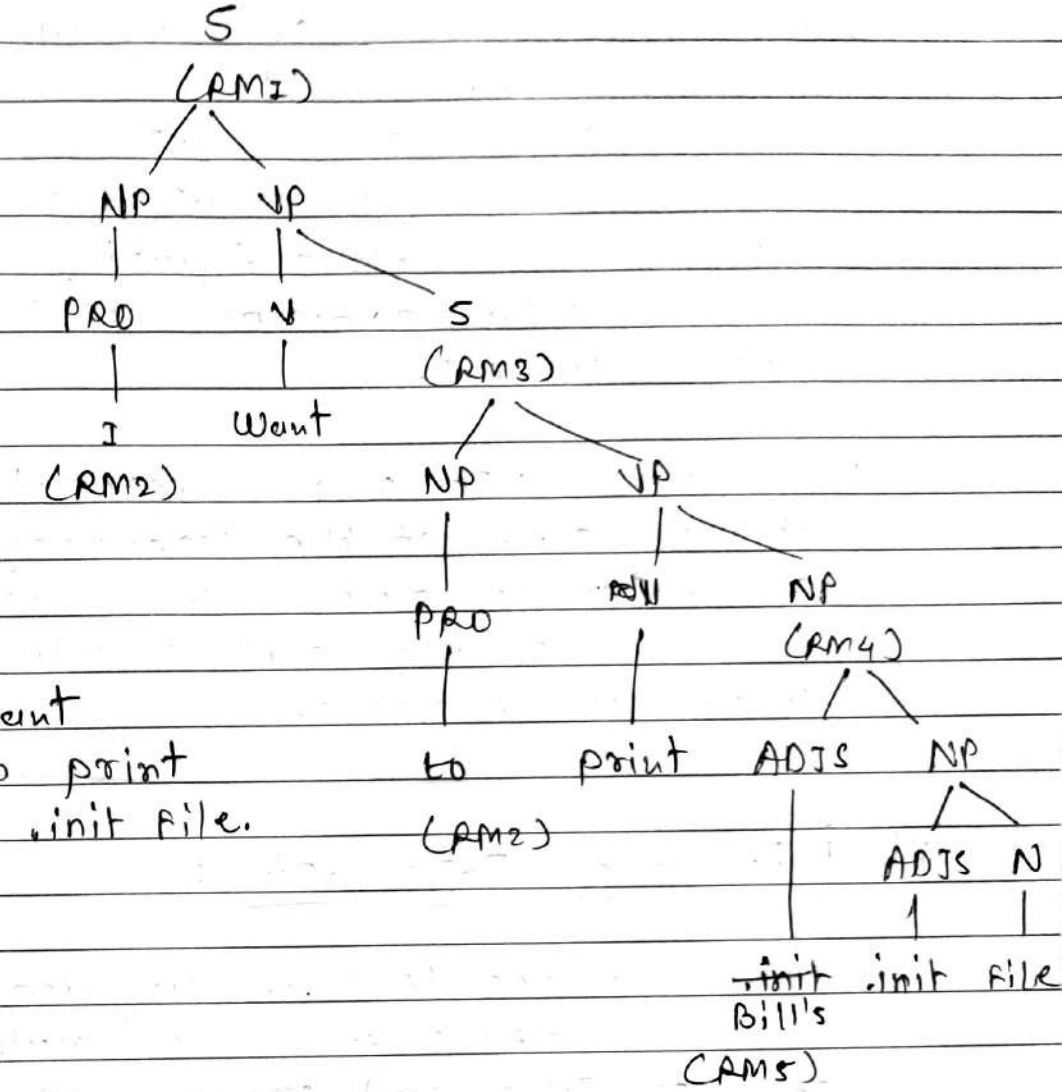
- The goal of this process, called parsing, is to convert the flat list of words that forms the sentence into a structure that defines the units that are represented by that flat list.

- The important thing here is that a flat sentence has been converted into a hierarchical structure & that the structure corresponds to meaning units when semantic analysis is performed.

- Reference markers (set of entities) are shown in the parenthesis in the parse tree.

- Each one corresponds to some entity that has been mentioned in the sentences.

- These reference markers are useful later since they provide a place in which to accumulate information about the entities as we get it.



→ I want to print Bill's .init file.

[3] Semantic Analysis:

- Semantic analysis must do two important things:

a. It must map individual words into appropriate objects in the knowledge base or databases.

b. It must create the correct structures to correspond to the way the meanings of the individual words combine with each other.

- The semantic level of linguistic processing deals with the determi-nation of what a sentence really means by relating syntactic features and disambiguating words with multiple definitions to the given context.

- This level entails the appropiate interpretation of the meaning of sentences, rather than the analysis at the level of individual words or phrases.

[4] Discourse Integration:

- The discource level of linguistic processing deals with the analysis of structure and meaning of text beyond a single sentence, making connections between words and sentences.

- At this level, Anaphora Resolution is also achieved by identifying the entity referenced by an anaphor (most commonly in the form of, but not limited to, a pronoun).

- An example is shown below.

- "I voted for Obama because he

was most aligned with my values" she said.

- With the capability to recognize & resolve anaphora relationships, document & query representations are improved.

- Structured documents also benefit from the analysis at the discourse level since sections can be broken down into (1) title, (2) abstract, (3) introduction, (4) body, (5) results, (6) Analysis, (7) conclusion and (8) references.

---

[5] Pragmatic Analysis:

- The final step toward effective understanding is to decide what to do as a result.

- One possible thing to do is to record what was said as a fact to be done with it.

- For some sentences, whose intended effect is clearly declarative, this is the precisely correct thing to do.

- But for other sentences, including this one, the intended effect is different.

- We can discover this intended effect by applying a set of rules that characterize

Page No.
Date

Cooperative dialogues.

- The final step in pragmatic processing is to translate, from the knowledge based representation to a command to be executed by the system.

- The pragmatic level of linguistic processing deals with the use of real-world knowledge and understanding of how this impacts the meaning of what is being communicated.

- By analyzing the contextual dimension of the document & queries, a more detailed representation is derived.

Q. ③ Application of NLP:

- Sentiment Analysis
- Text classification
- chatBots and Virtual Assistant
- Information extraction
- Machine Translation
- Text Summarization
- Auto-correct
- Speech Recognition

[i] Sentiment Analysis:

- Sentiment Analysis is also known as opinion mining. It is used on the web to analyze the attitude, behaviour, and emotional state of the sender.

- This application is implemented through a combination of NLP and statistics by assigning the values to the text (positive, negative or natural), identify the mood of the context (happy, sad, etc).

- Beyond determining simple polarity, Sentiment analysis understands sentiment in context to help better understand what's behind an expressed opinion, which can be extremely relevant in understanding and driving purchasing decisions.

[2] Text classification:

- Text classification is the process of categorizing the text into a group of words.

- By using NLP, text classification can automatically analyze text & then assign a set of predefined tags or categories based on its context.

- For eg., Spam Detection is used to detect unwanted e-mails getting to a user's inbox.

[3] Chat Bots & Virtual Assistants:

- Implementing the chat Bot is one of the important application of NLP. It is used by many companies to provide the customer's chat services.

- A virtual assistant is a software that uses speech recognition, natural language understanding, and natural language processing to understand the verbal commands of a user & perform actions accordingly.

[4] Information extraction:

- Information extraction is one of the most important applications of NLP.

- It is used for extracting essential information from understand or semi-structured machine-readable documents.

[5] Machine Translation:

- Machine translation is used to translate or text or speech from one natural language to another natural language while keeping the meaning intact.

- For eg. Google Translate can easily convert text from one language to another language. these tools are helping numerous people and businesses in breaking the language barrier and becoming successful.

[5] Text Summarization:

- Summerization is the task of condensing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements & the meaning of content.

- There are important applications for text Summarization in various NLP related task such as text classification, question answering, legal texts summarization, news summarization and headline generation.

[7] Auto-Correct:

- Microsoft corporation provides word processor softwares like Ms-word, power point for the spelling correction.

- Tools like Grammarly provide so many features in helping a person write better content. It is one of the most widely used applications of NLP that helps professionals in all job domains create better content.

[8] Speech Recognition:

- From smart home devices and appliances that take instructions, and can be switched on & off remotely, digital assistants that can set reminders, schedule meetings, play songs, to search engines that respond with relevant search results to user queries, speech recognition has become an indispensable part of our lives.

Q. (4) Spell checking in NLP.

- Spell check is a process of detecting and sometimes providing suggestions for incorrectly spelled words in text.

- In Computing, spell checker is an application program that flags words in a document that may not be spelled correctly.

- Spell checker may be stand-alone capable of operating on a block a text such as word-processor, electronic dictionary.

- A basic spell checker carries out the following sentences:

- it can scans the text and extracts the words contained in it.

- it then compares each word with a known list of correctly spelled words.

- An additional step is a language-dependent algorithm for handling morphology.

(4) Spelling errors can be divided as:

(1) Non-word errors:

- these errors are mostly common type of errors. you either miss a few keystrokes or let your fingers hurtle a bit longer. these are those errors words that cannot be found in the dictionary. this words are complex to provide the suggestion, so this might not be suggested.

(2) Real-word errors:

- sometimes instead of creating a non-word, you end up creating a real word, but one you didn't intend to do so. Eg. typing flower when you meant flour. these are those words that are acceptable error words in the dictionary

(3) Cognitive Errors:

- The previous two types of errors result not from ignorance of a word or its correct spelling. Cognitive errors can occur due to those factors. The words piece & peice are homophones (sound the same). So you are not sure which one is which. Sometimes your damn sure about your spellings.

(4) Short forms / Slang / Lingo:

- These are possibly not even spelling errors. You are trying hard to fit in everything within a text message or a tweet.

(*) Error Detection:

- Dictionary Lookup Technique:

- In this, Dictionary Lookup technique is used which checks every word of input text for its presence in dictionary. If that word present in the dictionary, then it is a correct word. Otherwise it is put into the list of error words.

* * *

Page No.
Date

Chapter: (4): Symbolic Reasoning Under Uncertainty

Q.(1) What is Reasoning :-

- the Reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts & beliefs.

- it is a general process of thinking rationally, to find valid conclusions.

Types of Reasoning :-

(1) Deductive Reasoning
(2) Inductive Reasoning
(3) Abductive Reasoning
(4) Common Sense Reasoning
(5) Monotonic Reasoning
(6) Non-monotonic Reasoning

Q.(2) Advantages & Disadvantages of N.M.R.

Advantages :-

- For real-world systems such as Robot navigation, we can use N.M.R.

- In N.M.R, we can choose probabilistic facts or can make assumptions.

Page No.
Date

⑥ Disadvantages :-

- In NMR, the old facts may be invalidated by adding new sentences.
- It can't be used for theorem proving.

Ex of NMR :-

- Suppose the knowledge base contains the following knowledge:

(1) Birds can fly
(2) Penguins cannot fly
(3) Pritty is a bird

- So, from the above sentences, we can conclude that pritty can fly.
- However, if we add one another sentence into knowledge base "pritty is a Penguine", which concludes "pritty can't fly", so it invalidates above conclusion.

⊛ Chapter: ⑤ : Probabilistic Reasoning ⊛

Page No.
Date

Q. ① What is Certainty Factor ?

- The Certainty Factor (CF) is a numeric value which tell us about how likely an event or a statement is supposed to be true.

- it is somewhat similar to what we define in probability, but the difference in it is that an agent after finding the probability of any event to occur can't decide what to do.

- Based on the probability and other knowledge that the agent has, this certainty factor is decided through which the agent can decide whether to declare the statement true or false.

- The value of the certainty factor lies between -1.0 to 1.0,

where, the negative 1.0 value suggests that the statement can never be true in any situation, and the positive 1.0 value defines that the statement can never be false.

Q.2 Applications of Bayesian Network.

(1) Gene Regulator Network
(2) Medicine
(3) Biomonitoring
(4) Document classification
(5) Information retrieval
(6) Semantic search
(7) Image Processing
(8) Spam filter
(9) Turbo code
(10) System Biology

Q.3 (a) Ignorance using Dempster-Shafer theory :-

- The Dempster-Shafer theory is designed to deal with the distinction between uncertainty and ignorance.

- Rather than computing the probability of a proposition, it computes probability that the evidence supports the propositions.

- This measure of belief is called a belief function, written Bel(X).

- We will return to coin fliping example of belief functions.

Page No.
Date

Eg. Suppose a shady character comes up to you and offers to bet you $1 that this coin will come up heads on the next flip.

- Given that coin might or might not be fair, what belief should you ascribe to the event that it comes up heads?

- Dempster-Shafer theory says that because you have no evidence either way, you have to say that the Belief: Bel(Heads) = 0 and also that Bel(¬Heads) = 0.

- This makes Dempster-Shafer reasoning system skeptical in a way that has some intuitive appeal.

- Now suppose you have an expert at your disposal who testifies with 90% certainty that the coin is fair (i.e he is 90% sure that P(Heads) = 0.5).

- Then Dempster-Shafer theory gives Bel(Heads) = 0.9 × 0.5 = 0.45. and likewise Bel(¬Heads) = 0.45. There is still a 10 percentage point "gap" that is accounted for by the evidence.

- Dempster-Shafer' shows how to combine evidence to give new values to Bel, and Shafer's work extends this into a

Complete computational model.

Q. (4) Rule-Based Systems :-

- A rule-based system in AI is a system that applies human-made rules to store, sort and manipulate data.

- Rule based system in AI requires a set of facts or source data, and a set of rules for manipulating that data.

- These rules are sometimes refered to as "IF statements" as they tend to follow the line of "IF X happens THEN do Y".

Ex Trouble shooting of water pumps

If pump failure then the pressure is low.

If pump failure then check oil level

If pump failure then pump failure.

- Rule based system consists of a library of such rules.

- Rules reflect essential relationship within the domain.

- Rules reflects ways to reason about the domain.

- Rules draw conclusions and points to actions, when specific information about the domain comes in. This is called inference.

- The inference is a kind of chain reaction like:

If there is a power failure then (see rules 1, 2, 3 mentioned above)
Rule 3 states that: there is pump failure, and

Rule 1 tells that the pressure is low, & Rule 2 gives a (useless) recommendation to check the oil level.

Q.(5) Fuzzy sets & Fuzzy logic:-

(*) Fuzzy sets:-

- Fuzzy set theory is used for specifying how well an object satisfies a fuzzy description.

Page No.
Date

Chapter 6: Fuzzy Playing

For Example:

- Consider the proposition "Anil is tall". Is this true, if Anil is 5'10"? most people would hesitate to answer "true" or "false", preferring to say, "sort of".
- Note that this is not a question of uncertainty about external world. we are sure of Anil's height. The issue is that the linguistic term "tall" does not refer to a sharp demarcation of objects into two classes and there are degrees of tallness.

= Due to this reason, fuzzy set theory is not a method for uncertain reasoning at all. Rather, fuzzy set theory treats all a fuzzy predicate and says that the truth value of Tall (Anil) is a number between 0 & 1, rather than just be true or false.

- The name "fuzzy set" derives from the interpretation of the predicate as implicitly defining a set of its members - a set that does not have sharp boundries.

Page No.
Date

⊛ Fuzzy Logic :-

- Fuzzy logic is a method for Reasoning with logical expressions describing membership in fuzzy sets.

For example :

- The complex sentence Tall(Anil) ∧ Heavy(Anil) has a fuzzy truth value that is a function of the truth (values) of its components.

- The standard rules for evaluating the fuzzy truth, T, of a complex sentence are

$$T(A \wedge B) = \min(T(A), T(B))$$
$$T(A \vee B) = \max(T(A), T(B))$$
$$T(\neg A) = 1 - T(A)$$

- Fuzzy logic is therefore a truth-functional system - a fact that causes serious difficulties.

- For example :

Suppose that $T(\text{Tall}(\text{Anil})) = 0.6$ and $T(\text{Heavy}(\text{Anil})) = 0.4$.

- Then we have $T(\text{Tall}(\text{Anil}) \wedge \text{Heavy}(\text{Anil})) = 0.4$. which seems reasonable but we

also
get the result $T(\text{Full}(\text{Anil}) \wedge \neg(\text{Full}(\text{Ani}))) = 0.4$ which does not.

- The problem arises from the inability of a truth-functional approach to take into account the correlations or anti-correlations among the component propositions.

Page No.

⊛ Chapter: ⑥ : Game Playing ⊛

6.① What is game :-

- A game has at least two players. Solitaire is not considered a game by game theory.

- The term "solitaire" is used for single-player games of concentration.

- An instance of a game begins with a player choosing from a set of specified alternatives. This choice is called a move.

- The moves made by a player may or may not be known to other players.

  - Games in which all moves of all players are known to everyone are called games of perfect information.

- Every instance of the game must end.

- When an instance of a game ends, each player receives a payoff.

- A payoff is a value associated with each player's final situation. A zero-sum game is one, in which elements of payoff matrix sum to zero.

In typical zero-sum game:

(1) Win = 1 point.
(2) Draw = 0 point.
(3) Loss = -1 point.

Q.(2) Minimax Algorithm :-

- mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory.

- it provides an optimal move for the player assuming that opponent is also playing optimally.

- mini-max algorithm uses recursion to search through the game-tree.

- mini-max algorithm is mostly used for game playing in AI. such as chess, checkers, tic-tac-toe, go, and various two-players game.

- This algorithm computes the minimax decision for the current state.

- In this algorithm two players play the game, one is called MAX & other is called MIN.

- Both the players fight if as the opponent player gets the minimum benefit while they get the maximum benefit.

- Both players of the game are opponent of each other, where MAX will select the maximized value & min will select the minimized value.

- The mini-max algorithm performs a depth-first-search algorithm for the exploration of the complete game tree.

- The mini-max algorithm proceeds all the way down to the terminal of the tree, then backtrack the tree as the recursion.

⑧ mini-max Algorithm :-

(1) The start node is MAX (player 1) node with current board configuration.

(2) Expand nodes down (play) to some depth of look-ahead in the game.

(3) Apply evaluation function at each of the leaf nodes.

(4) "Back up" values for each non-leaf nodes until computed for the root node.

(5) At MIN (player 2) nodes, the backed up value is the minimum of the values associated with its children.

(6) At MAX nodes, the backed up value is the maximum of the values associated with its children.

⊛ Working of mini-max Algorithm :-

- given is a example of mini-max Algorithm in which there are two players one is called Maximizer and other is called minimizer.

- Maximizer will try to get the maximum possible score, and minimizer will try to get the minimum possible score.

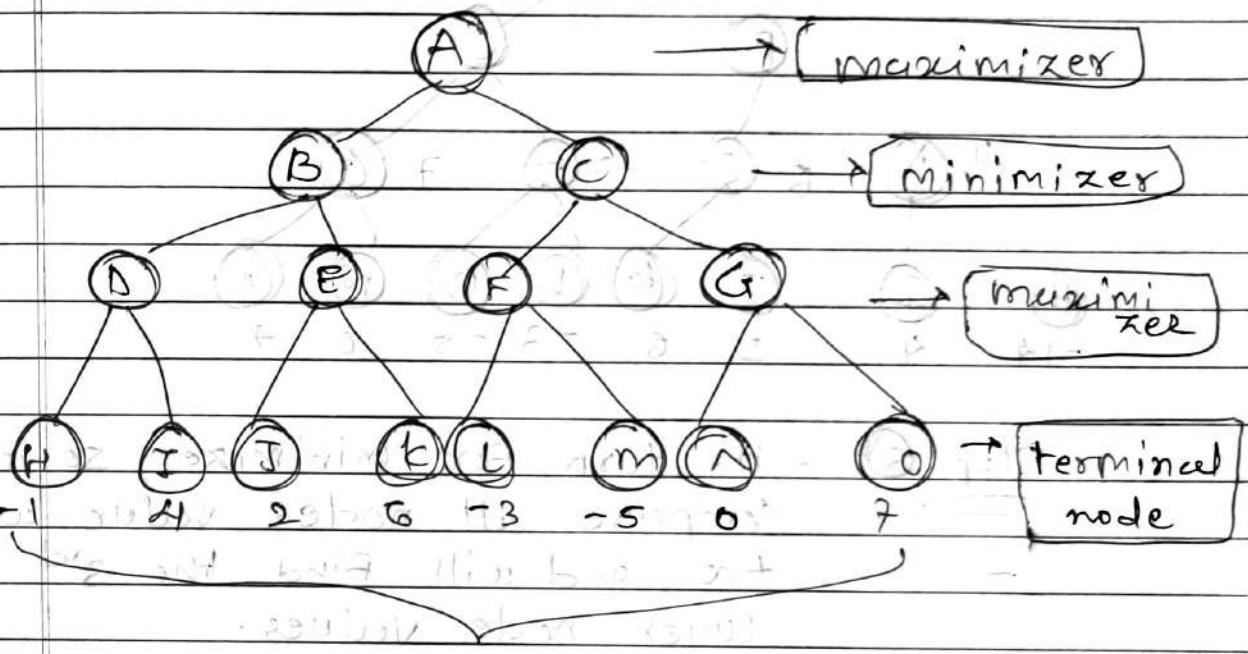
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.

- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs.

- Following are the main steps involved in solving the two-player game tree:

Step: (1)

- First step, the algorithm generates the entire game-tree & apply utility function to get the utility values for the terminal nodes.

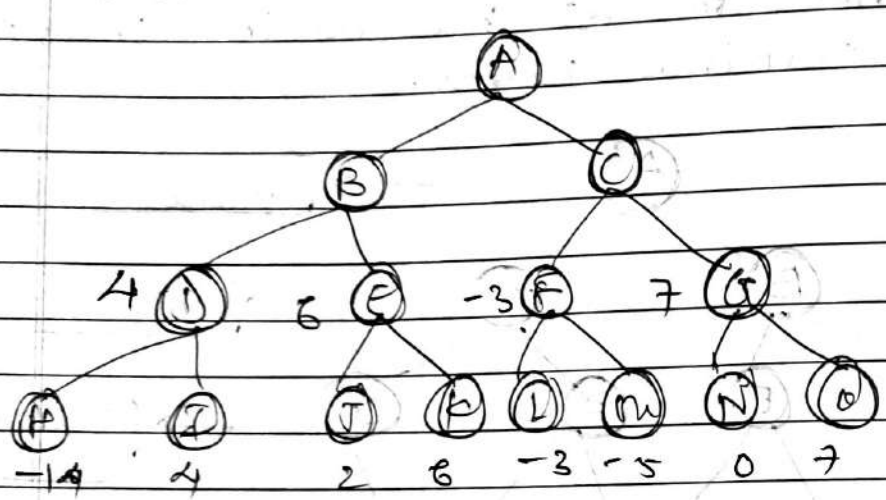


Step: (2)

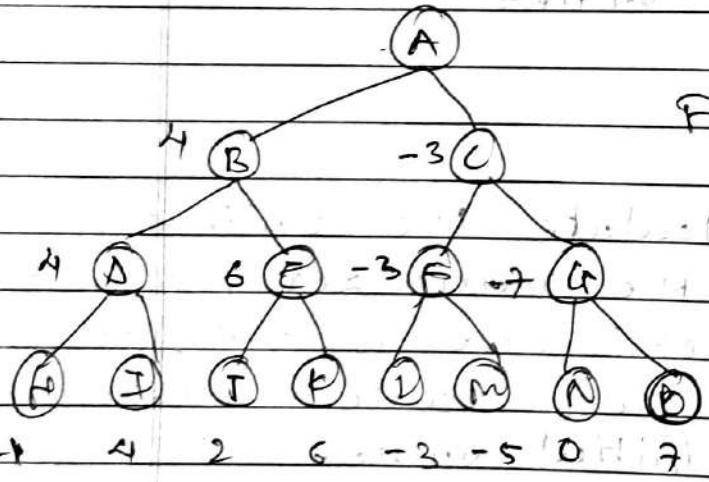
- Now we calculate the utilities ~~to~~ value for the maximizer, its initial values in worst-case is $-\infty$, so we will compare each value in terminal state with initial value.

- Then determines the Higher nodes values. it will find maximum among all the all.

For node D, max(-1, -∞) → max(-1, 4) → 4
For node E, max(2, -∞) → max(2, 6) → 6
For node F, max(-3, -∞) → max(-3, -5) → -3
For node G, max(0, -∞) → max(0, 7) → 7



Step: (3) - it's turn for Minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.



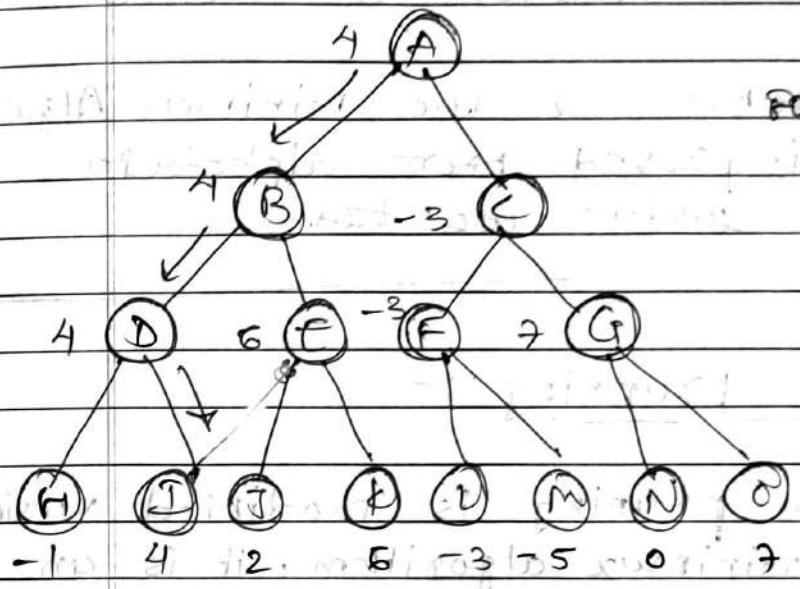
For node B = min(4, 6)
= 4

For node C = min(-3, 7)
= -3

Step: (4) Now, it's a turn for Maximizer, and it will again choose the maximum of all nodes values

Page No.
Date

Find the maximum value for the root node.



For node A,
max(4, -3) = 4

⊛ Properties of mini-max Algorithm :-

- Complete - Always gives a solution
- optimal - it is an optimal Algo
- time complexity - $O(b^d)$
  where, b = branching factor & d = depth of the tree.
- Space complexity - $O(bd)$.

⊛ Limitation / Drawback of mini-max Algorithm ⊛

- The main drawback of the minimax algorithm is that it gets really slow for complex games such as chess, go, etc.

- This type of games has a huge branching factor, and the player has lots of choices to decide.

- This limitation of the minimax Algo. can be improved from alphabeta pruning. ~~which we will~~

Q. (3) Alpha-Beta Pruning :-

- Alpha-Beta pruning is modified version of the minimax algorithm. it is an Optimization technique for minimax Algo.

- in minimax Algorithm we can not eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning.

- This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. it is also called as Alpha-Beta Algorithm

- Alpha-Beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

- The two-parameter can be defined as:

(1) Alpha :-

- The best (highest-value) choice we have found so far at any point along the path of maximizer.

- The initial value of alpha is $-\infty$.

(2) Beta :-

- The best (lowest value) choice we have found so far at any point along the path of Minimizer.

- The initial value of beta is $+\infty$.

(*) Condition of Alpha-Beta Pruning :-

- The main condition of Alpha-Beta pruning is:

$$\alpha \geq \beta$$

(*) key-points of Alpha-Beta Pruning :-

- The max player will only update the value of alpha.

- The Min player will only update the value of Beta.

- while backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.

- we will only pass the alpha, beta values to the child nodes.

(*) Algorithm of Alpha-Beta Pruning :-

/*
alpha is the best score for max along the path to state.
beta is the best score for min along the path to state.
*/

if the level is the top level, let alpha = -infinity; beta = +infinity.

if depth has reached the secred limit apply static evaluation function to state and return result.

(*) If player max :

Until all of state's children are examined with ALPHA-BETA or until alpha is equal to or greater than beta :

Page No.
Date

Call ALPHA-BETA (child, min, depth+1, alpha, beta);
note result
Compare the value reported with alpha; if reported value is larger reset alpha to the new value.
Report alpha

If player is min:

Until all of state's children are examined with ALPHA-BETA or until alpha is equal to or greater than beta:

Call ALPHA-BETA (child, max, depth+1, alpha, beta);
note result
Compare the value reported with the beta; if reported value is smaller, reset beta to the new value.
Report beta.

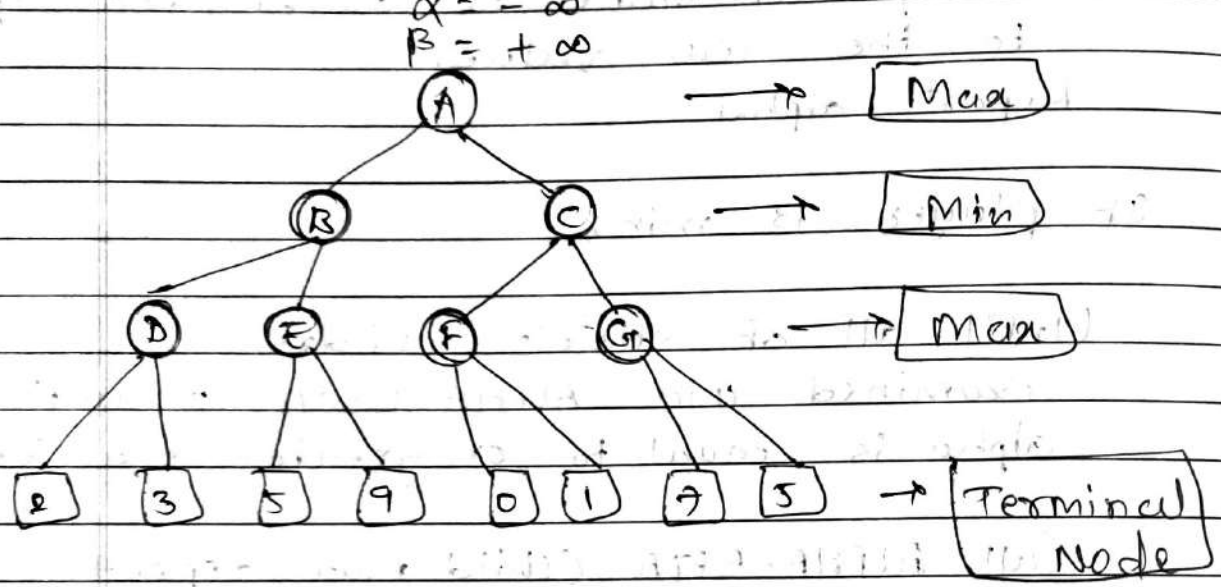
(*) Working of Alpha-Beta Pruning :-

- let us take an example of two-player search tree to understand working of Alpha-Beta Pruning.

Step: (1) At first step the MAX player will start first move from node A where $\alpha = -\infty$ & $\beta = +\infty$, these

Page No.
Date

Value of alpha and beta passed down to node B where again $\alpha = -\infty$ & $\beta = +\infty$, & Node B passes the same value to its child D.



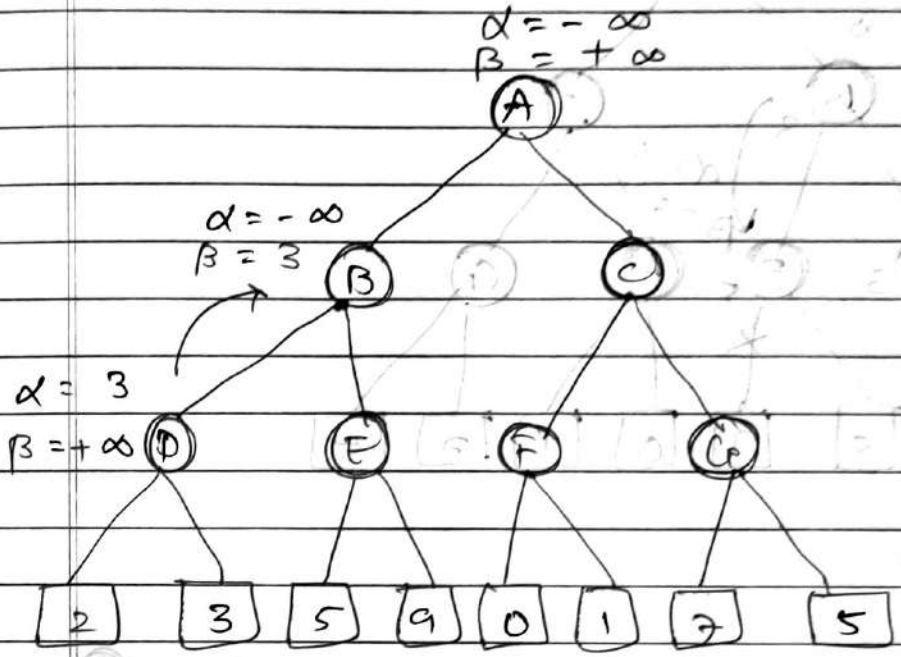
Step: ②

- At node D, the value of α will be calculated as its turn for Max.
- The value of α is compared with firstly 2 and then 3, and max(2,3) = 3 will be the value of α at node D and node value will also 3.

Step: ③ Now,

- Algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now, $\beta = +\infty$, will compare with the available subsequent nodes value, i.e min(∞, 3) = 3, hence at

node B now $\alpha = -\infty$, & $\beta = 3$.

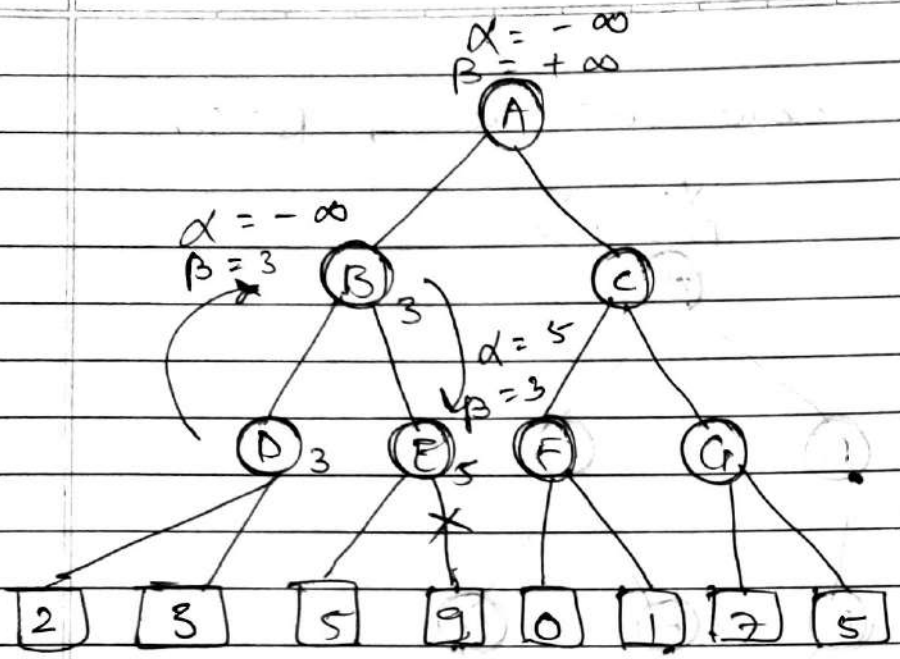


- in the next step, algorithm traverse the next successor Node B which is node E, and the values of $\alpha = -\infty$, & $\beta = 3$; will also be passed.

Step: (4)

- At node E, Max will take its turn, and the value of alpha will change.

- the current value of alpha will be compared with 5, so max $(-\infty, 5) = 5$, hence at node E $\alpha = 5$ & $\beta = 3$, where $\alpha \geq \beta$; so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



Step: ⑤

- C, α = 3 & β = +∞

Step: ⑥

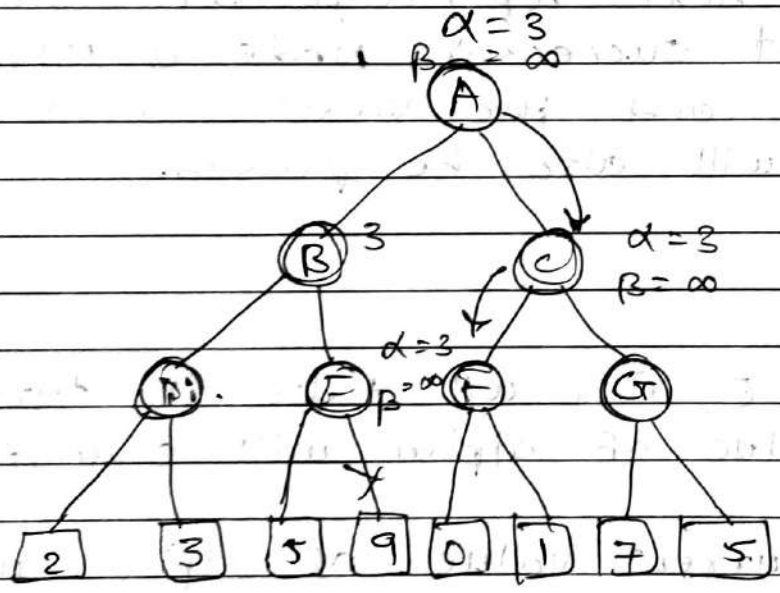
At node F,

1st max(3, 0) = 3

Then,

max(3, 1) = 3,

but F will become 7.



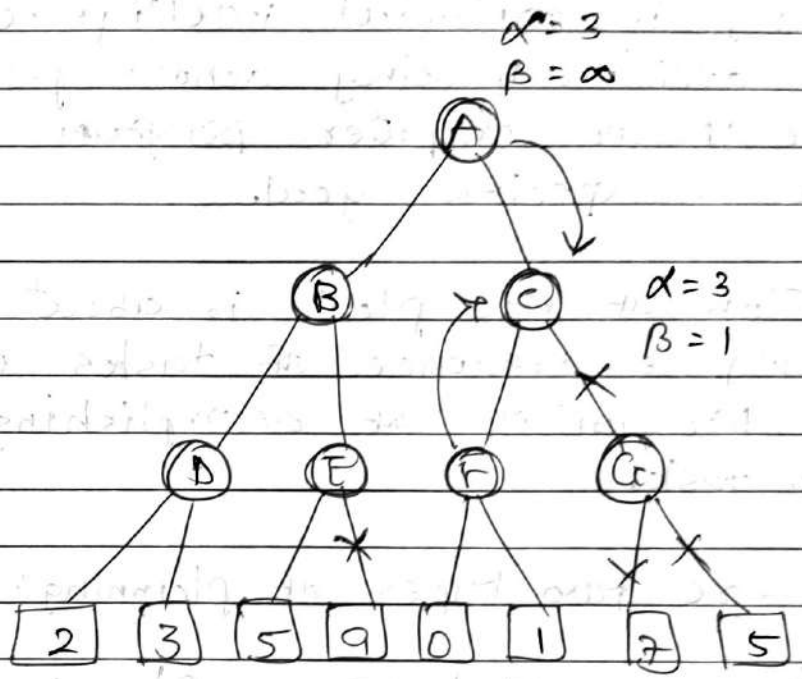
Step: ⑦ F return to node value 7 to node C, at (α = 3 & β = +∞

- here value of beta will be changed, it will compare with 1 so min(∞, 1) = 1,

Now, at C, α = 3 & β = 1. again it satisfies the condition α ≥ β, so the next child of C

Page No.
Date

which is Cr will be pruned, and the Algo will not compute the entries sub-tree Cr.



⑧ chapter : ⑦ : Planning ⑧

Page No.
Date

Q.① What is planning in AI?

- planning in artificial intelligence is about decision-making actions performed by robots or computer programs to achieve a specific goal.

- Execution of the plan is about choosing a sequence of tasks with a high probability of accomplishing a specific task.

- there are two types of planning:

  (1) Forward State Space Planning (FSSP)
  (2) Backward State Space Planning (BSSP)

Q.② Explain Planning Problem.

OR

Explain Block-world Planning problem.

- The block-world problem is known as the Sussman anomaly.

- The non-interlaced planners of the early 1970s were unable to solve this problem. Therefore it is considered odd.

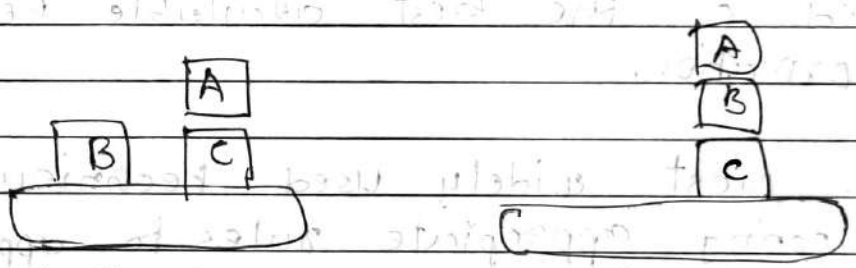
- When two sub-goals, G1 & G2 are given, a non-interleaved planner either produces a plan for G1 that is combined

with a plan G2 or vice versa.

- in the block-world problem, three blocks labeled 'A', 'B', and 'C' are allowed to rest on a flat surface.

- the given condition is that only one block can be moved at a time to achieve the target.

- the start position and target position are shown in the following diagram.



start state goal state

word planning problem.

Q.② Components of the planning system.

(1) choose the best rule to apply next, based on the best available heuristic information.

(2) Apply the choosen rule to compute the new problem state that arises from its application.

(3) Detect when a solution has been found.

(4) Detect dead ends so that they can be abandoned and the system's effort directed in more fruitful directions.

(5) Detect when an almost correct solution has been found and employ special techniques to make it totally correct.

---

[1] Choose the best rule to apply next, based on the best available heuristic information.

- The most widely used technique for selecting appropriate rules to apply is first to isolate a set of differences between desired goal state and then to identify those rules that are relevant to reduce those differences.

- if there are several rules, a variety of other heuristic information can be exploited to choose among them.

---

[2] Apply the chosen rule to compute the new problem state that arises from its application.

- in simple systems, applying rules is easy. Each rule simply specifies the problem state that would result from its application.

- In complex systems, we must be able to deal with rules that specify only a small part of the complete problem state.

[3] Detect when a solution has been found.

- A planning system has succeeded in finding a solution to a problem when it has found a sequence of operators that transforms the initial problem state into the goal state.

- How will it know when this has been done?

- in simple problem-solving systems, this question is easily answered by a straight forward match to the state descriptions.

- One of the representative systems for planning systems is predicate Logic. ~~Suppose that as a part of our goal, we have the pre.~~

[4] Detect dead ends so that they can be abandoned and the system's effort directed in more fruitful directions.

- As a planning system is searching for a sequence of operators to solve a particular problem, it must be able to detect when it is exploring a path that can never lead to a solution.

- The same reasoning mechanism that can be used to detect a solution can often be used for detecting a dead end.

---

[5] Detect when an almost correct solution has been found and employ special techniques to make it totally correct.

- The kind of techniques discussed are often useful in solving nearly decomposable problems.

Q.③ Non-linear Planning using Constraint Posting:

- Difficult problems cause goal interactions.

- The operators used to solve one sub-problem may interfere with the solution to a previous sub-problem.

- most problems require an interwined plan in which multiple sub-problems are worked on simultaneously.

- Such a plan is called nonlinear plan because it is not composed of a linear sequence of complete sub-plans.

(4) Constraint Posting :-

- The idea of constraint posting is to build up a plan by incrementally hypothesizing operators, partial orderings between operators, and binding of variables within operators.

- At any given time in the problem-solving process, we may have a set of useful operators but perhaps no clear idea of how those operators should be ordered with respect to each other.

- A solution is a partially ordered, partially instantiated set of operators to generate an actual plan, and we convert the partial order into any number of total orders.

* * *

Page No.
Date

⊛ chapter:

Search ⊛

⊛ Constraint Posting Vs State Space

⊛ state space search :-

- Moves in the space: modify world state via operator
- Model of time: Depth of node in search space
- Plan stored in: Series of state transition

⊛ Constraint Posting Search :-

- Moves in the space: Add operators, Order operators, Bind variables.
- Model of time: partially ordered set of operators
- Plan stored in: single node.

⊛ ① Hierarchical Planning :-

- In order to solve hard problems, a problem solver may have to generate long plans.
- it is important to be able to eliminate some of the details of the problem until a solution that addresses the main issues is found.

- then an attempt can be made to fill in the appropriate details.

- Early attempts to do this involved the use of macro operators, in which larger operators were built from smaller ones.

- in this approach, no details were eliminated from actual descriptions of the operators.

Q.⑤ Goal Stack Planning :-

- Goal stack planning is one of the earliest methods in Artificial intelligence in which we work backwards from the goal state to the initial state.

- We stack the goal state and we fulfiling the preconditions required to achieve the initial state.

- these preconditions in turn have thier own set of preconditions, which are required to be satisfied first.

- we keep solving these "goals" and "sub-goals" until we finally arrive at the initial states.

- we make use of a stack to hold these goals that need to be ~~satisfied~~ fulfilled as

well the actions that we need to perform for the same.

- Apart from the "Initial State" and the "Goal state", we maintain a "World State" configuration as well.

- Goal stack uses this world state to work its way from Goal State to Initial state.

- World State on the other hand starts off as the Initial state and ends up being transformed into the Goal state.

- At the end of this algorithm we are left with an empty stack and a set of actions which helps us navigate from the initial state to the world state.

⊛ Chapter : (10) : Expert Systems ⊛

Q.(1) What is expert system?

- The expert system is a part of Artificial intelligence, and it solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base.

- The system helps in decision making for complex problems using both facts and heuristics like a human expert. These it is called so because it contains the expert knowledge of a specific domain and can solve any complex problems of that particular domain. These systems are designed for a specific domain, such as medicine, science, etc.

- The performance of an expert system is based on the expert's knowledge stored in knowledge base. The more knowledge stored in knowledge base, the more system improves its performance.

⊛ Some popular examples of the Expert System :-

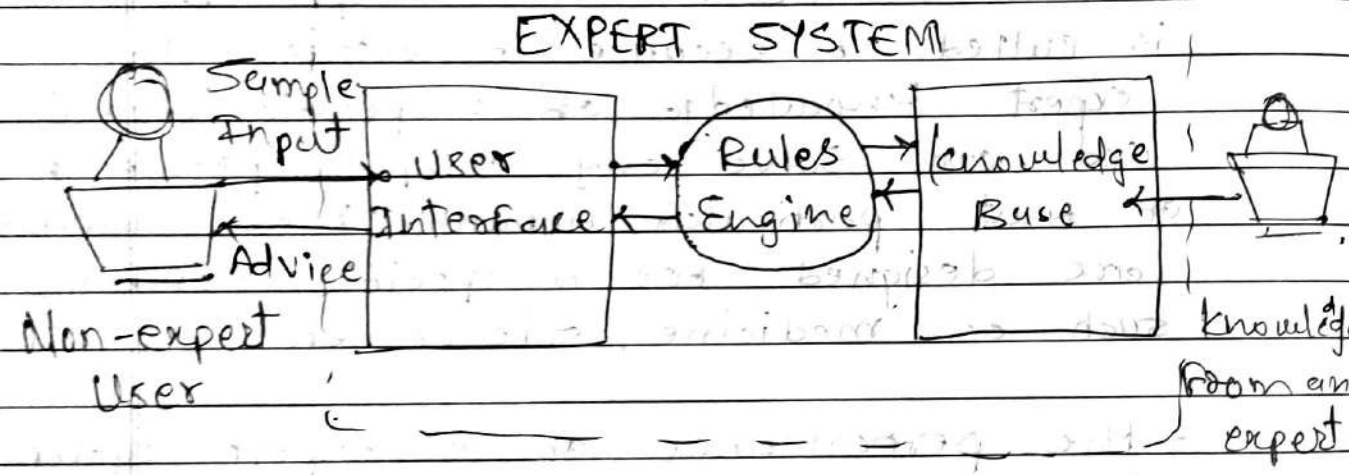
(1) DENDRAL
(2) MYCIN
(3) PXDES
(4) eaDeT

Page No.
Date

⑥ characteristics of Expert System :-

(1) High Performance
(2) Understandable
(3) Reliable
(4) Highly responsive

⑦ working of the Expert System :-



⑧ Architecture / Components of the Expert System.

- An expert system consists of three components :

(1) User Interface
(2) Inference Engine
(3) knowledge Base

Page No.
Date

④ Diagram :-

[1] User Interface :-

- With the help of user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine.

- After going getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

[2] Inference Engine (Rules of engine) :-

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system.

- It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps to in deriving an error-free solution of queries by the user.

- With the help of an inference engine, the system extracts the knowledge from the knowledge base.

Page No.
Date

There are two types of Inference Engine:

(1) Deterministic Inference Eingine
(2) Probabilistic Inference Engine

(1) Deterministic Inference Engine :-

- the conclusion drawn from this type of a inference engine are assumed to be true.

- it is based on facts and rules.

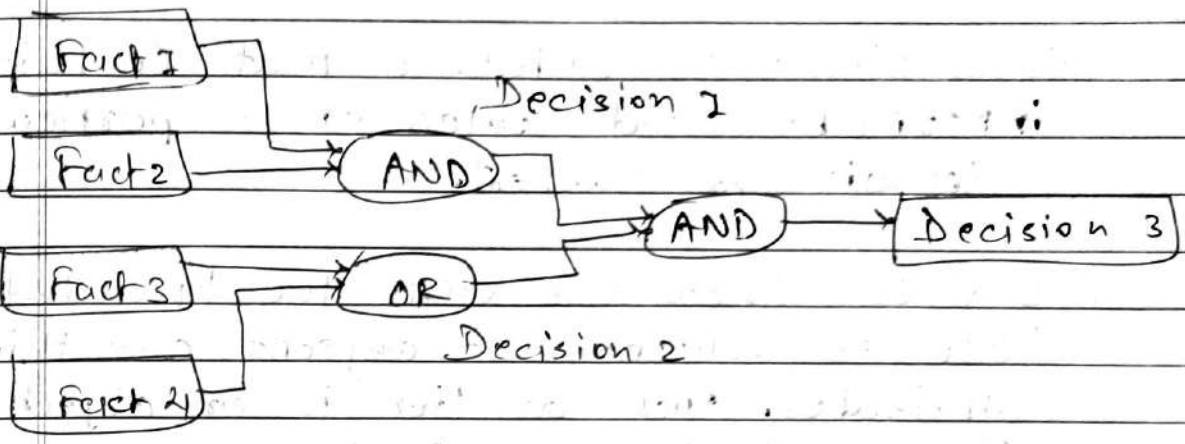
(2) Probabilistic Inference Engine :-

- This type of inference engine contains Uncertainty in conclusions, and based on the probability.

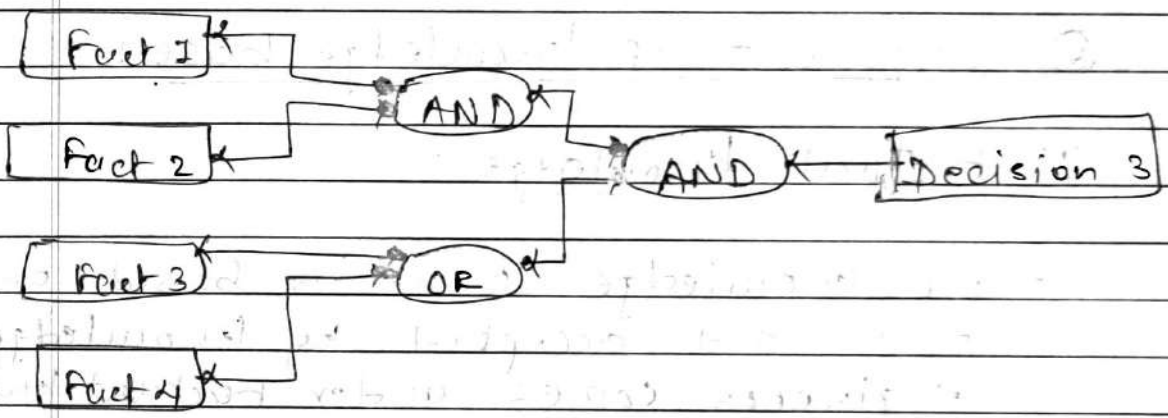
- Inference Engine uses the below modes to derive the solutions:

(1) Forward chaining : it starts from the known facts and rules, and applies the inference rules to add thier conclusion to the known facts.

(2) Backward chaining : it is base backward reasoning method that starts from the goal and works backward to prove the known facts.



⊛ Forward chaining ⊛



⊛ Backward chaining ⊛

[3] Knowledge Base :-

- The knowledge base is a type of storage that stores knowledge acquired from the different experts of the particular domain.
- it is considered as big storage of knowledge. the more knowledge base, the more precise will be the expert system.

- it is similar to database that contains information and rules of a particular domain or subject.

- One can also view the knowledge base as collection of objects and their attributes. Such as lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

④ Components of knowledge base :-

(1) Factual knowledge :-

- The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

(2) Heuristic knowledge :-

- This knowledge is based on practice, the ability to guess, evaluation and experiences.

⑤ Knowledge Representation :-

- it is used to formalize the knowledge stored in the knowledge base using the if-else rules.

Page No.
Date

⊛ Knowledge Acquisitions :-

- it is the process of extracting, organizing and structuring the domain knowledge. Specifying the rules to acquire the knowledge from various experts and store knowledge into the knowledge base.

Q. ③ Explain Capabilities / Characteristics, Limitations & Applications of Expert system.

⊛ Characteristics / Capabilities of ES :-

(1) Advising
(2) Provide decision-making capabilities
(3) Demonstrate a device
(4) Problem solving
(5) Explaining a problem
(6) Interpreting the input
(7) Predicting results
(8) Diagnosis

⊛ Limitations of Expert System :-

- the response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.

- its meintainance and development costs are very high.

- knowledge acquisition for designing is much difficult.

- For each domain, we require a specific ES, which is one of the big limitations.

- it cannot learn from itself and hence requires manual updates

④ Applications of Expert System :-

(1) In designing and Manufacturing domain
(2) In the knowledge domain
(3) In the finance domain
(4) In the diagnosis and troubleshooting of device
(5) Planning and scheduling

Q. ⑪ Explain Expert System shell.

- An Expert system shell is a software development environment.

- It contains the basic components of expert systems. A shell is associated with a prescribed method for building applications by configuring and instantiating these components.

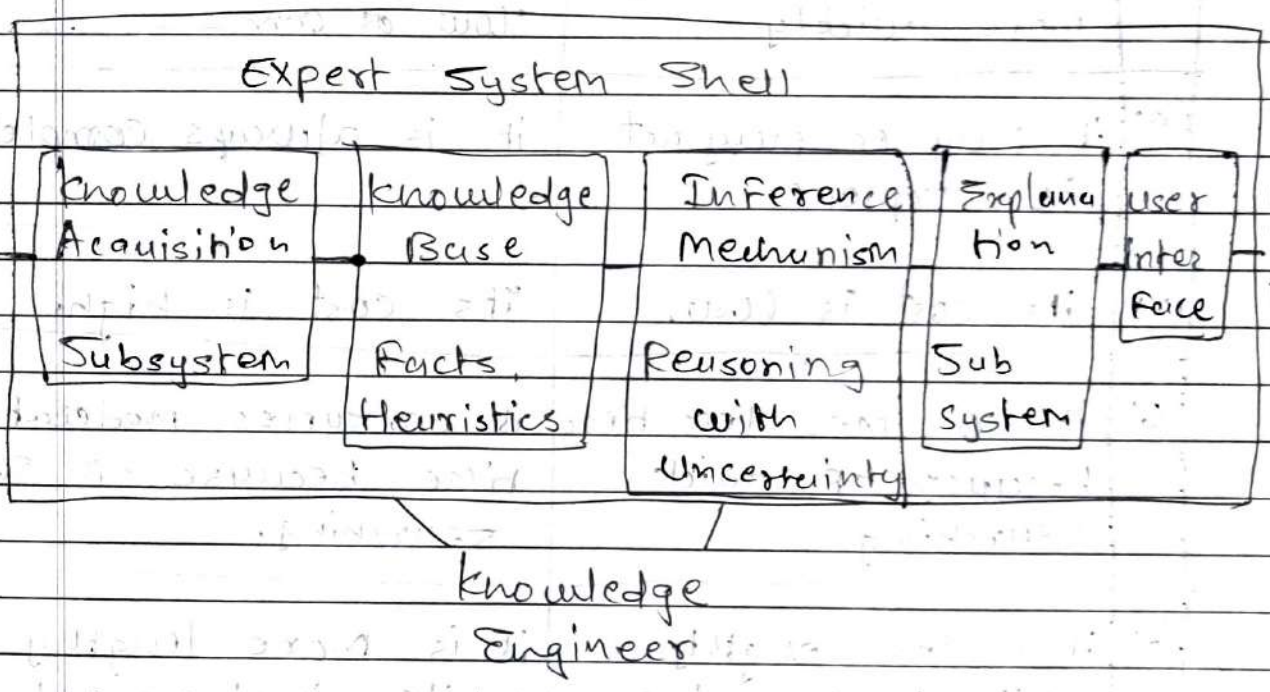
Page No.
Date

④ Shell Components and description :-

- The generic components of a shell:

1. The knowledge acquisition
2. The knowledge Base
3. The reasoning
4. The explanation &
5. The User Interface

- The knowledge base and reasoning engine are the core components.



(4) ~~knowledge~~

Problems, State, Space Search and Heuristic Search Techniques

Chapter: (2): Heuristic search Techniques

Q.(1) Differentiate Uninformed & Informed search.

No.	Informed Search	Uninformed Search
(1)	it is also known as Heuristic search.	it is also known as Blind search.
(2)	it uses knowledge for searching process	it doesn't use knowledge for the searching process.
(3)	it finds the solution more quickly	it finds the solution slow as com—
(4)	it may or may not be complete.	it is always complete.
(5)	its cost is low.	its cost is high.
(6)	it consumes less time because of quick searching	it consumes moderate time because of slow searching.
(7)	it is less lengthy while implemented.	it is more lengthy while implemented.
(8)	it is more efficient.	it is less efficient.
(9)	Ex Greedy search, A* search, AO* search Hill climbing Algorithm	Ex. DFS, BFS, Branch & Bound.

Page No.
Date

Q.② What is production system? Explain the components of production system.

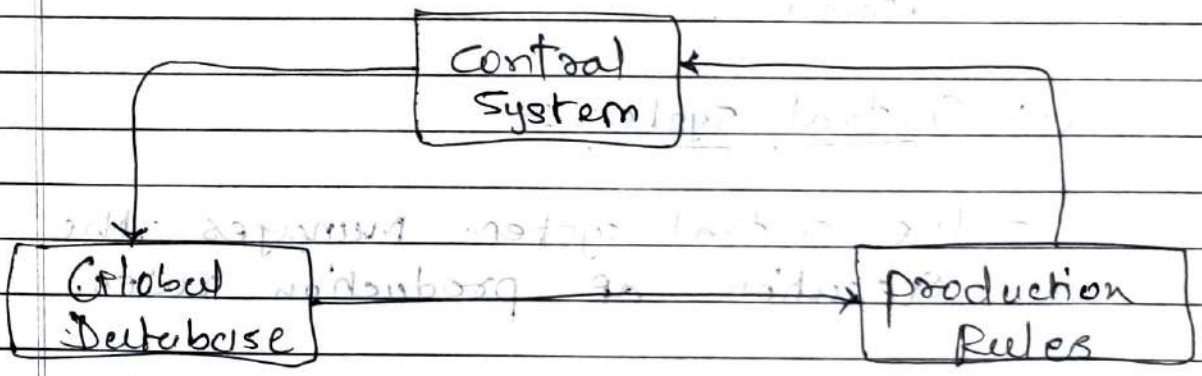
- Artificial Intelligence Production systems are the backbone of decision-making.

- These systems automate complex tasks through production rules, efficiently processing data and generating insights.

- Production systems key features are simplicity, modularity, adaptability and modifiability.

- AI production systems are classified into various types based on their characteristics guiding reasoning with control strategies like forward and backward chaining.

⊛ Components of Production System :-



⊛ components of AI P.S ⊛

- The components of an AI Production system encompasses three essential elements:

(1) Global Database :-

- The global database serves as the system's memory, storing facts, data and knowledge relevant to its operation.

- it is a repository that the production rules can access to make informed decisions and draw conclusions.

(2) Production Rules :-

- Production rules form the core logic of the system.

- They are a set of guidelines that the system follows while making decisions.

(3) Control System :-

- The control system manages the execution of production rules.

- It determines the sequence in which rules are applied, ensuring efficient processing and optimizing the system's performance.

Page No.
Date

Q. (3) What is Heuristic search? Explain with Example.

- A Heuristic is a technique that is used to solve a problem faster than the classic methods.

- These techniques are used to find the approximate solution of a problem when classical methods do not.

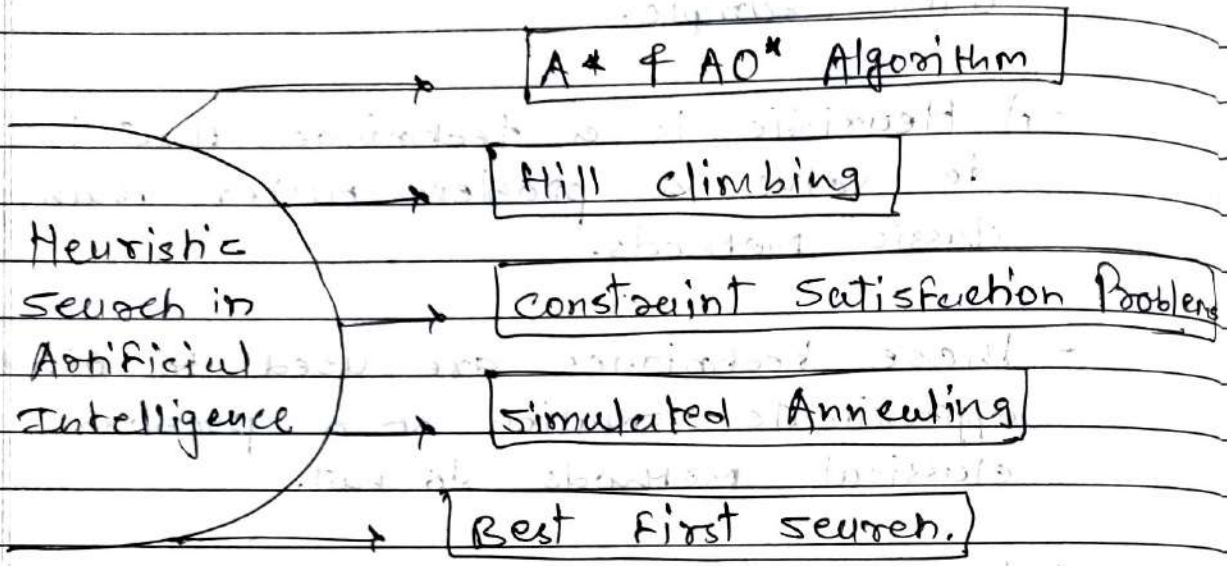
- Heuristic (or Heuristic function) takes a look at search algorithms. At each branching step, it evaluates the available information and makes a decision on which branch to follow.

- Heuristic use practical methods and shortcuts used to produce the solutions that may or may not be optimal, but those solutions are sufficient in a given limited timeframe.

- Heuristics are used in situations in which there is requirement of a short-term solution.

- On facing complex situations with limited resources and time Heuristics can help the companies to make quick decisions by shortcuts and approximated calculations.

⊛ Heuristic Search Techniques :-



⊛ There are two types of Heuristic techniques :-

(1) Direct Heuristic Search techniques
(2) Weak Heuristic search techniques

⊛ Direct Heuristic Search Techniques :-

- it includes Blind / uninformed Search and Blind control strategy. These search techniques are not always possible as they require much memory & time.

- These ~~search~~ techniques search the complete space for solution and use the arbitary ordering of operators.

Ex: Breadth-First-Search (BFS)
Depth-First-Search (DFS).

Page No.
Date

⊛ Weak Heuristic Search Techniques :-

- it includes informed search, Heuristic search and Heuristic control strategy.
- These techniques are helpful when they are applied properly to the right types of tasks.
- they usually require domain-specific information / knowledge.

Ex. Best-First-Search
A* Algorithm / search
AO* search
Hill climbing
Bidirectional search

⊛ Heuristic Function ($h(n)$) :-

- Heuristic is a function which is used in Informed search, and it finds the most promissing path.
- it takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.
- The Heuristic method, however, might not always give the best solution but it guaranteed to find a good solution in reasonable time.

- Heuristic function estimates how close a state is to the goal.

- it is represented by $h(n)$, and it calculates the cost of an optimal path between the pair of states.

- The value of Heuristic function is always positive.

⑥ Admissibility of the Heuristic Function:

$$h(n) \leq h^*(n)$$

- here $h(n)$ is heuristic cost, and $h^*(n)$ is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.

⑦ Heuristic Search Algorithms :-

(1) Best First Search Algorithm
(2) Hill climbing Algorithm
(3) A* Algorithm
(4) AO* Algorithm
(5) Constraint satisfaction propagation Algorithm
(6) Simulated Annealing Algorithm
(7) Bidirectional Search Algorithm
(8) Beam search Algorithm

- Heuristic search Algorithm expands nodes based on their heuristic value h(n).

⊛ Best First Search Algorithm ⊛

- Greedy best-first-search Algorithm always selects the path which appears best at that moment.
- it is the combination of Depth-first-search and Breadth-first-search algorithms.
- it uses the Heuristic function f search.
- Best-first-search allows us to make take the Advantages of both Algorithms.
- With the help of best-first-search, at each step, we can choose the most promising node.
- In the BFS algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by Heuristic function. i.e,

$$f(n) = g(n)$$

were, $h(n)$ = estimated cost from node n to the goal.

- the greedy BFS algorith is implemented by the priority queue.

- Heuristic search maintains two lists, OPEN & CLOSED list. in the CLOSED list, it places those nodes which have already expanded and in the OPENED list, it places nodes which have yet not been expanded.

⑧ Best first search Algorithm :-

(1) place the starting node into the OPEN list.

(2) if the OPEN list is empty, Stop & return value.

(3) Remove the node n, from the OPEN list which has the lowest value or h(n), and places it in the CLOSED list.

(4) Expand the node n, and generate the successor of node n.

(5) check each successor of node n, and find whether any node is a goal node or not. if any successor node is goal node, then return success & terminate the search. else proceed to step 6.

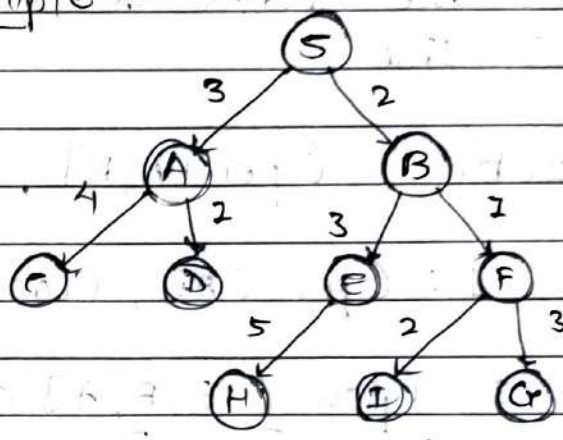
(6) For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list.

- if the node has not been in both list, then add it to the OPEN list.

(7) Return to step (2).

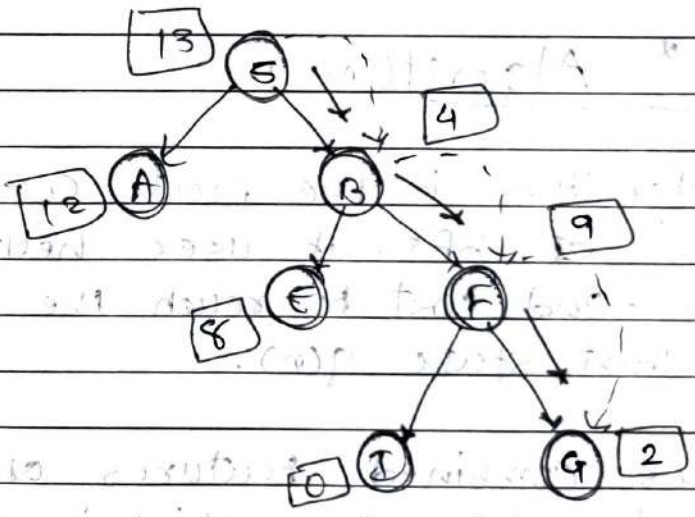
Page No.
Date

Example :-



node	Hen)
A	12
B	4
C	7
D	3
E	8
F	2
G	0
H	4
I	9
S	13

- Consider this example, or search problem.
- we will traverse it using greedy bfs.
- At each node is expanded using evaluation function f(n) = h(n), which is given in table.
- In this example, we are using two lists which are OPEN and CLOSED lists. Following are the iteration for traversing the above example.



Page No.
Date

⊛ Expand the nodes of S and put in CLOSED list

Initialization: Open [A, B], closed [S]

Iteration 1: Open [A], closed [S, B]

Iteration 2: Open [E, F, A], closed [S, B]
: Open [E, A], closed [S, B, F]

Iteration 3: Open [I, G, E, A], closed [S, B, F]
: Open [I, E, A], closed [S, B, F, G]

Final

Hence the solution path will be:
S ---> B ---> F ---> G

Time complexity: $O(b^d)$
Space complexity: $O(b^d)$
complete & Optimal.

⊛ A* Algorithm :-

- A* Algorithm is the most commonly known form of bfs. it uses heuristic function $h(n)$, and cost to reach the node n from the start state $g(n)$.

- it has combined features of UCS and greedy bfs, by which it solves the problem efficiently.

- A* Algorithm finds the shortest path through the search space using the heuristic function.

- The search algorithm expands less search tree and provides optimal result faster.

- A* algorithm is similar to UCS except that it uses $g(n)+h(n)$ instead of $g(n)$.

- In A* Algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called a "fitness number".

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

- $f(n)$: Estimated cost of the cheapest solution
- $g(n)$: cost to reach node n from start state
- $h(n)$: cost to reach from node n to goal node

A* Algorithm :-

(1) place the starting node in the OPEN list

(2) Check if the OPEN list is empty or not, if the list is empty then return value failure and stops.

Step: (3) Select the node from the OPEN list which has the smallest value of evaluation function ($g+h$), if node n is goal node then return success and stop, otherwise.

(4) Expand node n and generate all of its successors, and put n into the closed list.

- For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into open list.

(5) Else if node n' is already in OPEN & CLOSED, then it should be attached to the back pointer which reflects the lowest $g(n')$ value.

(6) Return to step (2).

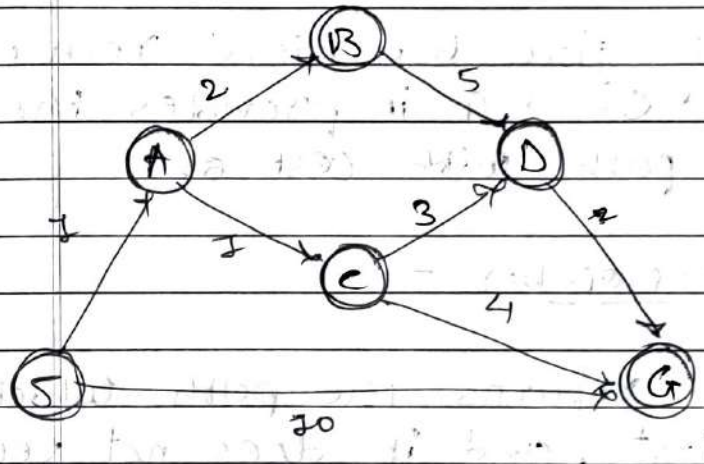
(4) Example:-

- In this example, we will traverse the given graph using A* Algorithm.

- The heuristic value of all states is given below table so we will calculate the $f(n)$ of each state using the formula $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of

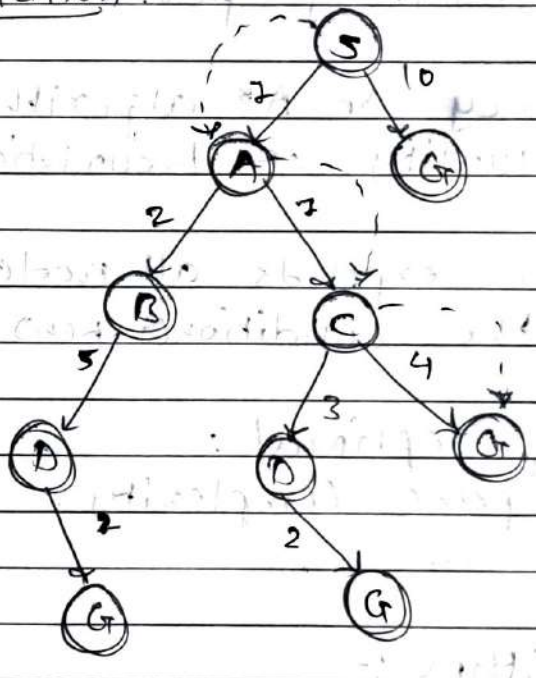
reach any node from start state.

- Here we will use OPEN & CLOSED lists.



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

② Solution:



Initialization: {(S, 5)}

Iteration 1: {(S --> A, 4) (S --> G, 10)}

Iteration 2: {(S --> A --> C, 4)
(S --> A --> B, 7)
(S --> G, 10)}

Page No.
Date

Iteration 3: { (S → A → C → G, 6),
(S → A → C → D, 11),
(S → A → B, 7),
(S → G, 10) }

Iteration 4: will give the final result, as S → A → C → G it provides the optimal path with cost 6.

Points to remember :-

- A* Algorithm returns the path which occured first, and it does not search for all remaining paths.
- The efficiency of A* algorithm depends on the quality of heuristic.
- A* algorithm expands all nodes which satisfy the condition $f(n)$.

Complete & optimal :
Time & space complexity : $O(b^d)$.

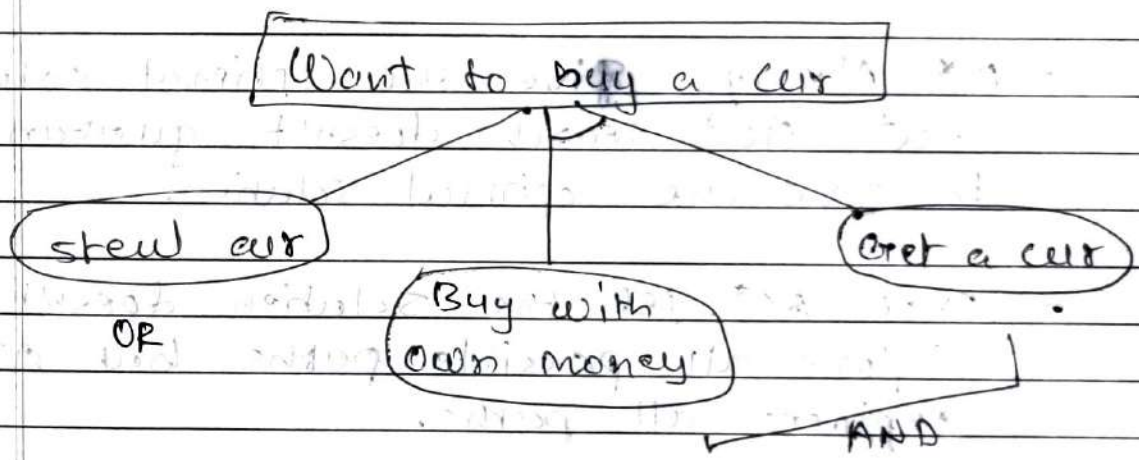
AO* Algorithm :-

- Best-First-search is what AO* Algorithm does.
- The AO* method divides any given difficult problem into a smaller group of problems

that are given then resolved using the AND-OR graph concept.

- AND-OR graphs are specialized graphs that are used in problems that can be divided into smaller problems.

- The AND side of graph represents a set of tasks that must be completed to achieve the main goal, while the OR side of the graph represents different methods for accomplishing the same main goal.



- in the above figure, the buying of a car may be broken down into smaller problems or tasks that can be accomplished to achieve the main goal in the above figure, which is an example of a simple AND-OR graph.

- The start state and target state is already known in the knowledge based

search strategy known as the AO* Algorithm, and the best path is identified by heuristics.

- The AO* algorithm is far more effective in searching AND-OR trees than A* Alg.

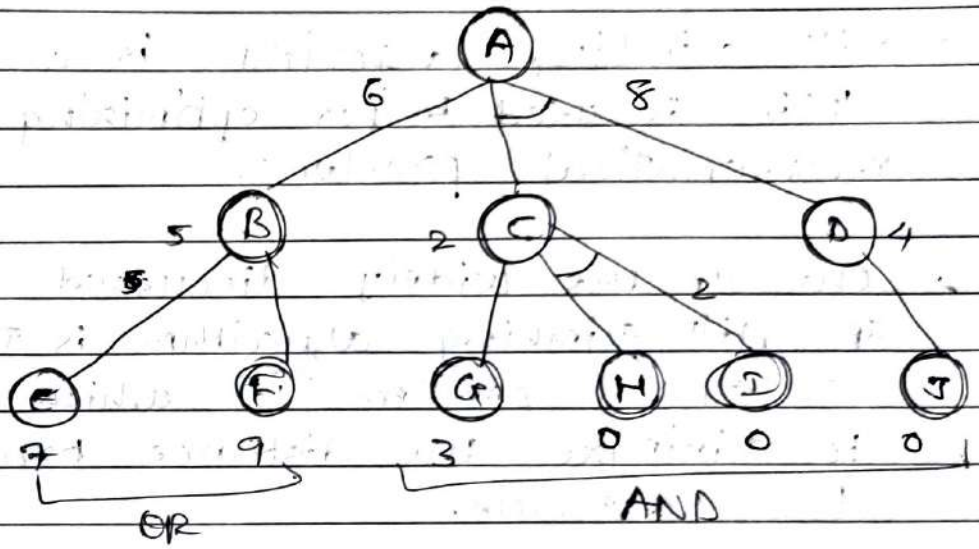
Difference Between the A* & AO* Algo:-

- A* Algorithm and AO* Algorithm both works on the best first search.
- they are both informed search and works on given heuristic values.
- A* Always gives the optimal solution but AO* search doesn't guarantee to give the optimal solution.
- Once AO* get the solution doesn't explore all possible paths but A* explores all paths.
- when AO* got compared to the A* Algo, the AO* algorithm uses less memory.
- Opposite to the A* Algorithm the AO* Algorithm can not go into endless loop

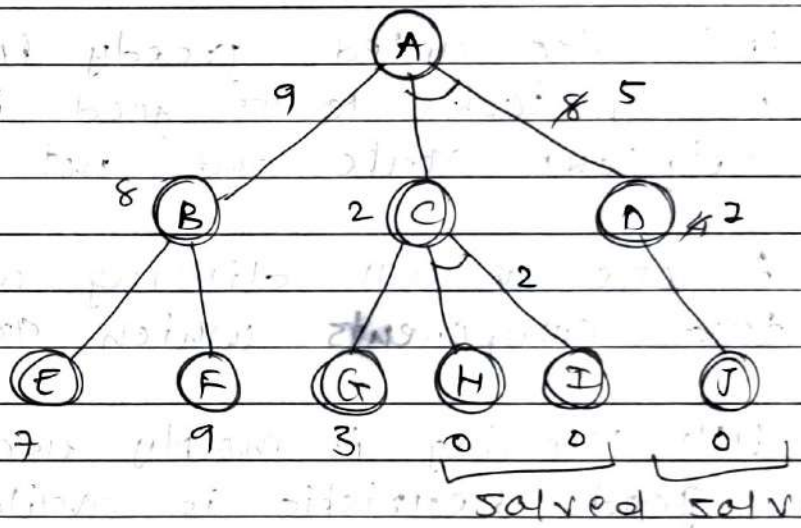
Page No.
Date

Start

Eg:



Solution:



⊛ Hill climbing Algorithm :-

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem.

- it terminates when it reaches a peak value where no neighbour has a higher value.

- Hill climbing algorithm is a technique which is used ~~to~~ for optimizing the mathematical problems.

- One of the widely discussed examples of Hill climbing algorithm is Traveling-Salesman problem in which we need to minimize the distance traveled by the salesman.

- it is also called greedy local search as it only looks to its good immediate neighbour state and not beyond that.

- A node of hill climbing algorithm has two components which are state & value.

- Hill climbing is mostly used when a good heuristic is available.

- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

⊛ Features of Hill climbing Algorithm :-

(1) Generate and Test Variant
(2) Greedy approach
(3) No backtracking

Page No.
Date

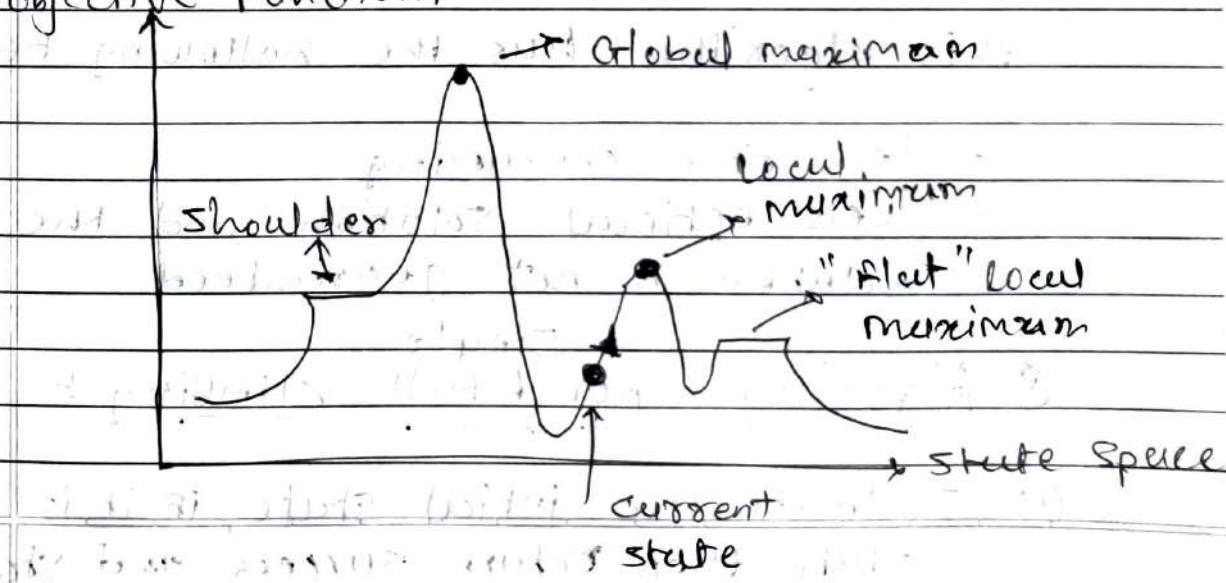
⊛ State-space diagram for Hill climbing:-

- The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and objective function/cost.

- On Y-axis we have taken the function which can be an objective or cost function, and state space on the x-axis.

- if the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum.

- if the function of Y-axis is objective function then, the goal of the search is to find the global maximum & local maximum.



Page No.
Date

⊛ Types of Hill climbing Algorithm :-

(1) Simple hill climbing
(2) Steepest - Ascent hill - climbing
(3) Stochastic hill climbing.

⊛ Simple Hill Climbing :-

- simple hill climbing is the simplest way to implement a hill climbing algorithm.

- it only evaluates the neighbour node state at a time and selects the first one which optimizes current cost and set it as the a current state.

- it only checks it's one successor state, and if it finds better than the current state, then move else be in the same state.

- this algorithm has the following features:
  - less time consuming
  - Less optimal solution and the solution is not guaranteed

⊛ Algorithm for Simple Hill climbing :-

(1) Evaluate the intial state, if it is goal state then return success and stop.

(2) Loop until a solution is found or there is no new operator left to apply.

(3) Select and apply an operator to the current state.

(4) Check new state:

a. If it is goal state, then return success and quit.

b. Else if it is better than the current state then assign new state as a current state.

c. Else if not better than the current state, then return to step (2).

(5) Exit.

Algorithm :-

* Problems / Limitations Hill climbing

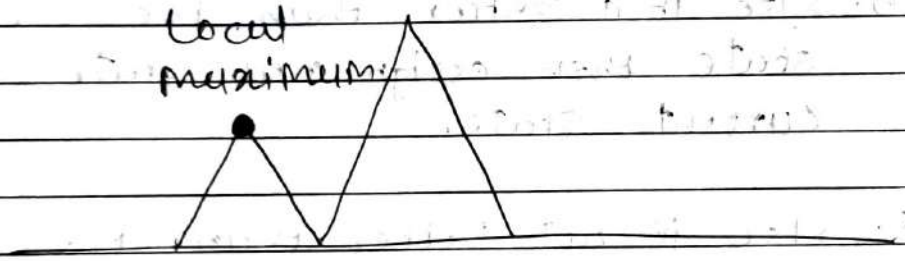
(1) Local Maximum :-

- A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution:

- Backtracking technique can be a solution of the local maximum in state space landscape.

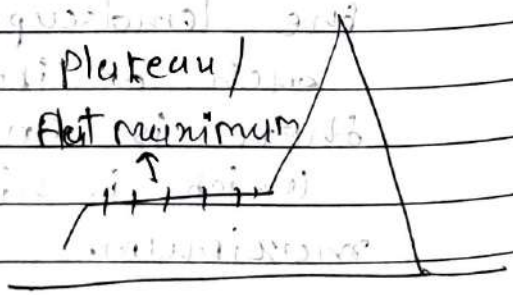
- Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well ~~as~~.



(2) Plateau :-

- A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value because of this algorithm does not find any best direction to move.

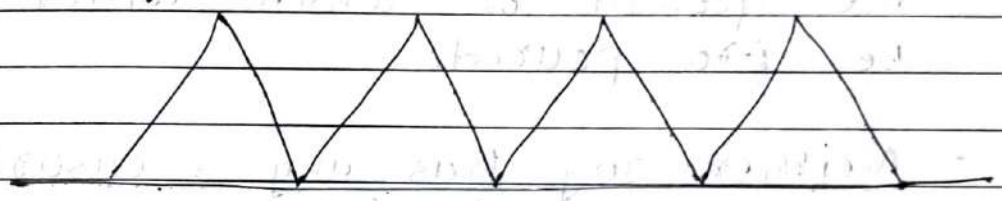
- The hill climbing search might be lost in the plateau area.



(3) Ridge :-

- A ridge is a special form of the local maximum. it has an area which is higher than its surrounding areas, but itself has a slope and cannot be reached in a single move.

Ridge



Solution :-

- with the use of bidirection search, or by moving in different directions, we can improve this problem.

Solution of Plateau :-

- The solution for the plateau is to take big steps or very little steps while searching, to solve the problem.

- Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.

Page No.
Date

Q.④ Water Jug problem :-

⊛ problem Definition :-

- You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured.

- Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

(1) Initial state :-

- We will represent a state of the problem as a tuple (x, y), where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug.

- Note that $0 \leq x \leq 4$, and $0 \leq y \leq 3$.

- Here the initial state is (0, 0), the goal state is : (2, n) for any value of n.

(2) Production Rules :-

Sr.	current state	Next state	Descriptions
(1)	(x,y) if $x<4$	$(4,y)$	Fill the 4 gallon jug.
(2)	(x,y) if $y<3$	$(x,3)$	Fill the 3 gallon jug
(3)	(x,y) if $x>0$	$(x-d,y)$	pour some water out of the 4 gallon jug
(4)	(x,y) if $y>0$	$(x,y-d)$	pour " 3 gallon jug
(5)	(x,y) if $x>0$	$(0,y)$	Empty the 4 gallon jug on the ground
(6)	(x,y) if $y>0$	$(x,0)$	Empty the 3 " " ground
(7)	(x,y) if $x+y\geq 4$ and $y>0$	$(4,y-(4-x))$	pour water from the 3 gallon jug into the 4 gallon jug until the 4 gallon jug is full.
(8)	(x,y) if $x+y\geq 3$ and $x>0$	$(x-(3-y),3)$	pour water from the 4 gallon jug into the 3 gallon jug until the 3 gallon jug is full
(9)	(x,y) if $x+y\leq 4$ and $y>0$	$(x+y,0)$	pour all water from the 3 gallon jug into 4 gallon jug.
(10)	(x,y) if $x+y\leq 3$ & $x>0$	$(0,x+y)$	pou.. 4 into 3 g. jug
(11)	$(0,2)$	$(2,0)$	pour 2 gallons from 3 gallon jug into the 4g.j
(12)	$(2,y)$	$(0,y)$	Empty the 2 gallons in the 4 gallon jug on the ground.

Page No.
Date

* One of the possible solution given as:

(3) productions for the water jug problem:

Gallons in the 4-g. jug	Gallons in the 3-g.j	Rule Applied.
(0)	0	2
0	3	9
3	0	2
3	3	7
4	2	5 or 12
0	2	9 or 11
2	0	--

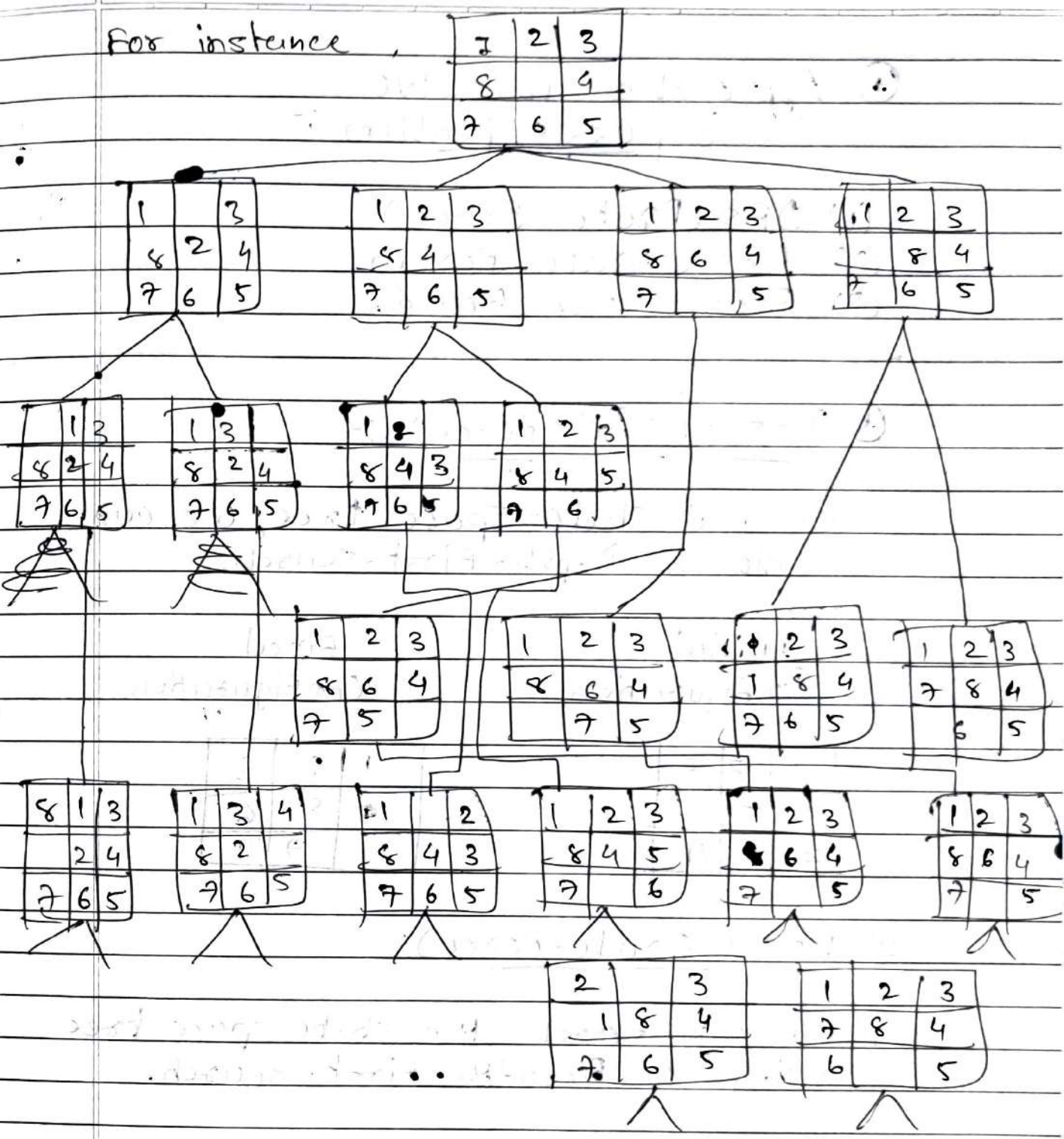
* 8-puzzle problem:-

- 8-puzzle problem is problem in which A 3 by 3 board with 8 tiles (each tile has a number from 1 to 8) and a single empty space is provided.

- The goal is to use the vacant space to arrange the numbers on the tiles such that they match the final arrangement.

- Four neighbouring (left, right, above, and below) tiles can be moved into the available area.

For instance,



⊛ State space Representation of 8-puzzle problem ⊛

Page No.
Date

① Approaches to solve 8-puzzle problem :-

(1) DFS (Brute-Force)
(2) BFS (Brute-Force)
(3) Branch and Bound.

② DFS (Brute-Force) :-

- on the state-space tree, we can do a Depth-First-search.

Initial Configuration

1	2	3
5	6	
7	8	4

Final Configuration

1	2	3
5	8	6
	7	4

③ BFS (Brute-Force) :-

- we can search the state space tree using a Breadth-First-search.

④ Branch and Bound :-

- By cost Function.

$$c(x) = g(x) + h(x)$$

Page No.
Date

Chapter : 1

Define AI :-

- Artificial intelligence is the capacity of a computer or robot to accomplish operations frequently carried out by intelligent people.

- According to an expert in the field, Ai is a set of algorithms capable of producing results without the need for explicit training.

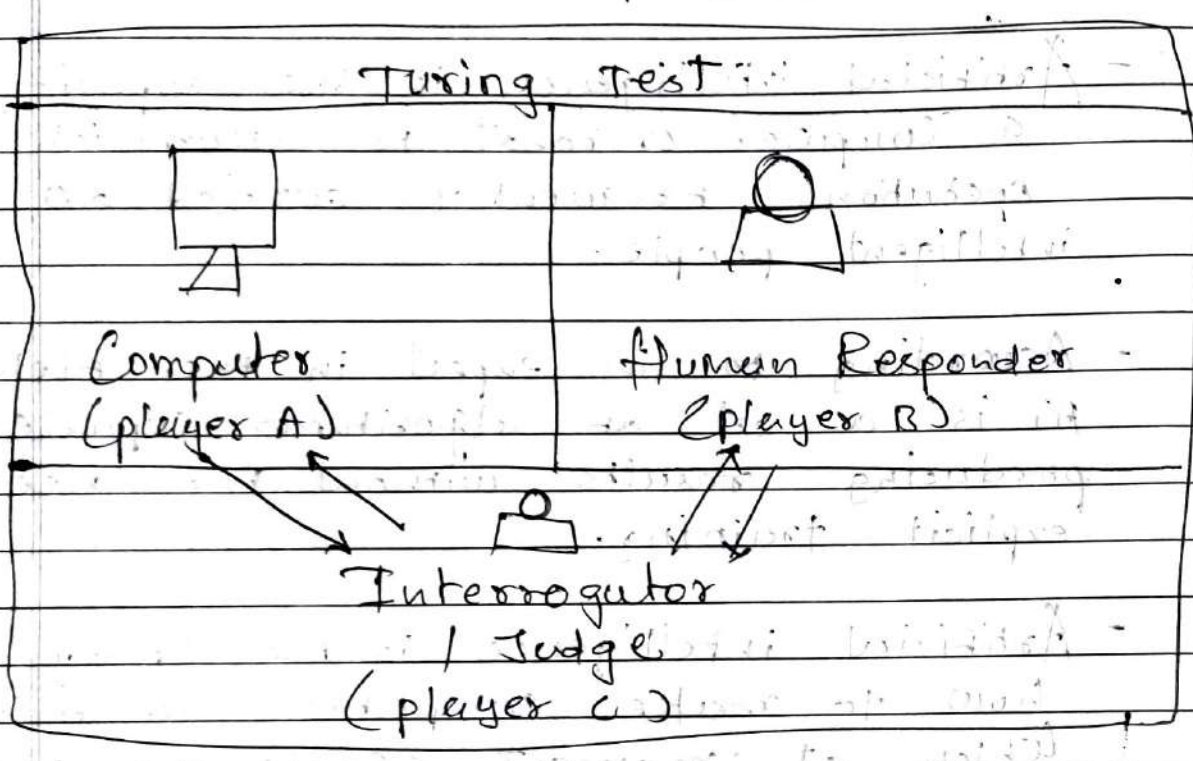
- Artificial intelligence is the study of how to make computers do things which, at moment people do better

  - Ai perspectives - perspective of
    - Intelligence
    - Business
    - Programming

- Turing Test → Developed by Alen Turing in 1950
  - used to determine whether or not a computer or machine can think intelligent like humans.

- if judge is unable to distinguish between the human and the machine based on the conversation, then the based

on the conversation, then the machine is said to have passed the "TT".



② Task Domains of Ai :-

④ Mundane Tasks :-

- Perception - Computer vision
  - Speech, Voice
- Natural Language Processing - Understanding, Language Generation, Language Translation.
- Planning
- Common Sense Reasoning
- Robot control.

Page No.
Date

① Formal Task :-

- Games - Go | chess | checkers | Tic-Tac-Toe.
- Mathematics - Geometry
  - Logic
  - Integration and Differentiation
- Theorum proving

② Expert Tasks :-

- Engineering - Design
  - Fault Finding
  - Manufacturing
  - Monitoring
- Scientific Analysis
- Financial Analysis
- Medical Diagnosis

③ AI Problems :-

- Complexity
- Non-linearity
- Context dependence
- Creativity
- Learning and ~~adaption~~ adaptation
- Multi-disciplinary
- Ethical considerations

Page No.
Date

① Applications of Ai :-

- Gaming
- Natural language processing
- Expert system
- Computer Vision Systems
- Speech Recognition
- Handwritten Recognition
- Intelligent Robots

⊛ chapter: ②

① Informed & Uninformed Search :-

② Production System :-

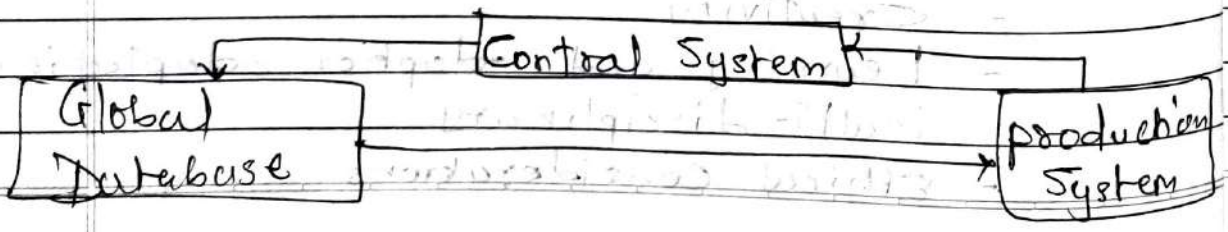
↓

Backbone of decision making

③ Features of production System :-

- Simplicity
- modularity
- adaptibility
- modifiability

④ Components of Production System :-



Page No.
Date

③ Global Database → serve Memory.
↓ Store - facts, knowledge & data's.
a repository that production rules use/access to make informed decisions & draw conclusions.

④ Production Rules :- core logic of system
- set of guidlines that system follow while making decisions.

⑤ Control System :- manage execution of production rules.
↓
determine sequence in which rules are applied, ensuring efficient processing & optimizing systems performance.

⑥ Heuristic Search :- solve problems faster than classic methods.
↓
used to find approximate solution.

⑦ Heuristic Techniques :-

- A* & AO* Algorithm
- Hill climbing
- Constraint Satisfaction problem
- simmulated Annealing
- Best First search
- Bidirectional Algorithm
- Beam Algorithm.

Page No.
Date

③ two types of Heuristic Techniques :-

(1) Direct Heuristic search Techniques :-

↓

Blind / uninformed search and Blind control strategy. → DFS, BFS & Bound f

(2) Week Heuristic search Techniques :-

↓

Informed search Heuristic search & Heuristic control Strategy → A*.

④ Heuristic Function (h(n)) :-

- Finds most promising path.
- take current state of the agent as its input and produce the estimation cost of how close agent is from the goal.
- Always positive.

⑤ Admissibility of Heuristic function :-

$$h(n) \leq h^*(n)$$

↓ ↓

heuristic Function or heuristic cost

↓ ↓

estimated Cost. or estimated function

Page No.
Date

⊛ Best-First-Search :-

Always select path which appears best to that moment.

Combination of BFS and DFS.

* Uses heuristic function & search.

$f(n) = h(n)$, → estimated cost from node n to the goal.

priority

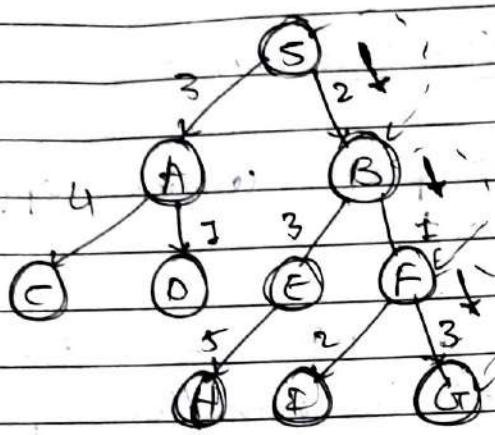
Greedy bfs Algorithm implemented by queue.

⊛ Algorithm of BFS :-

(1) Starting node → OPEN list
(2) if OPEN list empty → stop & return
(3) else Remove node n from → OPEN list which has lowest $h(n)$ value & place → CLOSED list.
(4) Expand node n → generate the successor of node n.
(5) Check each successor of node n & find whether node is → goal or ⊗
if any successor goal → return success & terminate search else → proceed to step (6).
(6) For each successor node, algorithm check for $f(n)$ → evaluation function then check if node → either in OPEN or CLOSED list.
→ if ⊗ in both list add to OPEN list
(7) else return to step (2).

Page No.
Date

Eg:



node	H(n)
A	12
B	4
C	7
D	3
E	8
F	2
G	0
H	4
I	9
S	13

Time & space complexity: $O(b^d)$
complete and optimal.

where b = branch factor
d = depth of the tree.

⑥ A* Algorithm :-

it uses heuristic function & cost to reach node n from the start state (g(n)).

Combine features of UCS & greedy bfs.

which solve problem efficiently.

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

- $f(n)$: estimated cost of the cheapest solution
- $g(n)$: cost to reach node n from the start node
- $h(n)$: Cost to reach node n from the goal node

⑦ A* Algorithm :-

(1) Place starting node → OPEN list.
(2) check OPEN list if empty or ⑦
if empty → return failure & stop.
value

(3) else select node from → OPEN list which has smallest value of evaluation function (f+h), if node n → goal node. return success & stop.

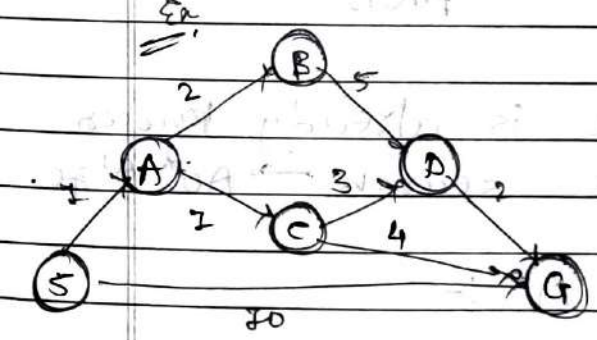
(4) else expand node n & generated all of its successors & put in into the → CLOSED list.

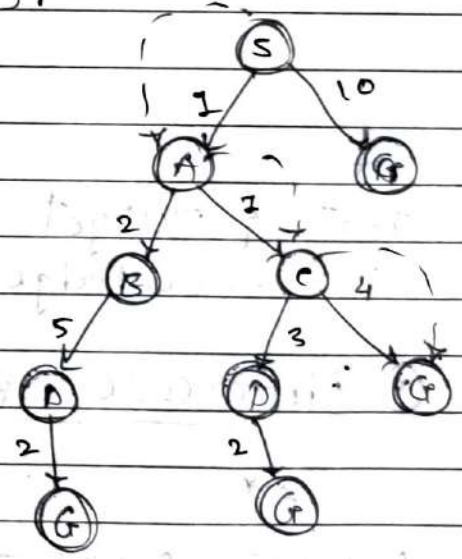
(5) - for each successor n' → check whether n' is already → OPEN or CLOSED if (X) → then compute F(n) for n' & place into → OPEN list.

(5) else if node n' → already in OPEN & CLOSED → then it should be attached to the backpointer which reflects the lowest g(n') value.

(6) Return to step (2).

Ex:





State	S	A	B	C	D	G
h(n)	5	4	3	2	6	0

Complete & optimal

Time & space complexity: $O(b^d)$.

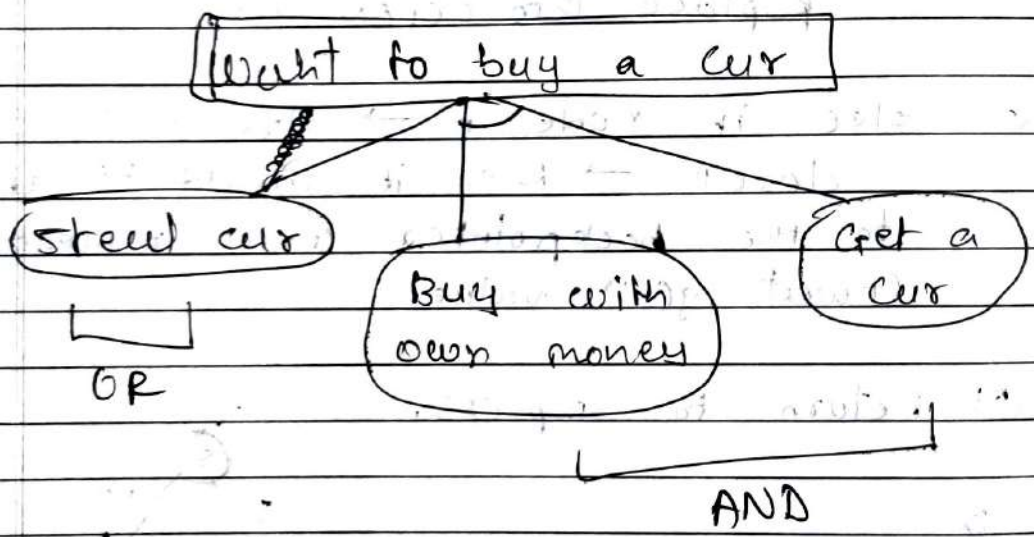
Page No.
Date

⊛ AO* Algorithm :-

divide difficult problems → smaller groups of problems which is resolved using AND-OR graph.

⊛ AND-OR Graph :-

- Represent a set of task that must be completed to achieve main goal.
- Represent different methods for accomplishing the same main goal.



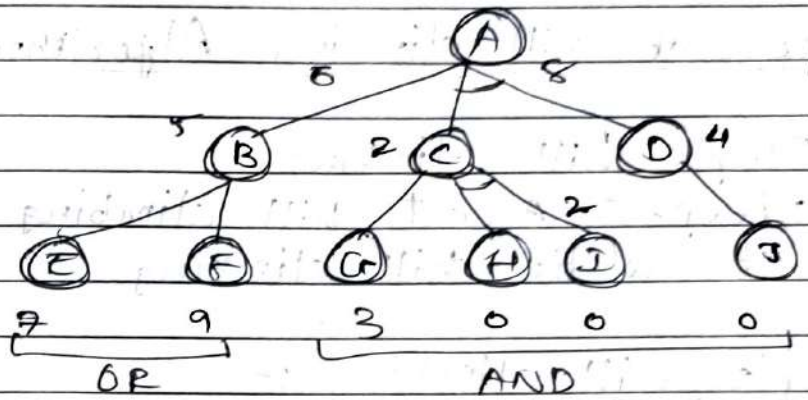
start & target state is already known in knowledge base search → AO* Algo.

⊛ Difference from A* :-

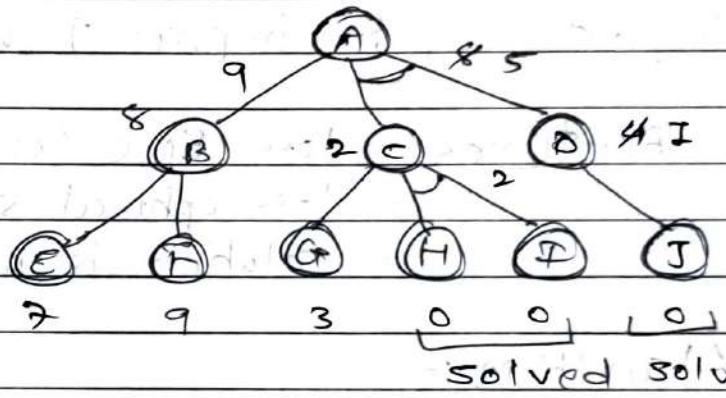
A*	AO*
memory more	memory less
optimum solution	not guarantee of o.solution
does explore all paths	doesn't explore all paths
go in endless loop	don't go in endless loop.

Page No.
Date

Ex:

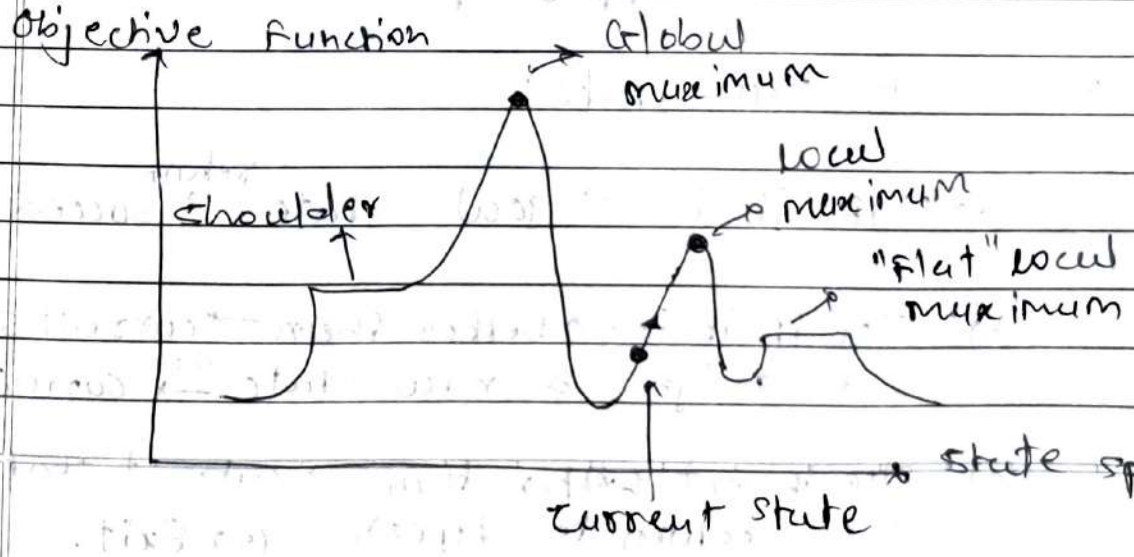


Soln:



③ Hill climbing Algorithm :-

Local searching algorithm → continuosly moves in increasing direction of increasing value → to find peak of mountain or best solution to the problem.



Page No.
Date

⊛ Types of hill climbing Algorithm :-

(1) Simple hill climbing
(2) steepest-Ascent hill climbing
(3) Stochastic hill climbing

⊛ simple hill climbing :-

↓

simplest way to implement hill climbing

⊛ Features : less time consuming
- less optimal solution
- solution is not guaranteed

⊛ Algorithm :-

(1) Evaluate → initial state; if → goal then return success & stop.

(2) else loop Until solution is found or there is no new operator left to apply.

(3) select and apply operator → to current state

(4) check new state :

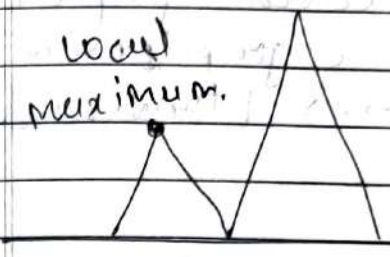
(a) : if it is → goal state → return success & stop

(b) : Else if it is → better than → current state → assign a new state as → current state

(c) : Else if not better than → current state then return to step(2). (5) Exit.

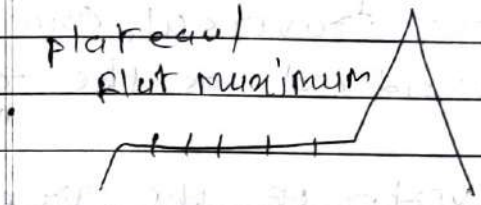
Page No.
Date

① Local maximum :-



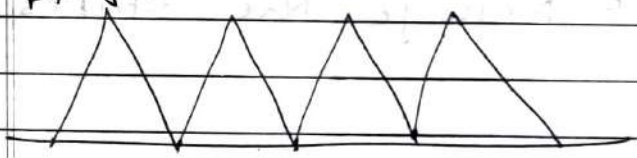
Solution : Backtracking

② plateau / flat maximum :-



Solution : take big steps & select random states.

③ Ridge :-



Solution : bidirectional search, means move to different directions.

④ Uniformed search techniques :-

(1) Breadth-first-search (Queue)
(2) Depth-first-search (stack)
(3) Branch & Bound. (priority Queue)

Page No.
Date

⊛ Breadth-first-search :-

- it is a Uniformed search graph traversing Algorithm in which the graph from the root node and explores [traverse] all the neighboring nodes.

- Then, it selects nearest node & explores all the unexplored nodes.

- while using BFS for traversal, any node in the graph considered as the root node.

- BFS puts every vertex of the graph into two categories - visited & non-visited.

- BFS uses Queue Data structure to traverse & manage the states.

⊛ Algorithm :-

(1) SET STATUS = 1 (ready state) for each node in G

(2) Enqueue the ~~following~~ starting node A and set its STATUS = 2 (waiting state)

(3) Repeat Step 4 and 5 until QUEUE is empty.

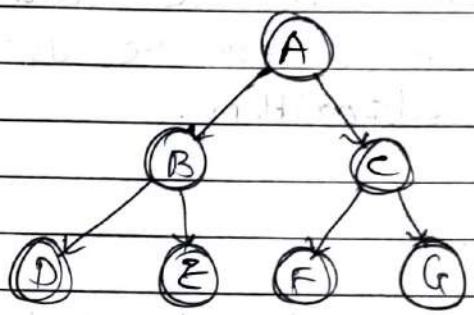
(4) Dequeue a node N. process it and set its STATUS = 3 (processed state).

(5) Enqueue all the neighboring of N that are in ready state (whose STATUS = 1) and set thier STATUS = 2 (waiting state)

[END OF LOOP]

(6) Exit.

Eg:



Steps:

A

A | B C

A, B | C D E

~~A, B~~ A

A, B, C | D E F G

A

A, B, C, D | E F G

A, B, C, D, E | F G

A, B, C, D, E, F | G

A, B, C, D, E, F, G

- Time Complexity O(bd).

① Depth-First-Search Algorithm :-

- The DFS starts with the initial node of graph G and goes deeper until we find the goal node or the node with no children.

- Because of recursive nature ~~use~~ stack data structure can be used to implement the DFS algorithm.

② Algorithm :-

(1) SET STATUS = 1 (ready state) for each node in G.

(2) Push the starting node A on the stack and set its STATUS = 2 (waiting state)

(3) Repeat step 4 & 5 until STACK is empty.

(4) POP the top Node n. Process it and set its STATUS = 3 (processed state)

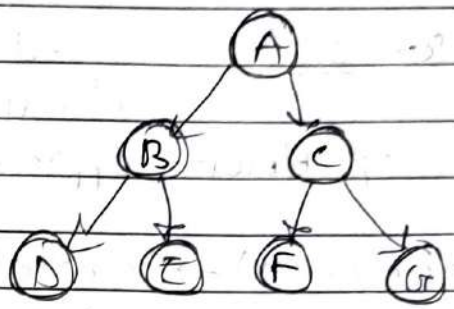
(5) Push on the stack all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state).

[END of LOOP]

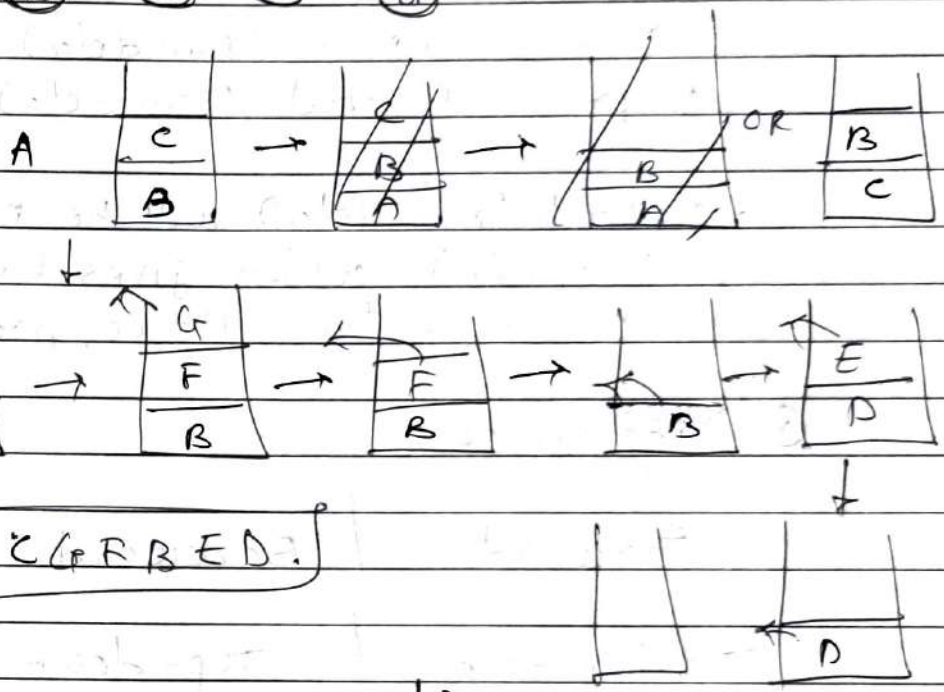
(6) Exit.

Page No.
Date

Ex



Steps :



→ A C G F B E D.

Time Complexity $O(b^d)$.

---

⑧ Chapter: ③ ~~top-down~~ approach.
↑ Bottom-up

④ Forward reasoning :- used by an Expert Systems.

Ex. Rule: human(A) → mortal(A).
Data: human(Mandela)
TO Proove: mortal(Mandela)

- The data matches the left-hand side of the rule - so, we get A = Mandela.
- From that, we can get mortal(Mandela) based on the rule.

Page No.
Date

④ Backward reasoning :- used by expert diagnostic systems.
+
Bottom-up Top-Down approach.

Eg. Rule : human(A) → mortal(A)
Data : human (mandela)
to prove : mortal (mandela)

- mortal (mandela) will be matched with mortal(A) which gives human(mandela) which is true. Hence, proved.

⑤ Difference :

Forward	Backward reasoning
Bottom-up	Top-down
Generate next level of tree by finding all rules whose	
left hand side matches with root node &	Right hand side matches with the root node &.
right hand side is used to create new configuration	left hand side is used to create new configuration.

Page No.
Date

⊛ Issues in knowledge Representation.

- knowledge Representation → conclusions from knowledge.

(1) Important Attribute (Instance / ISA)
(2) Relationship among Attributes (Inverse of ISA)...
(3) Choosing Granularity (detail of know.)
(4) Set of objects (properties true as member but ⊗ individual)
(5) Finding Right structure

⊛ Approaches to knowledge representation:

kr. enables fast & accurate access to knowledge & Understanding of the content.

(1) Representational Adequecy
(2) Inferential Adequecy
(3) Inferential Efficiency
(4) Acquisitional Efficiency

(1) → ability to represent all kind... k that needed to domain
(2) → ability to manipulate the representational structures to → new structure corresponding to new kno... added.
(3) → ability to incorporate additional (infered from old.) information from knowledge structure.
(4) → ability to acquire knowledge from automation methods.

Ans of Semantic Net.

Page No.
Date

① Knowledge representation methods :-

- Logical Representation
  - a. propositional logic
  - b. predicate logic
- Semantic Network Representation
  - a. IS-A relation (Inheritance)
  - b. kind-of-relation
- Frame Representation
  - a. jerry is cat
  - b. jerry is mammal
  - c. jerry is owned by priya.
  - d. jerry in brown colored.
  - e. All mammals are animal.
- production Rules

Ex a. Student Buy Books and pencils

Buy (Student, Books) ∧ Buy (student, pencils)

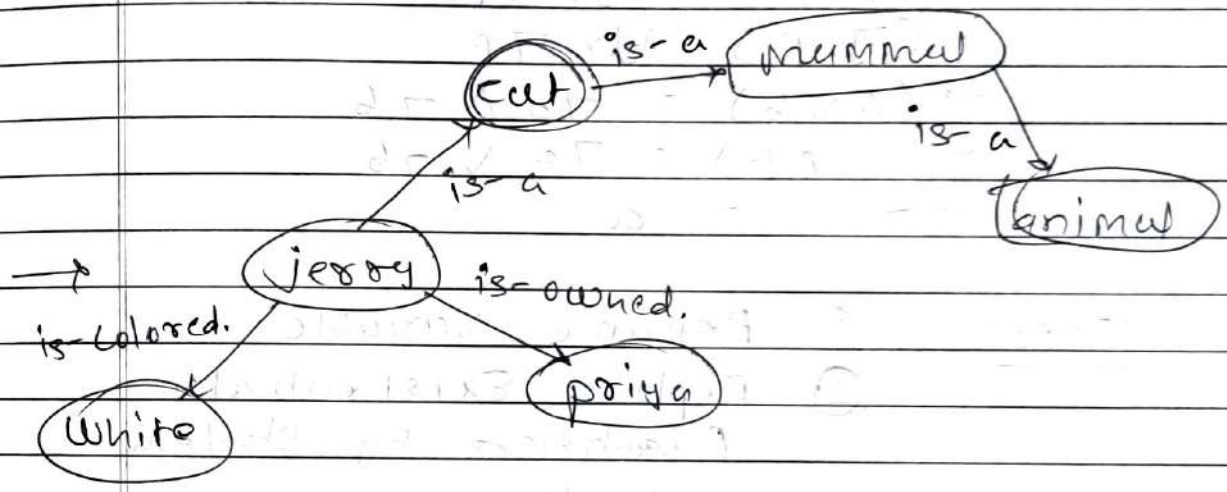
Ex b. All stu--

∀x : student (x) → Buy (x, Books) ∧ Buy (x, pencils)

② production Rules → Consists of (condition, action)

Ex IF (at bus stop AND bus arrives) THEN action (get into the bus).

slots	filters
Title	AI
Genre	Computer Science
Author	Rohan
Edition	Third Edition
year	2003
page	1152



⑦ Reproduction Rules :-

Steps : ① Negate statements to be prooved.

② Convert given facts into FOL.

③ Convert FOL into CNF (Conjuctive Normal Form)

④ Draw Resolution Graph.

Page No.
Date

④ FOL to CNF Rules ④

⇒ Eliminate '$\rightarrow$' & '$\leftrightarrow$'

$a \rightarrow b$ $\quad :. \neg a \vee b$

$a \leftrightarrow b$ $\quad :. a \rightarrow b \wedge b \rightarrow a$

⇒ Move '$\neg$' inward

- $\neg(\forall x P) = \exists x \neg P$
- $\neg(\exists x P) = \forall x \neg P$
- $\neg(a \vee b) = \neg a \wedge \neg b$
- $\neg(a \wedge b) = \neg a \vee \neg b$
- $\neg\neg a = a$

Steps:
1. Rename Variable
2. Replace Existential Quantifiers by skolem constant
   $\exists x$ Rich(x) = Rich(G1)
3. Drop Universal Quantifiers.

⑤ Prolog :-

⑥ Fail :-

- if we want to force a rule to fail under certain conditions then built in predicate called "Fail" is used in prolog.

- predicate fail tells prolog interpreter to fail a particular goal and subsequently force backtracking.

- All the sub goals defined after fail will never be executed.

- Hence, predicate fail, should always be used as the last sub goal in rule.

- it is also to be noted, that rule containing fail predicate will not produce any solution.

Ex. goal(x) :- failure(x), !. goal(x).

- failure(x) specifies the condition which makes goal(x) fail.

(*) cut :-

- Sometimes it is desirable to selectively turn off backtracking.

- cut always succeeds but cannot be backtracked.

- the cut effectively tells prolog to freeze all the decisions made so far in this predicate. That is, if required to back track, it will automatically fail without trying other alternatives.

- performance is main reason to use the cut.

- The symbol of cut predicate is "!".

Ex: minimum(x, y, x) :- x <= y, !.
minimum(x, y, y) :- x > y

- cut is used to tell the interpreter to not to look for solutions when it finds one.

⊛ Recursive Predicate :-

- Any function which calls itself is called recursive function.

- In prolog, recursion appears when a predicate contains a goal that refers to itself.

- This simply means a program calls itself typically until some final point is reached.

- In prolog and in any language, a recursive definition always has at least two parts.

- A first part that act like a stopping condition and a rule that call itself simplified.

- At each level the first fact is checked. if the fact is true then the recursion ends. if not the recursion continues.

- A recursive rule must never call itself with the same arguments. if that happen then program will never end.

Ex Find Factorial of given no.

```
fact.pl

fact(0, Result):-
        Result is 1.

fact(N, Result):-
        N>0,
        N1 is N-1,
        fact(N1, Result1),
        Result is Result1 * N.
```

```
output

Goal: fact(2, F).
F = 2

Goal: fact(3, F).
F = 6
```

⑧ programs :-

① to append two given lists into third:

- Append the list $L2$ at the end of another list $L1$ and put the resultant list in $L3$.

% If $L1$ is empty, resultant list will be equal to $L2$ (base case).

```
append_list ([], L2, L2).

append_list ([X|L1], L2, [X|L3]) :-
    append_list (L1, L2, L3).
```

Query:

```
?- append_list ([a,b,c], [p,q], Ans).

Ans = [a,b,c,p,q].
```

(2) to merge two sequentially ordered (ascending) list into one ordered list.

```
merge ([], [], []).
merge ([X], [], [X]).
merge ([], [Y], [Y]).
merge ([X|List1], [Y|List2], [X|List]) :-
        X <= Y, !,
        merge (List1, [Y|List2], List).

merge ([X|List1], [Y|List2], [Y|List]) :-
        merge ([X|List1], List2, List).
```

number of
(3) Find element in a list.

size([],0).
size([_|T],N):- Asynchronous Variable.

size(T, N1), N is N1+1.

Query:

?- size([1,2,3,4],N).
output → N=4

(4) Find nth element of a list.

nth_member(1,[X|L],X).
nth_member(N,[Y|L],X):-

N1 is N-1,
nth_member(N1,L,X).

Query:

?- nth_member(3,[1,2,3,4,5],X).
X=3

(5) Check if list is sorted

is_sorted([]).
is_sorted([_]).
is_sorted([X,Y|T]):-
X<=Y, is_sorted([Y|T]).

Page No.
Date

Query:

?- is_sorted ([1,8,2,9]).

Output → false

⑥ Find the sum of first N Natural numbers.

```
sumlist ([],0).
sumlist ([H|T],N):-
        sumlist (T,N1), N is N1 + H.
```

Query:

?- sumlist ([1,2,3],N).

output → N = 6

② gcd of two Numbers :-

```
predicates
        gcd (integer, integer, integer)

clauses

        gcd (M,0,M).
        gcd (M,N,Result):-
                rem = M mod N,
                gcd (N,Rem,Result).
```

Query: ?- gcd (6,4,Result).

output Result = 2