courses.csail.mit.edu/6.034f/ai3/ch1.pdf

ARTIFICIAL INTELLIGENCE

- **UNIT 1: Introduction**: Pg-1
  - Introduction to AI
  - What is Artificial Intelligence
  - Semantic Nets and Description Matching
  - Generate and Test
  - Means Ends Analysis
  - Problem Reduction-exploring Goal Trees.
  - SLE: Chinese Room

- **UNIT 2: Search and Rule Based systems**: Pg-20
  - Nets and Basic Search
  - Optimal search:
    - British Museum
    - Depth First
    - Breadth First
    - Hill climbing
    - Beam
  - Trees and Adversarial Search (UNIT 3)
  - Rule and Rule chaining:
    - Rule-based deduction systems
    - Procedures for forward and backward chaining
    - Rule-based systems as substrate
  - SLE: Best First Search and AO* algorithm

- **UNIT 3: Game playing and CSP**: Pg-49
  - MINIMAX Gaming
  - Alpha-Beta Pruning
  - Propagation of Probability bounds through opinion Nets
  - (constraint Satisfaction Problems)

SLE: Numeric constraints and Propagation
- Numeric constraints (cryptoArithmetic)

- **UNIT 4: Different Learnings:** Pg-67
  - Learning by Analyzing Differences
  - Introduction to Learning
  - Nearest Neighbors
  - Learning:
    - Identification Trees
    - Disorders
  - SLE: Inductive Logical Programming

- **UNIT 5: Neural Networks:** Pg-81
  - Training by Neural Nets
  - Deep Neural Nets
  - Back Propagation in NN
  - Convolution Neural Network (CNN)
  - Learning by Genetic Algorithms
  - SLE: Hopfield Neural Network

- **UNIT 6: SVM and Probabilistic Models:**
  - Support Vector Machines
  - Learning Probabilistic Models:
    - Statistical Learning
    - Learning with complete data
    - Learning with hidden data:
      - The EM algorithm
  - SLE: Learning Boosting

TEXTBOOK:

"Artificial Intelligence", Patrick Henry Winston, 3rd Edition, 1999 or later

UNIT - 1

Introduction

- **Introduction to AI:**
  - Artificial Intelligence is the study of the computations that makes it possible to perceive, reason and act.
  - Engineering goal of artificial intelligence is to solve real-world problems as a collection of ideas about representing knowledge, using knowledge and assembling systems.
  - Scientific goal of artificial intelligence is to determine which ideas about representing knowledge, using knowledge and assembling systems explain various sorts of intelligence.
  - A scientific and engineering discipline devoted to:
    - understanding principles that make intelligent behavior possible in natural or artificial systems.
    - developing methods for the design and implementation of useful, intelligent artifacts.

- **What is Artificial Intelligence:**

Artificial Intelligence is the ability of a computer or other machine to perform those activities that are normally thought to require intelligence.

It is the subfield of computer science concerned with the concepts and methods of symbolic inference by computer and symbolic knowledge representation for use in making inferences. AI can be seen as an attempt to model aspects of human thought on computers.

It is also sometimes defined as trying to solve any problem by computer faster than a human can solve.

- Operational definition of AI
  - Systems that "act" like humans : Turing test

The turing test is a method of inquiry in AI for determining whether or not a computer is capable of thinking like a human being.

  - systems that "think" like humans: cognitive science

The cognitive science brings together computer models from AI and experimental techniques to try to construct precise and testable theories of the workings of the human mind.

  - Systems that "think" rationally: logic based AI

The law of thought approach was supposed to govern the operation of the mind, this initiated the field called logic.

  - systems that "act" rationally: Rational agents

Rational behavior is doing the right thing that which is expected to maximize goal achievement given the available information.

- What AI can do:
  - Intelligent systems can help experts to solve difficult analysis problems
  - Intelligent systems can help experts to design new devices
  - Intelligent systems can learn from examples
  - Intelligent systems can provide answers to english questions using both structured data and free text

- Criteria for Success:

To determine if research work in AI is successful, we should ask three key questions:

1. Is the task defined clearly?
2. Is there an implemented procedure performing the defined task? If not, much difficulty may be lying under a rug somewhere

3. Is there a set of identifiable regularities or constraints from which the implemented procedure gets its power? If not, the procedure may be an ad hoc toy, capable perhaps of superficially impressive performance on carefully selected examples, but incapable of deeply impressive performance and incapable of helping you to solve any other problem.

To determine if an application of AI is successful, you need to ask additional questions:

1. Does the application solve a real problem?
2. Does the application open up a new opportunity?

- **Semantic Nets and Description Matching:**

— Semantic Nets:

A representation that sees both direct and indirect service throughout artificial intelligence.

1. Good Representations are the key to Good Problem Solving:

A representation is a set of conventions about how to describe a class of things.

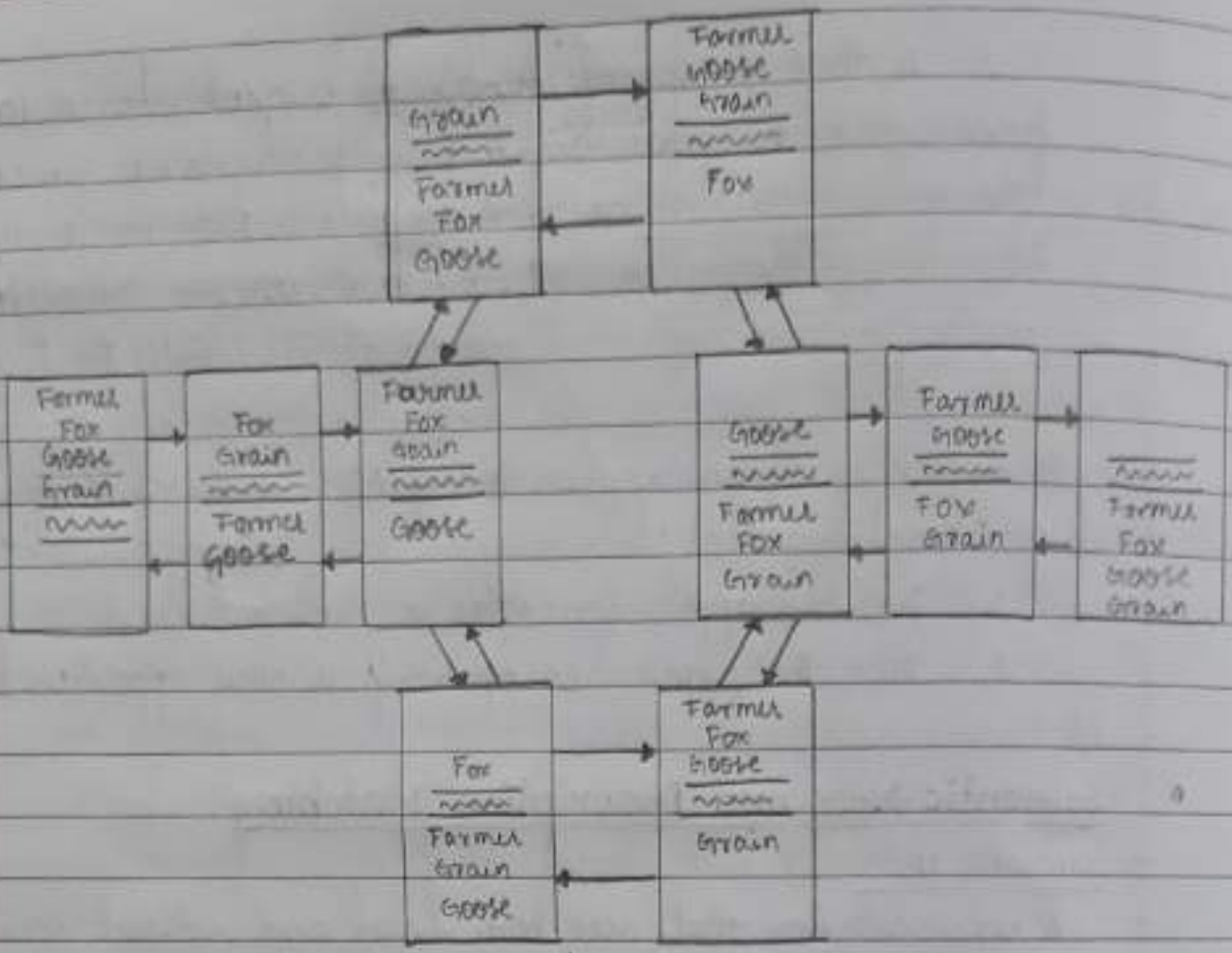
A description makes use of the conventions of a representation to describe a particular thing.

Finding the appropriate representation is a major part of problem solving.

Representation Principle: Once a problem is described using an appropriate representation, the problem is almost solved.

Example: The Farmer, Fox, Goose and Grain:

A farmer wants to move himself, a fox, a goose and some grains across a river but the boat is so tiny that he can take only one of his possessions across any trip. Unfortunately, an unattended fox will eat a goose and an unattended goose will eat grain, so the farmer must not leave the fox alone with the goose or the goose alone with the grain.



2. Good Representations Support Explicit, Constraint-Exposing Description:

Node-and-link representation for 'The Farmer, Fox, Goose and Grain' problem is that it makes the important objects and relations explicit.

The representation also is good because it exposes the natural constraints inherent in the problem. Some transitions are possible, others are impossible. A transition is possible if there is a link, otherwise it is impossible.

- Good Representations make the important objects and relations explicit: You can see what is going on in a glance
- They expose natural constraints: You can express the way one object or relation influences another.
- They bring objects and relations together: You can see all you need to see at one time, as if through a window.
- They suppress irrelevant detail: You can keep rarely used details out of sight, but still get to them when necessary.

- They are transparent: You can understand what is being said
- They are complete: You can say all that needs to be said
- They are concise: You can say what you need to say efficiently
- They are fast: You can store and retrieve information rapidly
- They are computable: You can create them with an existing procedure

3. A representation has four Fundamental Parts:
   - A lexical part that determines which symbols are allowed in the representation's vocabulary (determines that nodes and links are involved)
   - A structural part that describes constraints on how the symbols can be arranged (specifies that links connect node pairs)
   - A procedural part that specifies access procedures that enable you to create descriptions, to modify them and to answer questions using them.
   - A semantic part that establishes a way of associating meaning with the descriptions. (establishes that nodes correspond to arrangements of the farmer and his possessions and links correspond to river traversals)

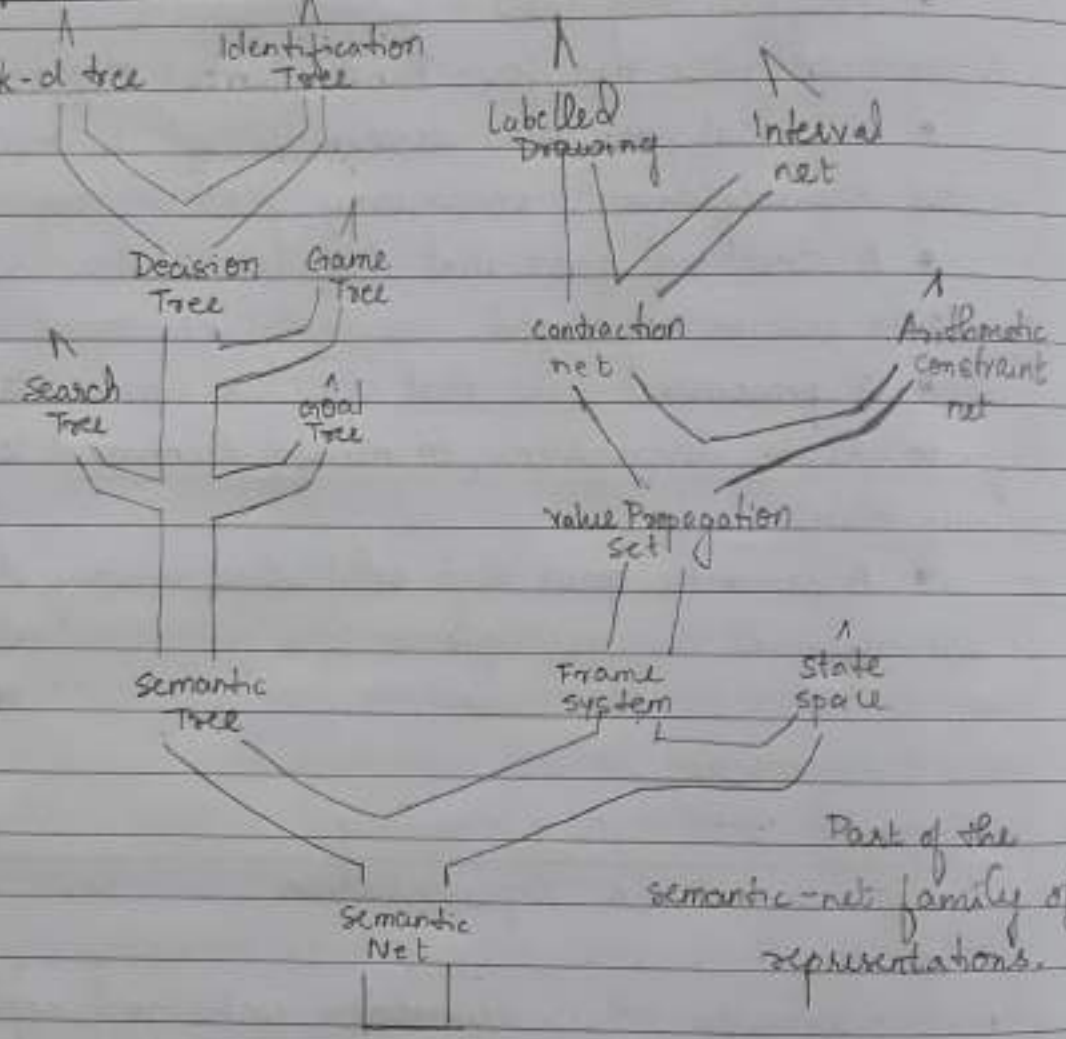
4. Semantic Nets convey Meaning:

A semantic net is a representation
in which
- lexically, there are nodes, links and application-specific link labels
- structurally, each link connects a tail node to a head node
- semantically, the nodes and links denote application-specific entities

with constructors that
- construct a node
- construct a link, given a link label and two nodes to be connected.

with readers that
- produce a list of all links departing from a given node.

- produce a list of all links arriving at a given node
- produce a tail node, given a link
- produce a head node, given a link
- produce a link label, given a link.



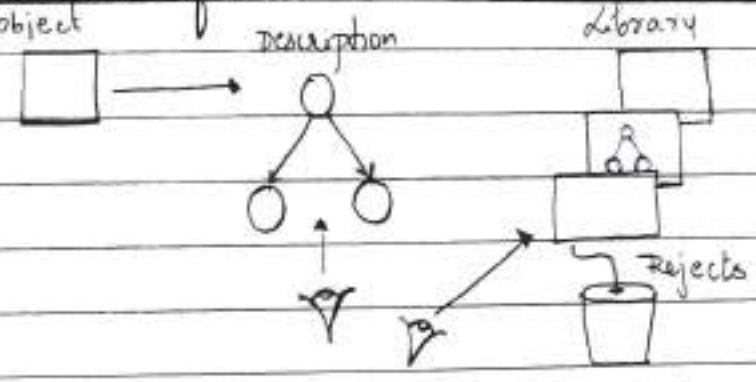
- Describe and Match Method:

The basic idea behind the describe and match method is that you can identify an object by first describing it and then searching for a matching description in a description library.
(Half-english and half-program is called Procedural English)

Describe and Match method expressed in procedural English:

To identify an object using describe and match,
- Describe the object using a suitable representation.
- Match the object description against library descriptions until there is a satisfactory match or there are no more

library descriptions.
- If you find a satisfactory match, announce it, otherwise, announce failure



The design and match paradigm

1. Feature based Object Identification Illustrates Describe and Match:

Feature-based object identifiers consist of a feature extractor and a feature evaluator. The feature extractor measures simple characteristics such as an object's area. Values obtained by the feature extractor become the coordinates of a feature point in feature space, a multidimentional space in which there is one dimension for each feature measured. To identify an unknown object, you compare the distances between its feature point and the feature points of various idealized objects. The most likely identity of the unknown object is determined by the smallest distance.

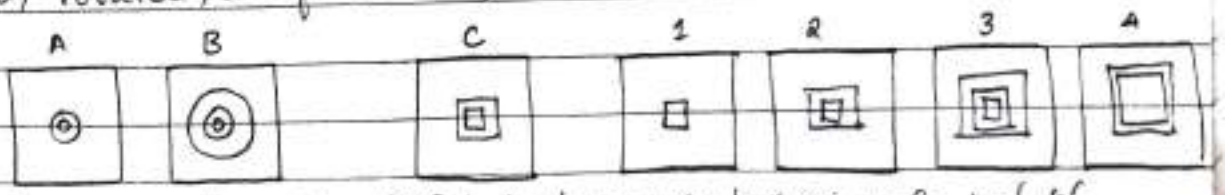
2. The Describe and Match Method and Analogy problems:

* Geometric Analogy Rules Describe Object Relations and object Transformations:

Analogy uses two part rules.

- One rule part describes how the objects are arranged in the source and destination figures. (right of, left of, inside of etc.)
- The other rule part describes how the objects in the source figure are transformed into objects in destination figure. (scaled, rotated, or reflected etc)

Ex:

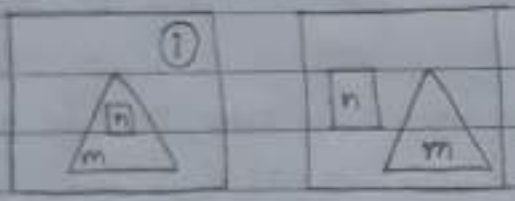


Select an answer figure X, such that A is to B as C is to X gives the best fit

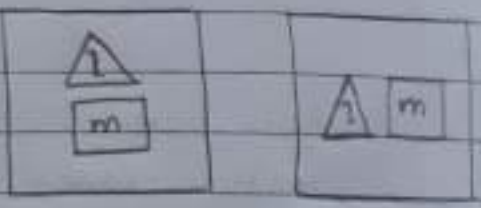
One way to start is to describe rules that explain how A becomes B and how C becomes each of the answer figures. Then, you can match the rule that explains how A becomes B to each rule that explains how C becomes the answer. The best match between rules identifies the best answer. This describe and match paradigm can be used to solve analogy problem.

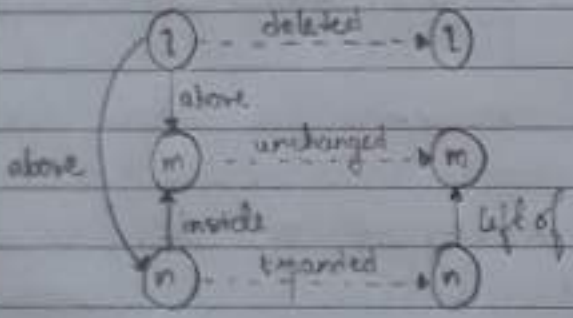
Example: Geometric Analogy

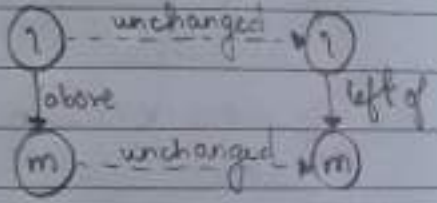
a)



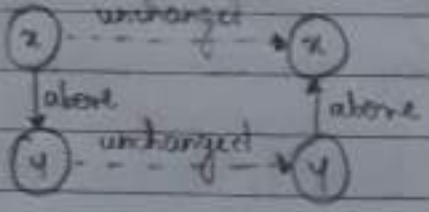
b)







c)



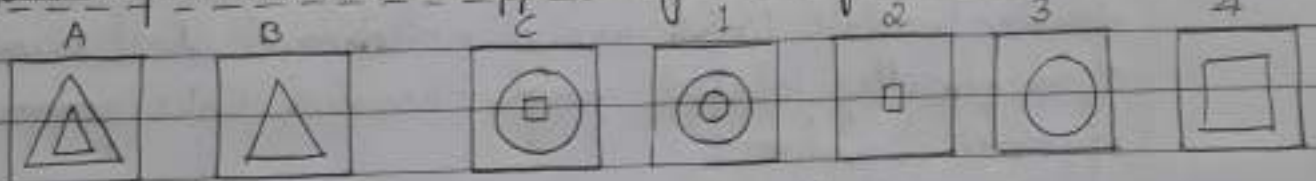
A geometric analogy net is a representation
That is a semantic net
In which

- The nodes denote dots, circles, triangles, squares, rectangles and other geometric objects
- Some links denote relations among figure objects, specifically inside, above and to the left of
- Other links describe how figure objects are transformed. The possibilities are addition, deletion, expansion, contraction, rotation and reflection, and combinations of these operations.

* Ambiguity complicates Matching:
When all the objects in the source and destination figures have distant shapes it is easy to decide how to form the rule describing the transformation.
But in the mentioned example here an ambiguous change is made. Thus in general, for each source and destination pair many rules are possible and for each rule, there may be many ways to match it against another rule.

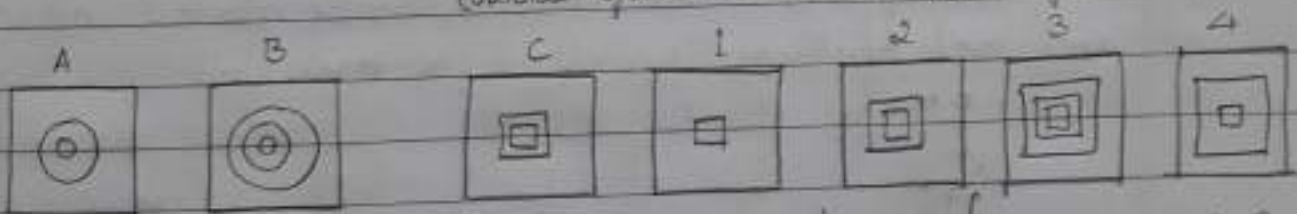
An ambiguous change. Here, the large triangle may have been deleted or the small one may have been deleted and the large one shrunk.

* Good representation supports good performance:

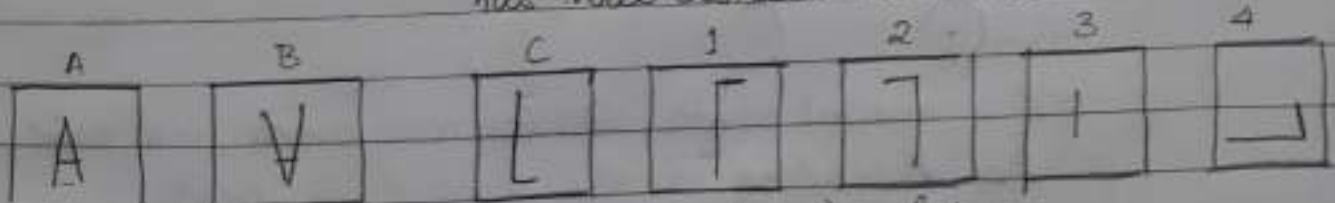


A to B: inside object is deleted

C to 3: is correct
C to 4 is possible if 3 was not present (outside object is deleted and inside object is expanded)



C to 3: is the only correct answer as 3 has three elements like in B.



A to B: rotation
or A to B: reflection

C to 2: (preferred) rotation
C to 1: reflection

3 The Describe and Match Method and Recognition of Abstractions:

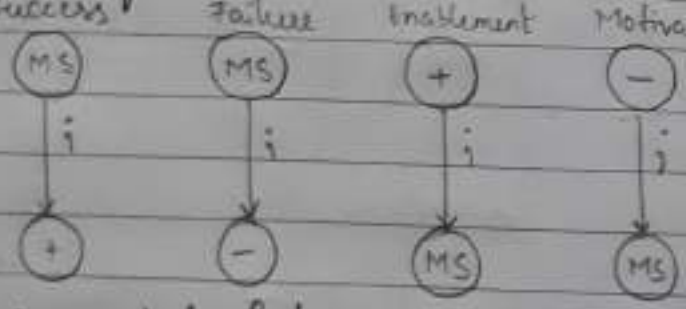
* story Plots can be viewed as combinations of Mental states and Events:

To describe plots using a semantic net we require the three types of node : mental states (MS); positive events (+), negative events (-) and three link labels : initiates (i) - meaning that the mental state or event at the tail of an i link leads to the one at the head of the link; terminates (t) - meaning that the mental state or event at the tail turns off the one at the head; corefers (c) - meaning that the mental state or event at the tail refers to the same mental state or event as the one at the head. Links labeled with c have two heads: double-headed links are a notational short hand for pairs of identically labeled single-headed links pointing in opposite direction.

With three node types and three link labels:

$3 \times 3 \times 3 = 27$ node-link-node combinations

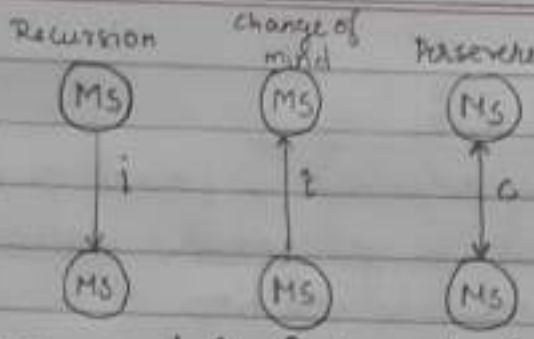
Out of 27, 15 are natural, easily stated interpretation and each of these is called a base unit.



combinations in which mental states and events initiate one another

If a mental state initiates an event, we have success or failure depending on the sign of the event.

If an event initiates a mental state, we witness enablement or motivation depending on the sign of the event.

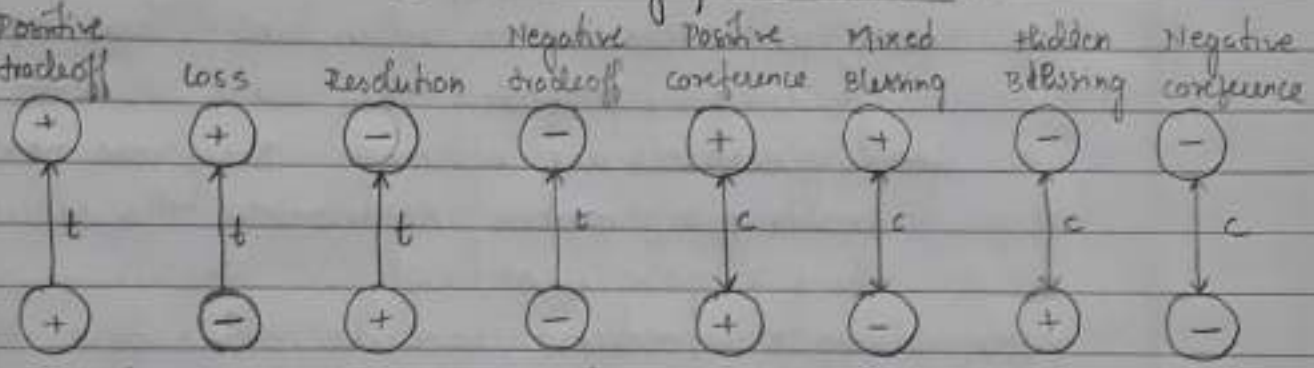


Mental states joined by initiate, terminate or conjer links

If one mental state initiates another, we say that recursion has occured.

If one mental state terminates another, we have a change of mind.

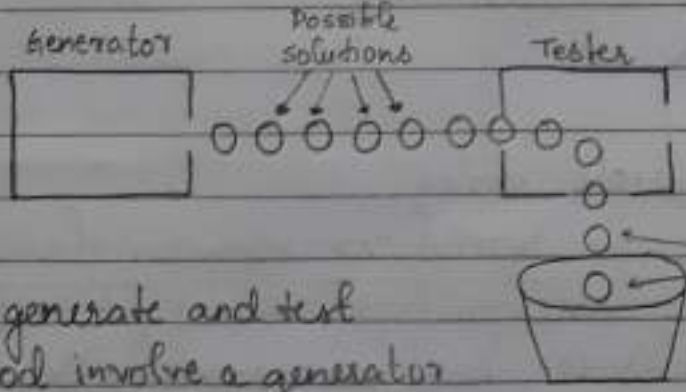
If a mental state persists over a period of time, the individual involves is exhibiting perseverence.



Positive events and negative events joined by terminate or coreference links

Base units often overlap producing recognizable aggregates called composite units. Together, base units and composite units constitute abstraction units.

- **The Generate and Test Method**:



the generate and test method involve a generator and a tester.

Generator
- enumerates possible solutions

Tester
- evaluates each proposed solution either accepting or rejecting that solution

To perform generate and test,
- Until a satisfactory solution is found or no more candidate solutions can be generated,
  - Generate a candidate solution
  - Test the candidate solution
- If an acceptable solution is found, announce it, otherwise announce failure

A good generator has three properties:
1. Good generators are complete: They eventually produce all possible solutions
2. Good generators are nonredundant: They never compromise efficiency by proposing the same solution twice.
3. Good generators are informed: They use possibility limiting information, restricting the solutions that they propose accordingly.

Informability is important, because otherwise there are often too many solutions to go through.

- **The Means-Ends Analysis Method:**

The state of a system is a description that is sufficient to determine the future. In a state space, each node denotes a state and each link denotes a possible one-step transition from one state to another.

A state space is a representation
That is a semantic net
In which
- The nodes denote states
- The links denote transitions between states

The current state corresponds to where you are, the goal state corresponds to where you want to be, and the problem is to find a sequence of transitions that leads from the

initial state to the goal state.

The purpose of means-end analysis is to identify a procedure that causes a transition from the current state to the goal state, or at least to an intermediate state that is closer to the goal state.

To perform means-end analysis,
- Until the goal is reached or no more procedures are available,
  - Describe the current state, the goal state, and the difference between the two.
  - Use the difference between the current state and goal state, possibly with the description of the current state or goal state, to select a promising procedure.
  - Use the promising procedure and update the current state.
- If the goal is reached, announce success; otherwise, announce failure.

1. Difference-Procedure Tables often Determine the Means:

Example: A travel situation in which the problem is to find a way to get from one city to another. One traveler's preferences might link the preferred transportation procedure to the difference between states, described in terms of the distance between the cities involved, via the following difference-procedure table:

Distance	Airplane	Train	car
More than 300 miles	✓		
Between 100 and 300 miles		✓	
Less than 100 miles			✓

Thus, the difference-procedure table determines generally what to do, leaving descriptions of the current state and destination state with no purpose other than to specify the origin and destination for the appropriate procedure.

- **The Problem-Reduction Method:**

Sometimes it is possible to convert difficult goals into one or more easier-to-achieve subgoals. Each subgoal in turn may be divided still more finely into one or more lower-level subgoals.

When using the problem-reduction method, you generally recognize goals and convert them into appropriate subgoals. When so used, problem reduction is often called, equivalently, goal reduction.

1. **Moving Blocks Illustrates Problem Reduction:**

The MOVER procedure

It obeys commands like:

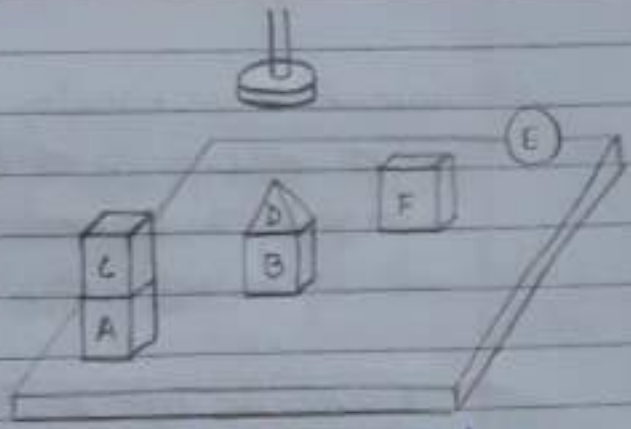
PUT <Block name> on <another block name>

To obey, MOVER plans a motion sequence for a one-handed robot that picks up only one block at a time. MOVER consists of procedures that reduce given problems to simpler problems, thus engaging in what is called problem reduction.

The names for these procedures are mnemonics for the problems that the procedures reduce.

- PUT-ON arranges to place one block on top of another block.
- GET-SPACE finds space on the top of a target block for a traveling block.
- MAKE-SPACE helps GET-SPACE, when necessary, by moving obstructions until there is enough room for a traveling block.
- GRASP grasps blocks.
- CLEAR-TOP does the top clearing.
- GET-RID-OF gets rid of an obstructing object by putting it on the table.
- UNGRASP makes the robot's hand let go of whatever it is holding.
- MOVE moves objects by moving the robot hand.

Algorithm:



Put block A on block B
- Grasp D
- Move D to somewhere on the table
- Ungrasp D
- Grasp C
- Move C to somewhere on the table
- Ungrasp C
- Grasp A
- Move A on B
- Ungrasp A

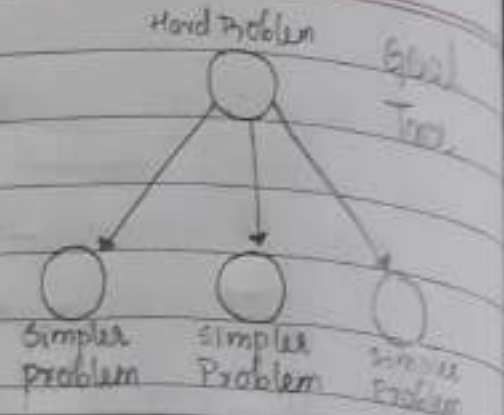
- First, PUT-ON asks GET-SPACE to identify a place for A on top of B.
- GET-SPACE appeals to MAKE-SPACE as D is on top of B.
- MAKE-SPACE asks GET-RID-OF to help by getting rid of block D
- Using PUT-ON, D is placed on the table.
- Now PUT-ON asks GRASP to grasp block A. But GRASP realises that block C is on A
- GRASP asks CLEAR-TOP for help which inturn asks help from GET-RID-OF.
- Then GET-RID-OF places block C on the table using PUT-ON.
- Now GRASP can do its job and PUT-ON can ask MOVE to move A on top of B.
- Finally PUT-ON asks UNGRASP to let go of block A.

2. The key idea in Problem Reduction is to Explore a Goal Tree.

A semantic tree is a semantic net with special links called branches each of which connects two nodes.

A goal tree, is a semantic tree in which nodes represent goals and branches indicate how you can achieve goals by solving one or more subgoals. Each node's children correspond to

immediate subgoals; each node's parent corresponds to the immediate supergoal. The top node, the one with no parent is the root goal.



A semantic tree is a representation
That is a semantic net
In which
- Certain links are called branches. Each branch connects two nodes; the head node is called the parent node and the tail node is called the child node.
- One node has no parent; it is called the root node. Other nodes have exactly one parent.
- Some nodes have no children; they are called leaf nodes
- When two nodes are connected to each other by a chain of two or more branches, one is said to be the ancestor; the other is said to be the descendant.

With constructors that
- Connect a parent node to a child node with a branch link.

With readers that
- Produce a list of a given node's children.
- Produce a given node's parent.

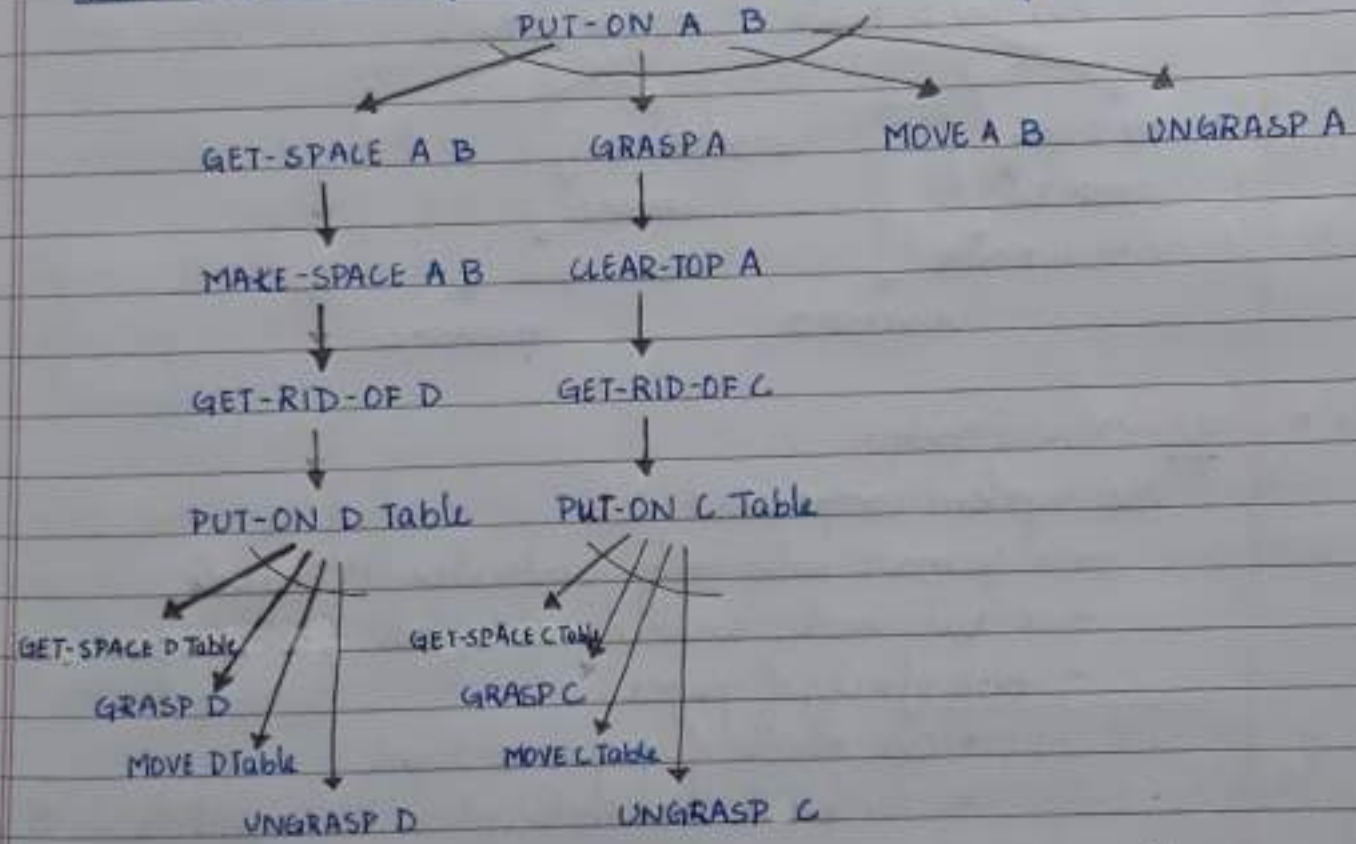
Goals that are satisfied only when all of their immediate subgoals are satisfied are called And goals. The corresponding nodes are called And nodes and are marked by placing arcs on their branches.

Goals that are satisfied when any of their immediate subgoals are satisfied are called Or goals. The corresponding, unmarked nodes are called Or nodes.

Goals that are satisfied directly without reference to any subgoals are called leaf goals and the corresponding nodes are called leaf nodes.

Hence Goal trees are often known as And-Or Trees.

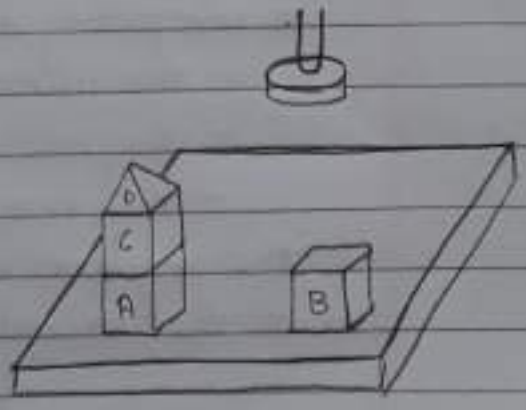
Goal tree for the previous example of moving block A on block B



Example 2:

To put block A on block B

- Grasp D
- Move D on table
- Ungrasp D
- Grasp C
- Move C on table
- Ungrasp C
- Grasp A
- Move A on B
- Ungrasp A



- **SLE: Chinese Room:**

The system comprises:

- a human who only understands English
- a rule book, written in English
- two stacks of paper
  - One stack of paper is blank
  - The other has indecipherable symbols on them

In computing terms:

- the human is the CPU
- the rule book is the program
- the two stacks of paper are storage devices.

The system is housed in a room that is totally sealed with the exception of a small opening.

Process:

- The human sits inside the room waiting for pieces of paper to be pushed through the opening.
- The pieces of paper have indecipherable symbols written upon them.
- The human has the task of matching the symbols from the outside with the rule book.
- Once the symbol has been found the instructions in the rule book are followed.
  - may involve writing new symbols on blank pieces of paper
  - or looking up symbols in the stack of supplied symbols.
- Eventually, the human will write some symbols onto one of the blank pieces of paper and pass these out through the opening.

Summary:

Simple rule processing system is employed but in which the rule processor happens to be intelligent but has no understanding of the rules. The set of rules might be very large but this is philosophy and so the practical issues are ignored.

Searle's Claim:

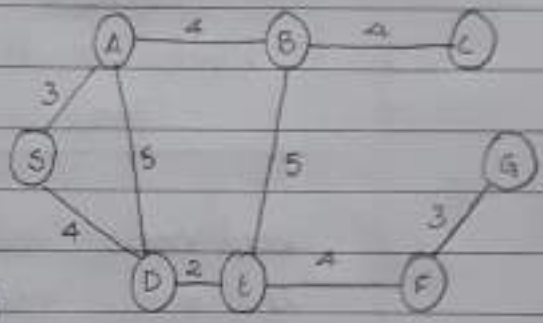
- We have a system that is capable of passing the Turing Test (systems that act like humans) and is therefore intelligent according to Turing.
- But the system does not understand chinese as it just comprises a rule book and stacks of paper which also does not understand chinese.
- Therefore, running the right program does not necessarily generate understanding.

UNIT - 2

Search and Rule Based Systems

- ## Nets and Basic Search :

Example: City S to City G



To find an appropriate path through the highway map, we need to consider two different costs:

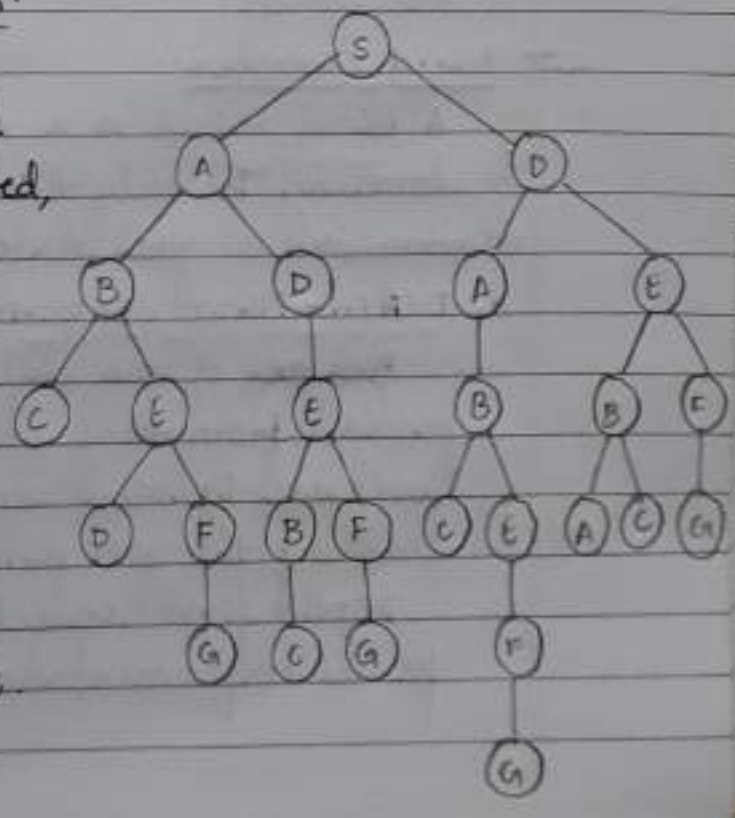
1. The computation cost expended when finding a path.
2. The travel cost expended when traversing the path.

Finding a really good path is worth a lot of search time. On the other hand, if we need to make the trip only once and if its hard to find any path, then you may be content as soon as you find some path, even though you could find a better path with more work.

1. Net Search is really Tree Search:

Most obvious way to find a solution is to look at all possible paths. With looping paths eliminated, you can arrange all possible paths from the start node in a search tree, a special kind of semantic tree in which each node denotes a path.

Each child node denotes a path that is a one-city extension of the path denoted by its parent.



A search tree is a representation
That is a semantic tree
In which
- Nodes denote paths
- Branches connect paths to one-step path extensions
With writers that
- Connect a path to a path description
With readers that
- Produce a path's description.

Root node: node with no parent
Leaf node: node with no children
Branching factor: If a node has b children, it is said to have a branching factor of b.
Partial Path: Path that doesnot reach the goal
Complete Path: Path that reaches the goal.
Expanding: determining the children of a node. Nodes are said to be open until they are expanded, whereupon they become closed.

Blind Search:

A blind search is a search that has no information about its domain. The only thing that a blind search can do is distinguish a goal state from a non-goal state.

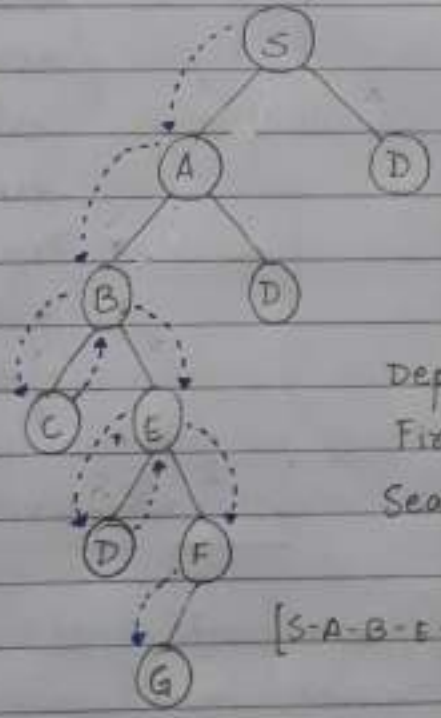
a. DEPTH-FIRST SEARCH:

Pick one of the children at every node visited and work forward from that child. Other alternatives at the same level are ignored completely, as long as there is hope of reaching the goal using the original choice.

Depth first search uses a stack data structure for its implementation.

To conduct a depth-first search
- Form a one-element queue consisting of a zero-length path that consists only the root node.
- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue, create new paths by extending the first path at all the neighbors of the terminal node.
  - Reject all new paths with loops.
  - Add new paths, if any, to the front of the queue.
- If the goal is found, announce success, otherwise announce failure.

One alternative is selected and pursued at each node until the goal is reached or a node is reached where the further downward motion is impossible. When further downward motion is impossible, search is restarted at the nearest ancestor node with unexplored children.



Depth First Search

[S-A-B-E-F-G]

Advantage:
- requires less memory
- takes less time to reach the goal

Disadvantage:
- no guarantee of finding the solution
- due to deep down searching it may go to infinite loop

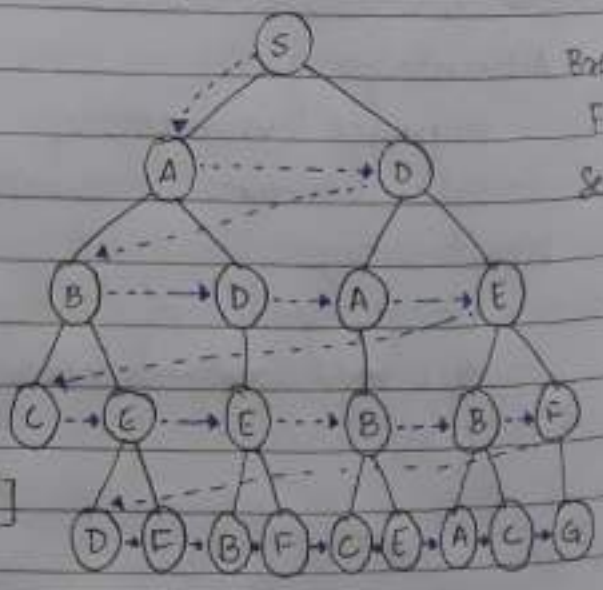
b. BREADTH-FIRST SEARCH

Breadth-first search pushes uniformly into the search tree. It checks all paths of a given length before moving on to any longer paths.

Breadth-first search is implemented using FIFO queue data structure.

To conduct Breadth-first search,
- Form a one-element queue consisting of a zero-length path that contains only the root node.
- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.
  - Reject all new paths with loops.
  - Add the new paths, if any, to the back of the queue.
- If the goal node is found, announce success; otherwise, announce failure.

In breadth-first search, downward motion proceeds level by level until the goal is reached.



Breadth First Search

[S-D-E-F-G]

Advantages:
- Provides a solution if any solution exists.
- If multiple solutions are present, then BFS will provide the minimal solution which requires least number of steps.

Disadvantages:
- Requires a lot of memory
- Takes a lot of time is the goal is away from the root node

- Heuristically Informed Methods

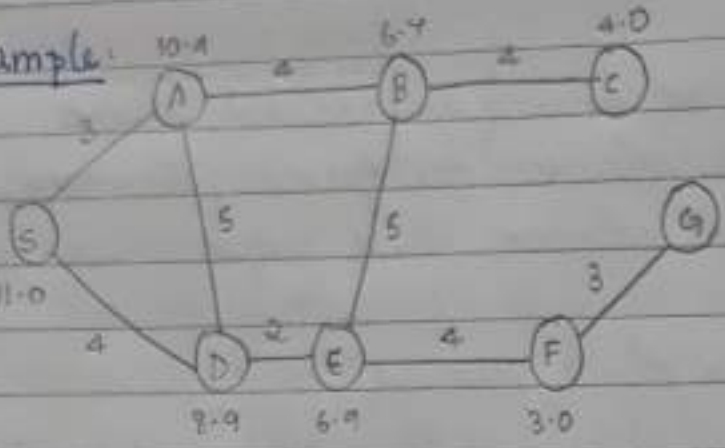
Informed search algorithm contains information about its domain, like how far the goal is, path cost etc. This knowledge helps to find the goal node more efficiently.

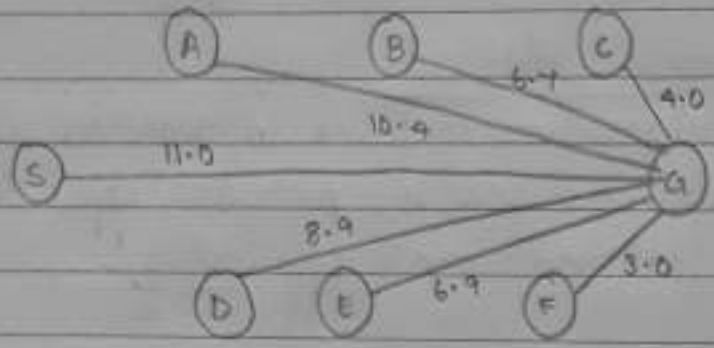
a. HILL CLIMBING:

Hill climbing method continuously moves in the direction of increasing value to find the best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.

To conduct hill-climbing search,
- Form a one-element queue consisting of a zero-length path that contains only the root node.
- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.
  - Reject all new paths with loops.
  - Sort the new paths, if any, by the estimated distances between the terminal nodes and the goal.
  - Add the new paths, if any, to the front of the queue.
- If the goal node is found; announce success, otherwise announce failure.

Example:

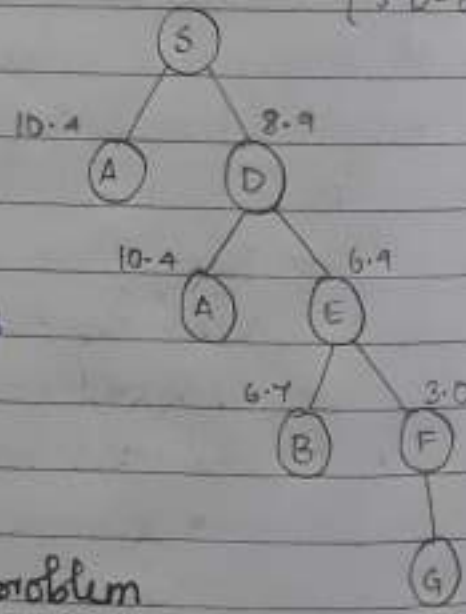




The straight-line distance from each city to the goal shows the heuristic measure of remaining distances

Hill climbing is depth first search with a heuristic measurement that orders the choices as nodes are expanded.

The numbers beside the nodes are straight-line distances from the path-terminating city to the goal city.



When faced with a search problem note that, more knowledge generally leads to reduced search time.

Sometimes knowledge reduces search time by enabling you to restate a problem in terms of a smaller, more easily searched net.

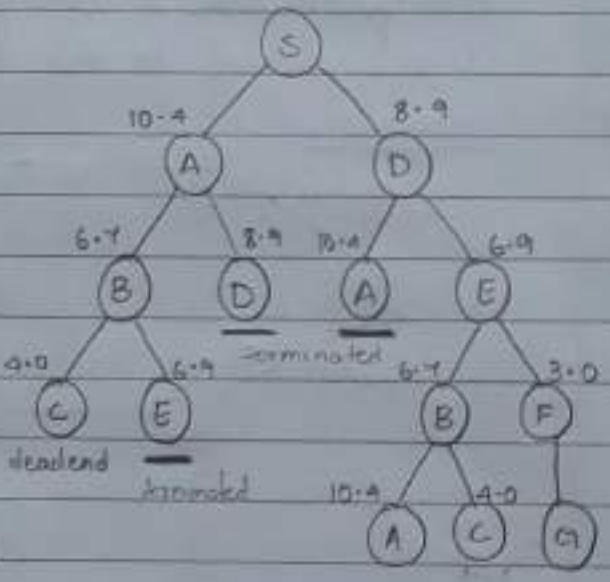
b. BEAM SEARCH:

Beam search is similar to BFS as it progresses level by level. Beam search moves downward only through the best w node at each level; the other nodes are ignored. whenever beam search is used, there are only w nodes under consideration at any depth.

Example:

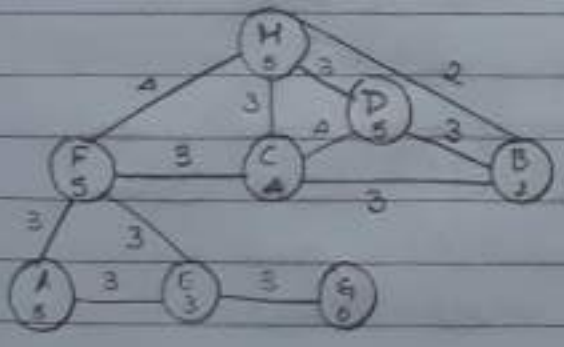
The numbers beside the nodes are straight-line distances to the goal node. Investigation spreads through the search tree level by level.

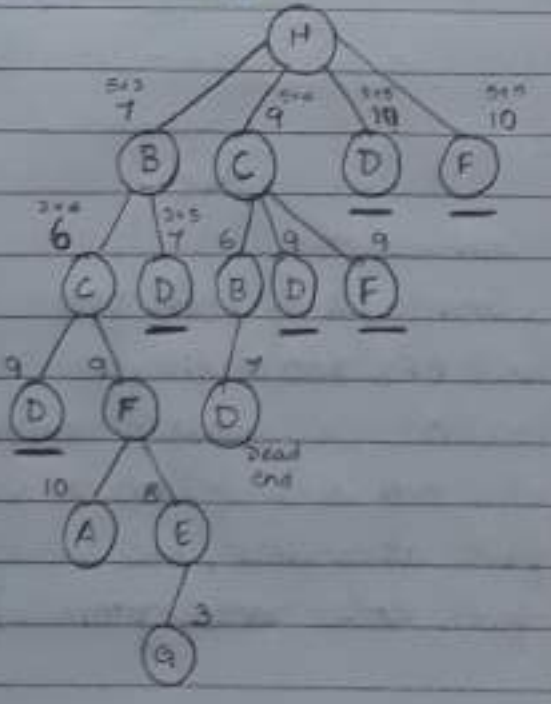
But only the best w nodes are expanded, here w = 2. The remaining paths, those shown terminated by underbars are rejected.



Example 2:

Considering w = 2





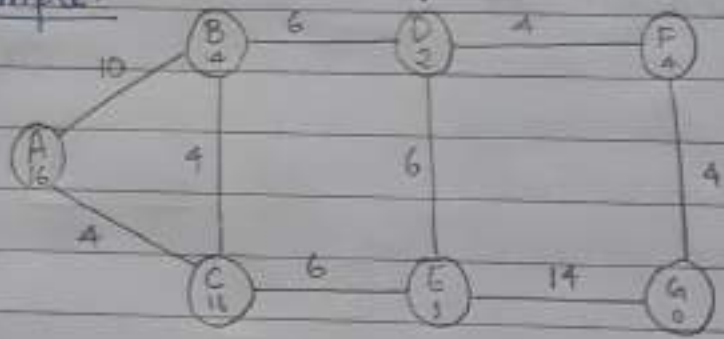
- **SLE: Best-First Search**

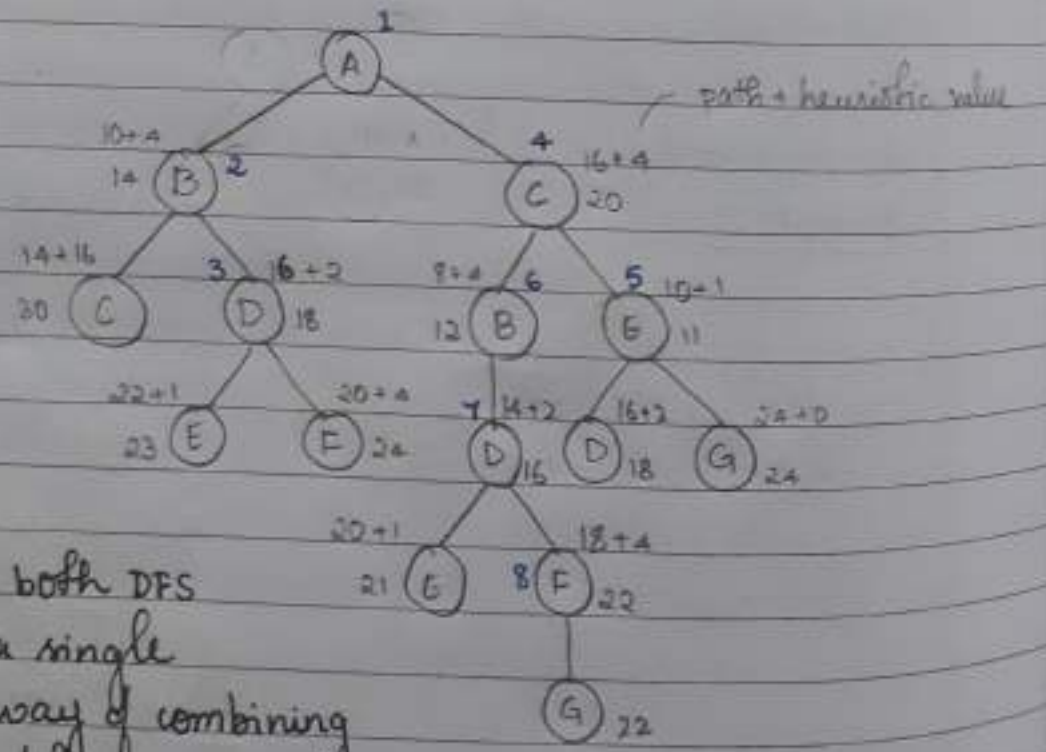
c. BEST-FIRST SEARCH: / A* ALGORITHM

In best first search, forward motion is from the best open node so far, no matter where that node is in the partially developed tree.

The paths found by best-first search are likely to be shorter than those found with other methods, because best-first search always moves forward from the node that seems closest to the goal node.

Example:





Combines the advantages of both DFS and BFS into a single method. One way of combining the two is to follow a single path at a time, but switch paths whenever some competing path looks more promising than the current one does.

NOTE:

1. DFS is good when unproductive partial paths are never too long
2. BFS is good when the branching factor is never too large.
3. Hill climbing is good when there is a natural measure of distance from each place to the goal and a good path is likely to be among the partial paths that appear to be good at each choice point.
4. Beam search is good when there is natural measure of goal distance and a good path is likely to be among the partial paths that appear to be good at all levels.
5. Best search is good when there is natural measure of goal distance and a good partial path may look like a bad option before more promising partial paths are played out.

- **Optimal Search:**

Here the cost of traversing a path is of primary importance.

- The Best Path:

a. BRITISH MUSEUM:

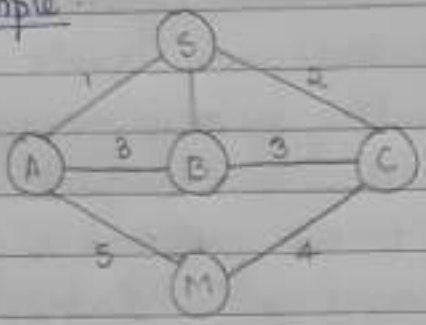
In British museum method, to find the shortest path through a net, we find all possible paths and then select the best one from them.

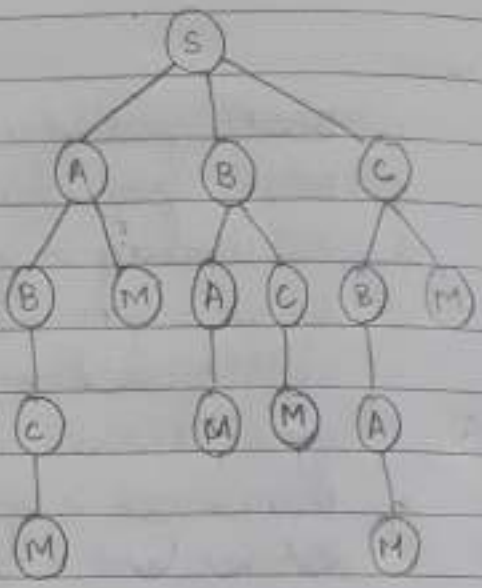
If we wish to find all possible paths, either a depth-first search or a breadth-first search will work but search continues until all the solutions are found.

But the drawback of this method is that if the size of the search trees is large, finding all possible paths is extremely difficult.

This technique is conceptual, not a practical technique as the number of possibilities is enormous.

Example:



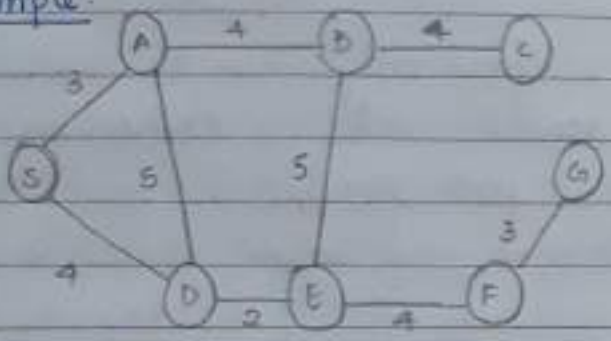


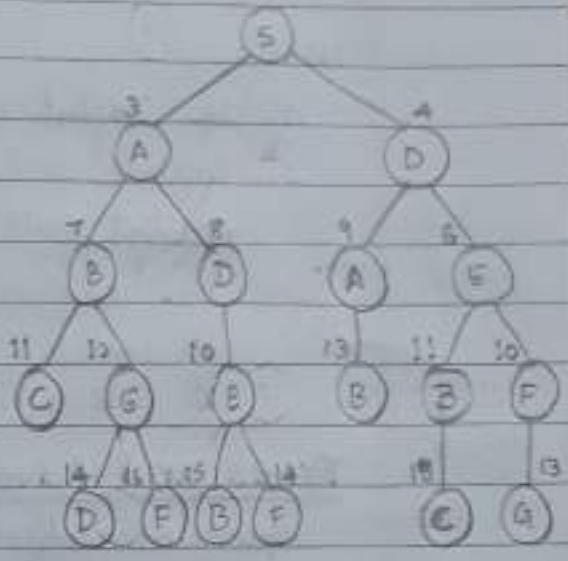
b. BRANCH AND BOUND:

Branch and bound always keeps track of all partial paths contending for further consideration. The shortest one is extended next level, creating as many new partial paths as there are branches. Next, these new paths are considered along with the remaining old ones: again, the shortest is extended. This process repeats until the goal is reached along some path. Because the shortest path was always the one chosen for extension, the path first reaching the goal is likely to be the optimal path.

To conduct a branch and bound search,
- Form a one-element queue consisting of a zero-length path that contains only the root node.
- Until the first path in the queue terminates at the goal node or the queue is empty,
  - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node.
  - Reject all new paths with loops.
  - Add the remaining new paths, if any, to the queue
  - Sort the entire queue by path length with least cost paths in front.
- If the goal node is found, announce success, otherwise failure

Example:



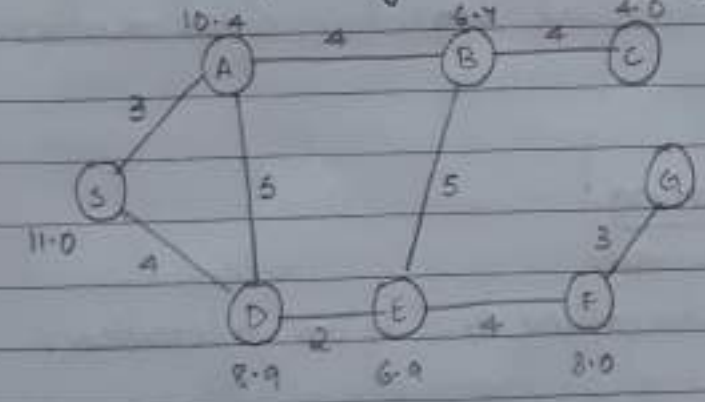


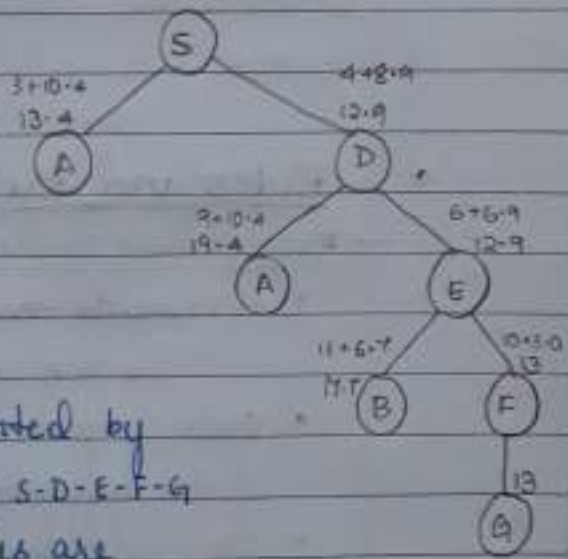
Path :- S-D-E-F-G

The length of the complete path from S to G (S-D-E-F-G) is 13. Only the paths emerging from S-D-E have to be considered, as they may provide a shorter path

Branch-and-bound search can be greatly by using guesses about the remaining distances and also distance already accumulated. The guessed distance added to the definitely known distance already traversed should be good estimate of total path length.

e(total path length) = d(already travelled) + e(distance remaining)





Branch and bound search is augmented by underestimates determines that the path S-D-E-F-G is optimal. The numbers beside the nodes are accumulated distances plus underestimates of distances remaining. Underestimates quickly push up the lengths associated with bad paths. In this example many fewer nodes are expanded than would be expanded with branch and bound search operating without underestimates

NOTE:

1. The British museum is good only when the search tree is small.
2. Branch and bound search works by extending the least-cost partial path until that path reaches the goal. Adding underestimates to branch and bound search improves efficiency.

UNIT 3

- **Trees and Adversarial search:**

Game situations can be represented in trees and you can learn how those trees can be searched so as to make the most promising move.

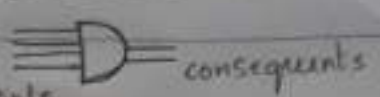
An adversarial search is where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

searches in which two or more players with conflicting goals are trying to explore the same search space for the solution are called adversarial searches, often known as Games.

- **Rules and Rule Chaining:**

— Rule Based Deduction System

- Statements that is something true is an assertion.
  "Stretch has long legs"; "Stretch is a giraffe"
- Each if pattern is a pattern that may match one or more of the assertions in a collection of assertions
- The collection of assertions is called working memory.
- The then pattern specify new assertions to be placed into working memory and the rule-based system is said to be a deduction system.
- Each if pattern is an antecedent and each then pattern is a consequent.



- when the then patterns specify actions rather than assertions then the rule based system is a reaction system
  "Put the item into the bank"
- Forward Chaining is the process of moving from the if patterns to the then patterns using the if patterns to identify appropriate situations for the deduction of a new assertion or the performance of an action
- when an if pattern matches an assertion - the antecedent is satisfied. When all if patterns of a rule is satisfied the rule is triggered when a triggered rule establishes a new assertion or performs an action, it is fired

Example: A Toy Deduction System Identifies Animals

A robot called Robbie decides to build a ZOOKEEPER by creating if-then rule for each kind of animal in the zoo. The consequent side is simple assertion of animal identity and the antecedent side is characteristics sufficient to completely reject all incorrect identifications

Zoo contains seven animals: a cheetah, a tiger, a giraffe, a zebra, an ostrich, a penguin and an albatross

```
Z1: if   ?x has hair                    } determines if a particular
     then ?x is a mammal                } animal is a mammal
Z2: if   ?x gives milk                  }
     then ?x is a mammal

Z3: if   ?x has feathers                } determines if a particular
     then ?x is a bird                  } animal is a bird
Z4: if   ?x flies                       }
         ?x lays eggs                   }
     then ?x is a bird

Z5: if   ?x is a mammal                 } determines if a particular
         ?x eats meat                   } mammal is a carnivore
     then ?x is a carnivore             } (determine by act of dinner)
```

z6: If ?x is a mammal
?x has pointed teeth
?x has claws
?x has forward pointing eyes
then ?x is a carnivore
} determines if a particular mammal is a carnivore (by physical characteristics)

z7: If ?x is a mammal
?x has hoofs
then ?x is an ungulate
z8: If ?x is a mammal
?x chews cud
then ?x is an ungulate
} determines if a particular mammal is an ungulate

z9: If ?x is a carnivore
?x has tawny color
?x has dark spots
then ?x is a cheetah
} determines if a particular carnivore is a cheetah

z10: If ?x is a carnivore
?x has tawny color
?x has black stripes
then ?x is a tiger
} determines if a particular carnivore is a tiger

z11: If ?x is an ungulate
?x has long legs
?x has long neck
?x has tawny color
?x has dark spots
then ?x is a giraffe
} determines if a particular ungulate is a giraffe

z12: If ?x is an ungulate
?x has white color
?x has black stripes
then ?x is a zebra
} determines if a particular ungulate is a zebra

z13: If ?x is a bird
?x does not fly
}

?x has long legs
?x has long neck
?x is black and white
then ?x is an ostrich

} determines if a particular bird is an ostrich

Z14: If ?x is a bird
?x does not fly
?x swims
?x is black and white
then ?x is a penguin

} determines if a particular bird is a penguin

Z15: If ?x is a bird
?x is a good flyer
then ?x is an albatross

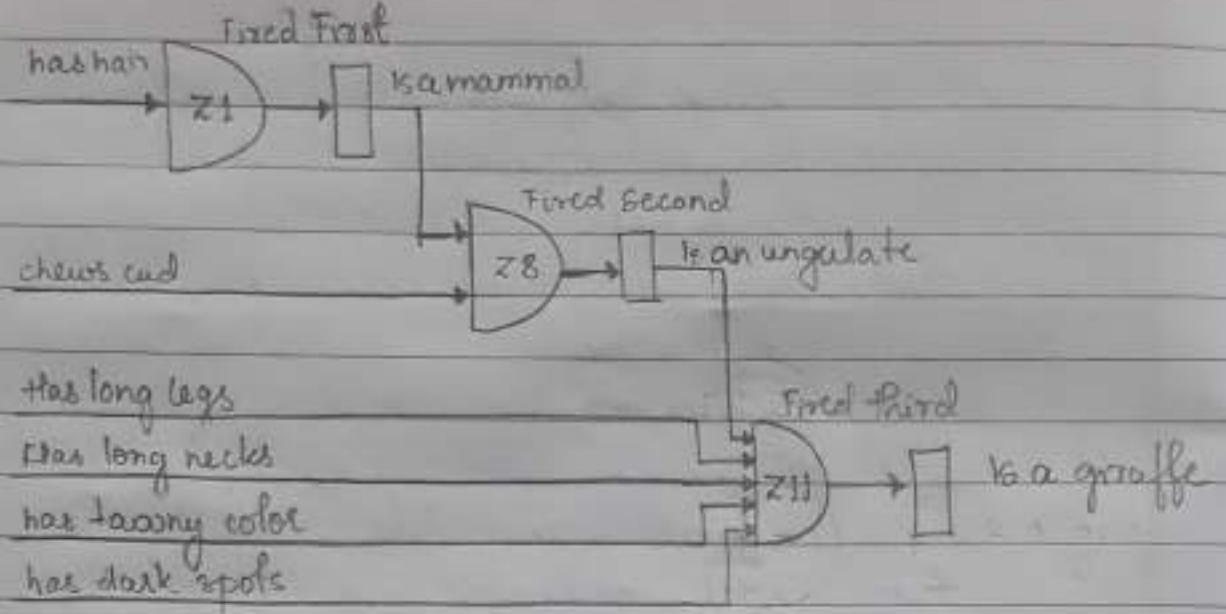
} determines if a particular bird is an albatross

Assertions: Stretch has hair
Stretch chews cud
Stretch has long legs
Stretch has tawny color
Stretch has long neck
Stretch has dark spots

Result: 1. Stretch has hair —rule z1 fires→ Strech is a mammal
2. Stretch is a mammal and chews cud —rule z6 establishes→ strech is an ungulate
3. Stretch is an ungulate, has long legs, long neck, tawny color and dark spots —all the antecedents of rule z11 is satisfied→ strech is a giraffe

In inference nets, the D shaped objects represent rules, whereas vertical bars denote given assertions and vertical boxes denote deduced assertions.

Assertions flow through a series of antecedent - consequent rules from given assertions to conclusion.



- **Backward Chaining**: a rule-based system can form a hypothesis and use the antecedent-consequent rules to work backward toward hypothesis supporting assertions.

Example: Backward Chaining

ZOOKEEPER might form the hypothesis that a given animal, Swifty is a cheetah and then reason out whether that hypothesis is viable.

1. ZOOKEEPER forms the hypothesis that Swifty is a cheetah considers rule Z9.

   If requires Swifty is carnivore
   Swifty has tawny color
   Swifty has dark spots

2. ZOOKEEPER must check whether Swifty is a carnivore
   Two rules may be considered: Z5 and Z6
   considers rule Z5:

   If requires Swifty is a mammal
   Swifty eats meat

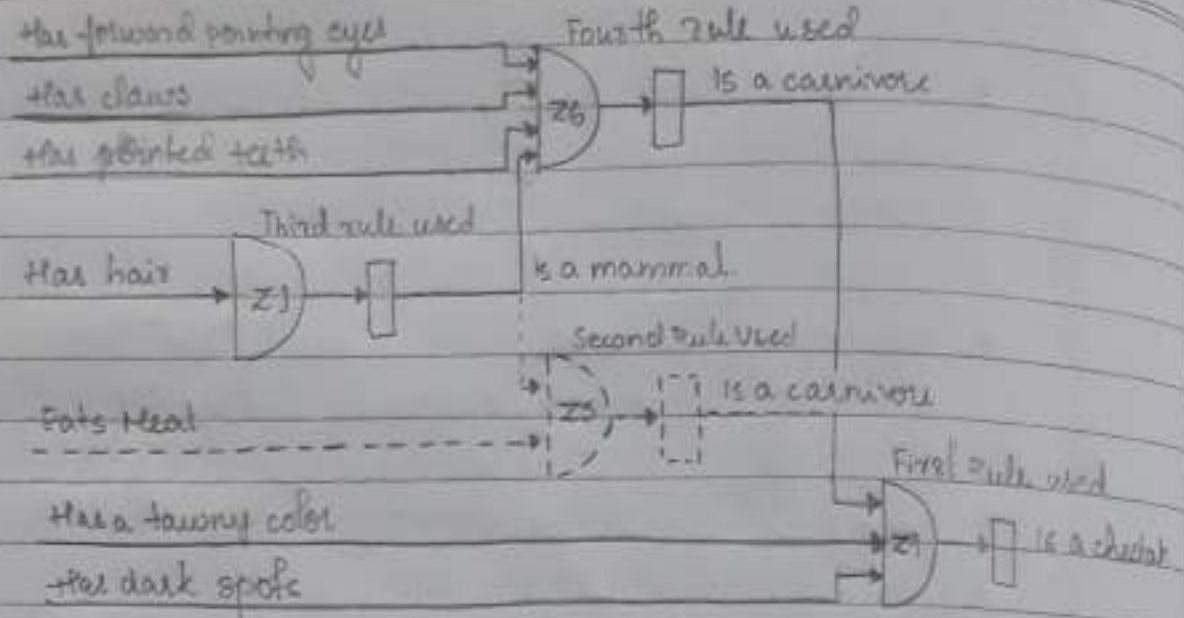
   considers rule Z6:

   If requires Swifty is a mammal
   Swifty has pointed teeth
   Swifty has claws
   Swifty has forward pointing eyes

3. ZOOKEEPER must check whether Swifty is a mammal.
Two rules may be considered z1 and z2
Considering rule z1:
It requires Swifty has hair
4. ZOOKEEPER must check whether Swifty has hair
Assuming that ZOOKEEPER already knows that Swifty has hair
so, Swifty must be a mammal.
Now back to rule z5.
5. ZOOKEEPER must check whether Swifty eats meat.
Assuming thats unknown.
Try to use z6 to establish Swifty is a carnivore.
6. ZOOKEEPER now knows that Swifty is a mammal (rule z1)
It must check whether Swifty has pointed teeth, claws and forward pointing eyes
Assuming all these features are present. Hence,
Swifty is a carnivore.
Now back to rule z9.
7. ZOOKEEPER, now knows that Swifty is carnivore.
It requires Swifty has tawny color
Swifty has dark spots
Assuming ZOOKEEPER knows that Swifty has these features
Thus Rule z9 supports the original hypothesis that
Swifty is a cheetah.

Thus, ZOOKEEPER is able to work backward through the antecedent-consequent rules, using desired conclusions to decide for what assertions it should look.

knowing something about an unknown animal enables identification via backward chaining. Here, the hypothesis that Swifty is cheetah leads to assertions that support that hypothesis.



- The problem determines whether chaining should be forward or backward:
  - If the facts that you have or may establish can lead to a large number of conclusions, but the number of ways to reach the particular conclusion in which you are interested are small, then there is more fan out than fan in, and you should consider back board chaining.
  - If the number of ways to reach the particular conclusion in which you are interested is large, but the number of conclusions that you are likely to reach using the facts is small, then there is more fan in than fan out, and you should consider forward chaining.

When neither fan in nor fan out dominates:
- If you have not yet gathered any facts and you are interested in only whether one of many possible conclusions is true, use backward chaining.
- If you already have in hand all the facts you are ever going to get, and you want to know everything you can conclude from those facts, use forward chaining.

Example:

P0: If (OR 'x? is a noob',
'x? is disenchanted')
then 'x? visits the Oracle'

P1: If 'x? visits the Oracle'
then 'x? picks the One'

P2: If (AND 'x? is a noob',
'x? visits The Oracle')
Then 'The oracle says x? is not The One',
'x? is disenchanted'

P3: If (AND (OR 'x' is disenchanted',
'x? is a noob'),
'The Oracle says x? is not the One')
then 'x? practices Kung Fu'

P4: If (AND 'x? is disenchanted',
'x? loves y?')
then 'x? proclaims love for y?'

P5: If (AND 'The oracle says x? is not The One',
'y? proclaims love for x?'
'y? picks the One')
then 'x? is the One'

Assertions:

A0: Neo is a noob
A1: Trinity is disenchanted
A2: Trinity loves Neo
A3: Trinity proclaims love for Neo

[Neo is the One]

Simulate backward chaining with the hypothesis and draw the goal tree.

1 Hypothesis: Neo is the One
considering Rule P5:
If requires [The Oracle says Neo is not the one
AND [Trinity proclaims love for Neo (A3)
Trinity picks the One

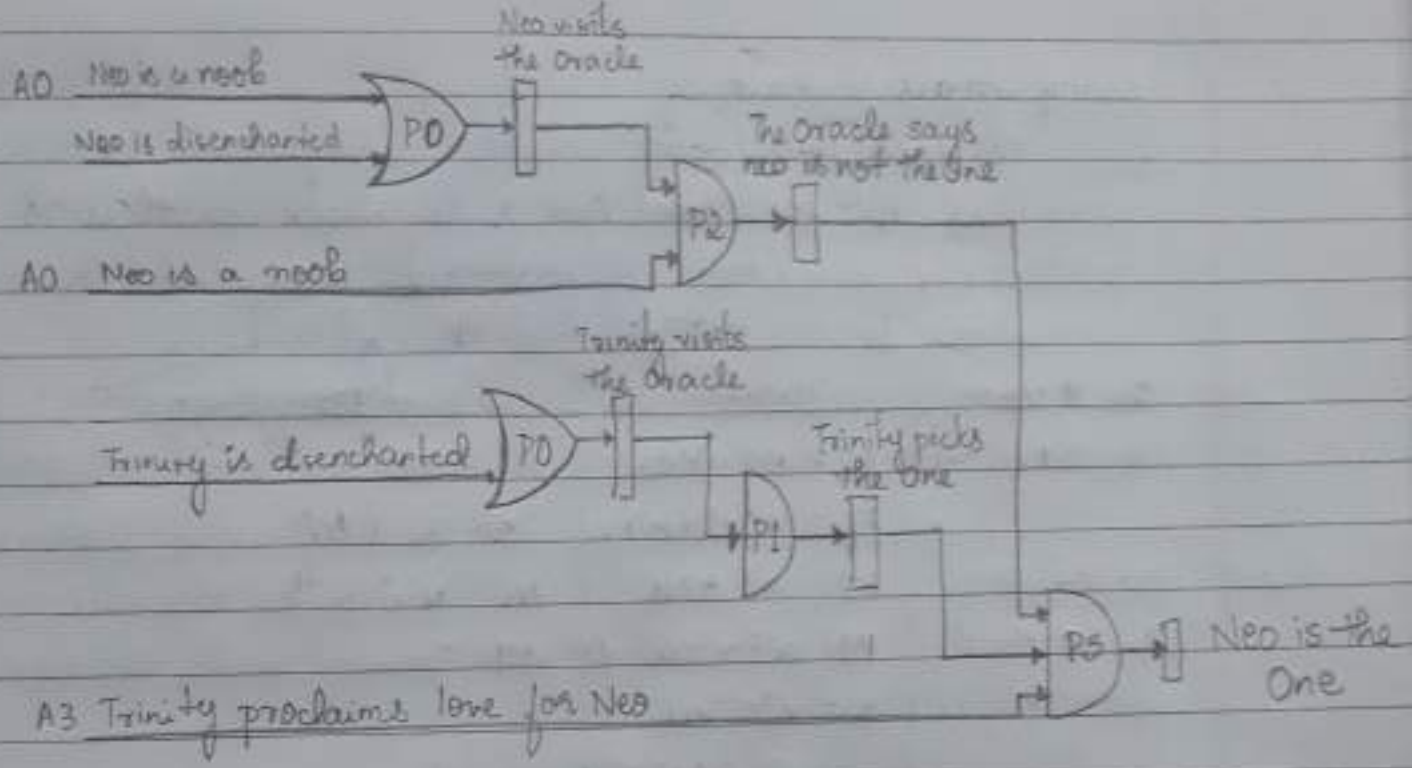
2. To check: 'The Oracle says Neo is not the One'
Considering Rule P2:
If requires AND [Neo is a noob (A0)
Neo visits the Oracle

3. To check: 'Neo visits the Oracle'
Considering Rule P0:
If requires OR [Neo is a noob (A0)
Neo is disenchanted
Thus Neo visits the Oracle
By rule P2: 'The Oracle says Neo is not the One',
'Neo is disenchanted' (new assertion)

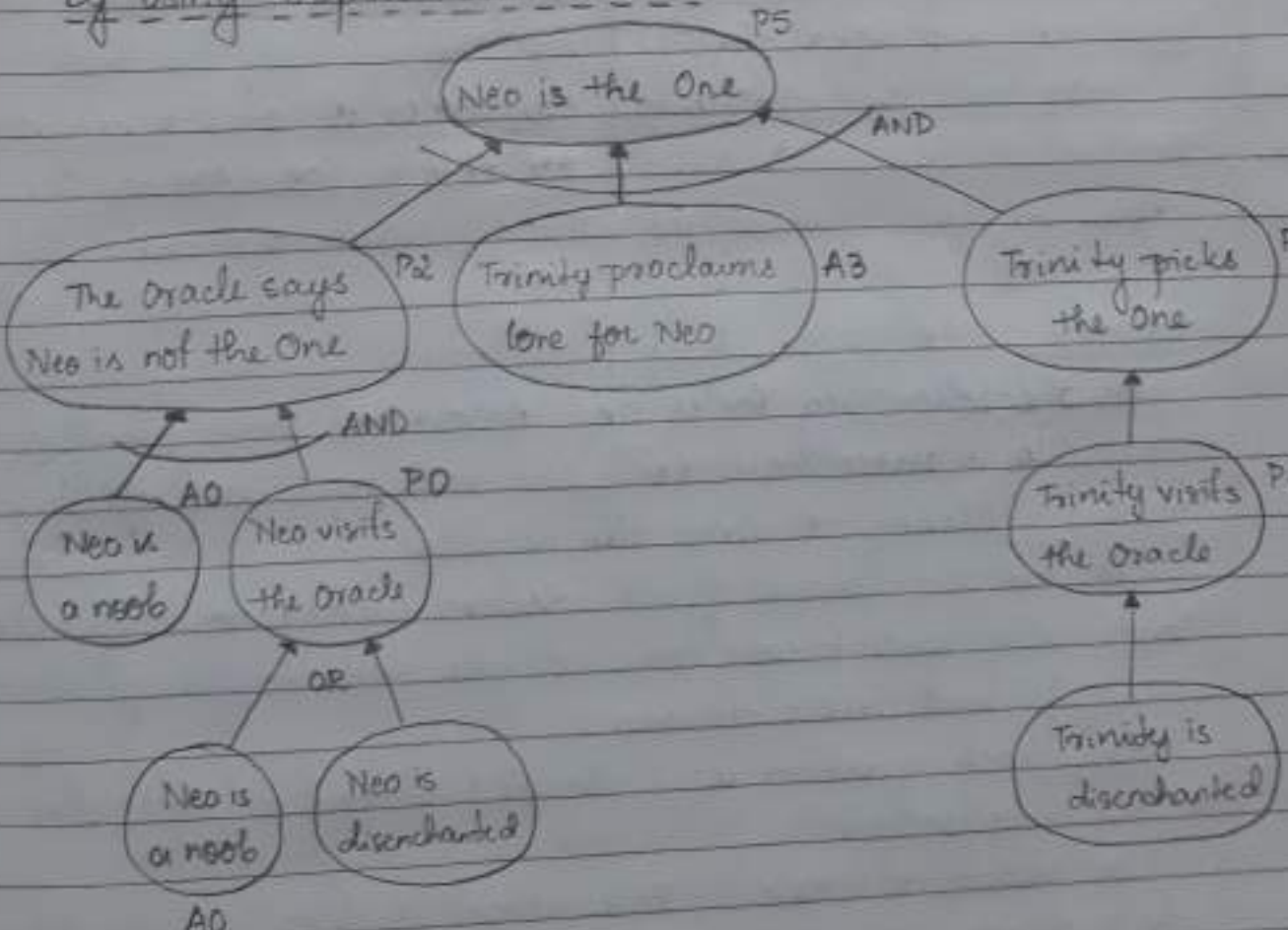
4. To check: 'Trinity picks the One' :(Back to rule P5)
Considering Rule P1:
If requires Trinity visits the Oracle

5. To check: 'Trinity visits the Oracle'
Considering Rule P0
If requires OR [Trinity is a noob
Trinity is disenchanted
By rule P4:
As 'Trinity proclaims love for Neo' (A3)
then Trinity is disenchanted - (New assertion)
and Trinity loves Neo (A2)
Hence 'Trinity visits the Oracle'

6. By rule P1: Trinity picks the One
Hence Rule P5 supports the hypothesis and concludes that 'Neo is the One'



By using Depth First Search:



Using forward chaining:

NOTE: For each rule check whether antecedents match the assertions. For each matched rule check whether it could produce new assertion. If first rule fires, add new assertions to the list till no rule gets fired.

	Matches	Fired	New assertions
1.	P0, P4	P0: x? - Neo	A4: Neo visits the Oracle
		P0: x? - Trinity	A5: Trinity visits the Oracle
2.	P0, P4, P1	P1: x? - Neo	A6: Neo picks the One
		P1: x? - Trinity	A7: Trinity picks the One ✓
		P4: x? - Trinity	
		y? - Neo	
3.	P0, P4, P1, P2	P2: x? - Neo	A8: The Oracle says Neo is not the One
			A9: Neo is disenchanted
		P2: x? - Trinity	A10: The Oracle says Trinity is not the One
4.	P0, P4, P1, P2, P5	P5	"Neo is the One"

Hence by assertions A8, A3, A7 → "Neo is the One"

Example:

For the following Rules and Assertions, prove whether the hypothesis is correct or wrong. Use both Backward chaining and Forward chaining to show the results.

You are the source of all wisdom among your friends. That's why your friends come to you with tough questions about life and career decisions.

One such question is: "Should I become a rock star or pursue another job?"

You decide to write a Rule based system to help figure out which of your friends should indeed become rock stars.

Prove the hypothesis "Krish should become a rock star."

Assertions:

A0: Alison is in A1
A1: Brad is in A1
A2: Alison is musically gifted
A3: Krish is a rebel
A4: Krish takes dance lessons.

Rules:

P0: If 'x? is in A1'
then 'x? is charismatic'

P1: If (OR 'x? is nimble',
'x? takes dance lessons')
then 'x? can dance like Prabhudeva'

P2: If (OR 'x? is a rebel',
(AND 'x? has money',
'x? chooses style over comfort'))
then 'x? wears leather'

P3: If (AND 'x? is related to Lata Mangeshkar'
(NOT 'x? goes to NIE'))
then 'x? chooses style over comfort'

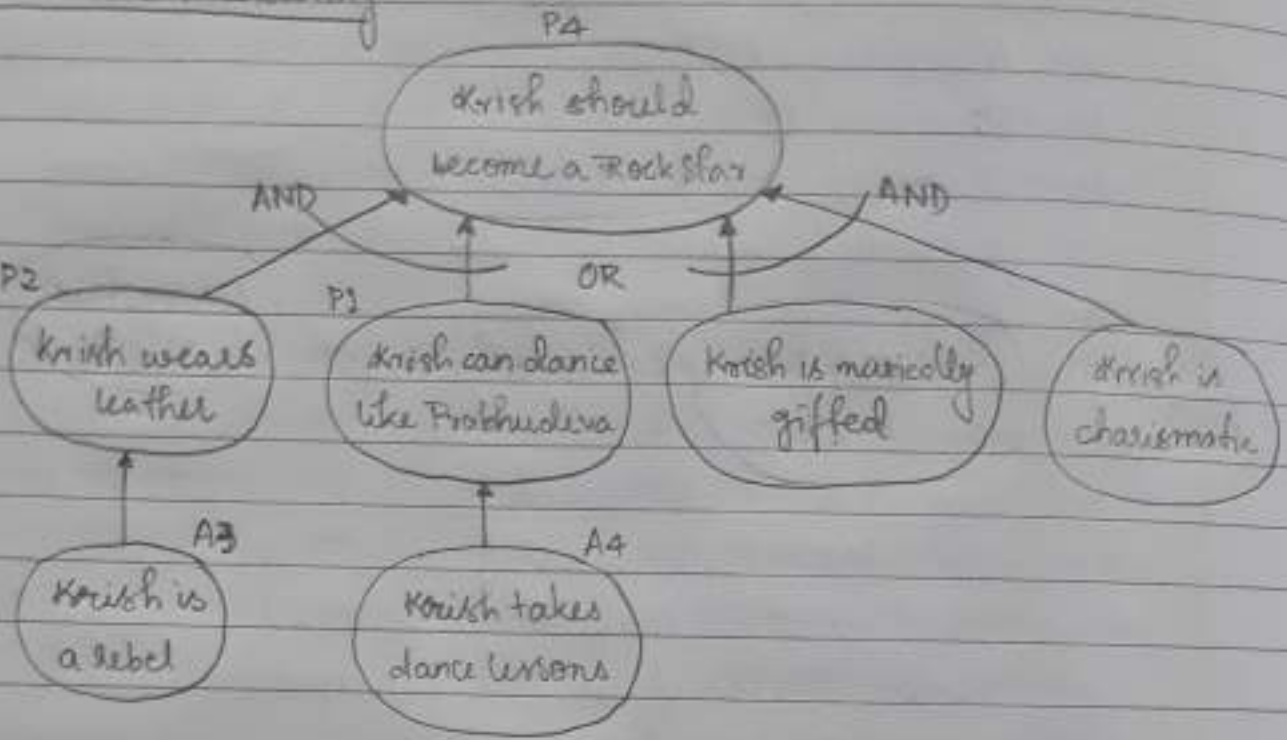
P4: If (OR (AND 'x? wears leather',
'x? can dance like Prabhudeva'),
(AND 'x? is musically gifted',
'x? is charismatic'))
then 'x? should become a rockstar'

NOTE:

In backward chaining:

If hypothesis is in the list of assertions, then we are done. Else find all the rules that matches the hypothesis. Then construct a goal tree of antecedants. Apply Depth-First search in recursion and if stuck do back tracking.

Backward Chaining:



Forward chaining:

	Matches	Fired	New assertions
1.	P0, P2, P1	P0: x? - Alison	A5: Alison is charismatic
		P0: x? - Brat	A6: Brat is charismatic
		P2: x? - Krish	A7: Krish wears leather
		P1: x? - Krish	A8: Krish can dance like Prabhudeva
2.	P0, P2, P1, P4	P5	"Krish should become a Rockstar"

Hence by A7 and A8: "Krish should become a Rockstar."

Example

A newspaper journalist wants to prove that Tony Stark is Iron Man, starting from the following rules:

P0: If 'a? sells weapons'
then 'a? is a genius'

P1: If (AND 'a? is a genius',
(OR 'a? is captured',
'a? is evil'))
then 'a? builds a suit'

P2: If '?a plots against ?b'
then '?a is evil',
'?b is captured'

P3: If (AND '?a sells weapons',
'?b is evil')
then '?a catches ?b selling illegal weapons',
'?a stops selling weapons')

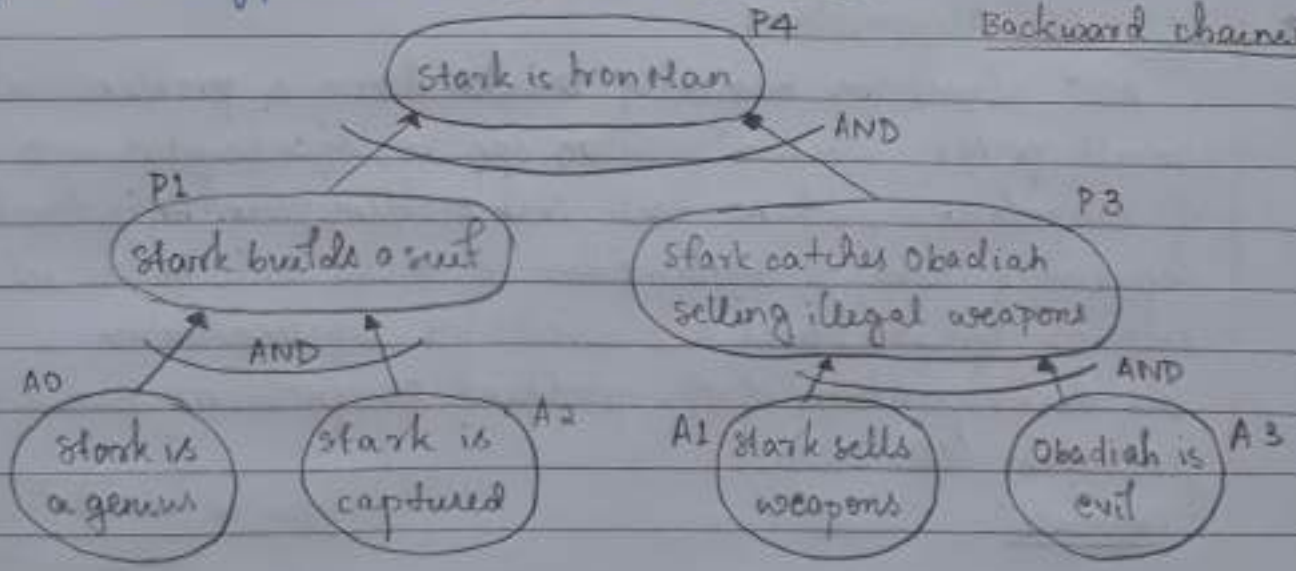
P4: If (AND '?a builds a suit',
'?a catches ?b selling illegal weapons')
then '?a is Iron Man'

The journalist starts with four assertions for backward chaining:

A0: 'Stark is a genius'
A1: 'Stark sells weapons'
A2: 'Stark is captured'
A3: 'Obadiah is evil'

Using these assertions, perform backward chaining starting from the hypothesis: 'Stark is Iron Man'

Backward chaining



Another journalist uses the following assertions with same set of rules

A0: 'Obadiah is a genius'
A1: 'Stark sells weapons'

A2: 'Obadiah plots against Stark'
A3: 'Stark is genius'
Prepare your table with new assertion added at the end of the assertion list and find whether 'Stark is Iron man'.

Forward Chaining

	Matched	Fired	New assertions
1.	P0, P2	P2: a? - Obadiah b? - Stark	A4: Obadiah is evil A5: Stark is captured
2.	P0, P2, P1, P3	P1: a? - Obadiah P1: a? - Stark	A6: Obadiah builds a suit A7: Stark builds a suit ✓
3.	P0, P2, P1, P3	P3: a? - Stark b? - Obadiah	A8: Stark catches Obadiah selling illegal weapons. ✓ A9: Stark stops selling weapons
4.	P0, P2, P1, P3, P4	P4: a? - Stark b? - Obadiah	"Stark is Iron Man"

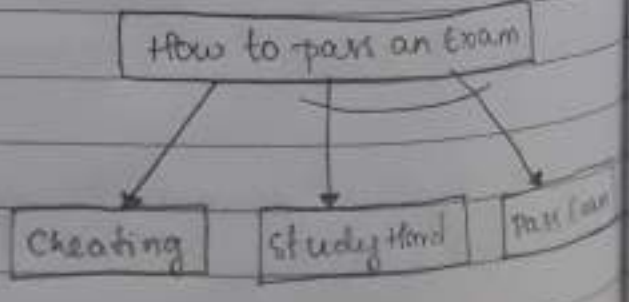
Hence, assertions A7 and A8 ⟶ "Stark is Iron Man".

- **SLE: AO* Algorithm**

AO* algorithm basically breaks down a problem into small pieces. When a problem can be subdivided into a set of subproblems, where each subproblem can be solved separately and a combination of these will be a solution. AND-OR graphs / trees are used for representation. The decomposition of the problem or problem reduction generates AND arcs.

Example:

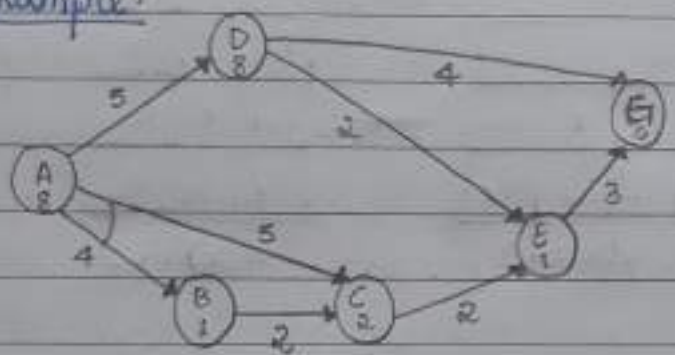
To pass any exam, we have two options, either cheating or hard work. Here we have replicated the arc (AND operation) between



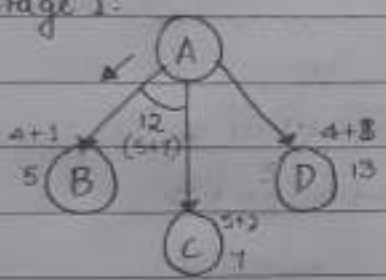
the work hard and the pass is because by doing the hard work there is greater possibility of passing an exam than by cheating.

AO* might not provide an optimal solution always. This is because it does not explore all the solution path once it gets a solution.

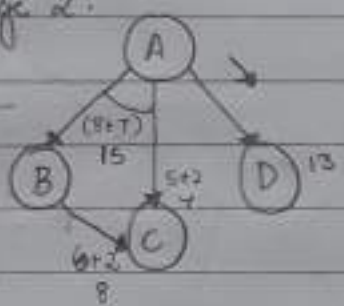
Example:



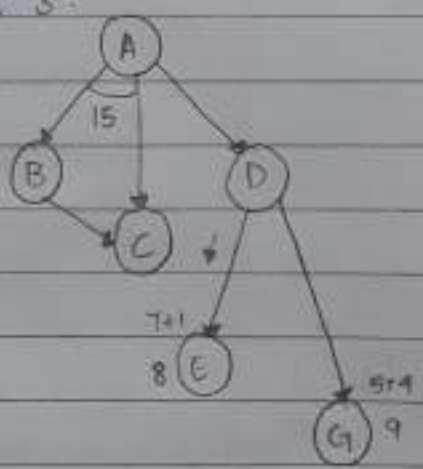
stage 1:



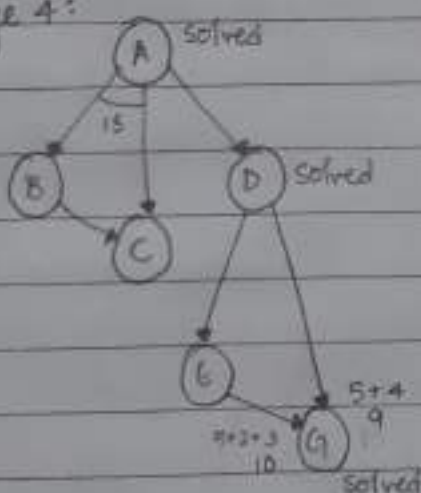
stage 2:



stage 3:



Stage 4:



Assignment:

Problem state

2	8	3
1	6	4
7	x	5

Goal state

1	2	3
8	x	4
7	6	5

Solution.

2_{-1}	8_{-1}	3_0
1_{-1}	6_{-1}	4_0
7_0	x	5_0

$h(x) = -4$ →

2_{-1}	8_{-1}	3_0
1_{-1}	x	4_0
7_0	6_0	5_0

$h(x) = -3$ →

8_{-1}	2_0	3_0
1_{-1}	x	4_0
7_0	6_0	5_0

$h(x) = -2$ ↓

1_0	2_0	3_0
8_0	x	4_0
7_0	6_0	5_0

Goal state

$h(x) = 0$

UNIT - 03

Game playing and CSP

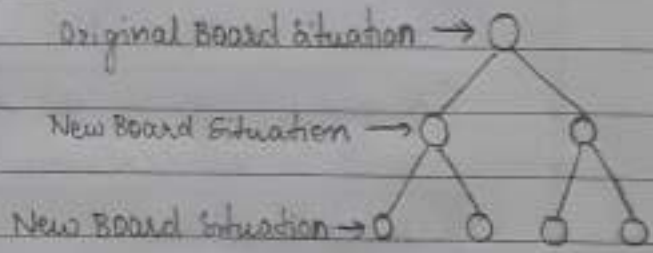
- **Trees and Adversarial Search:**

Adversarial search is where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

Adversarial searches, often known as games are searches in which two or more players with conflicting goals are trying to explore the same search space for the solution. Game situations can be represented as trees.

- Nodes Represent Board Positions:

The natural way to represent what can happen in a game is to use a game tree, which is a special kind of semantic tree in which the nodes denote board configurations, and the branches indicate how one can be transformed into another by a single move.



Game raises a new issue: competition. The nodes in a game tree represent board configurations and the branch indicates how moves can connect them.

The ply of a game tree p, is the number of levels of the tree, including the root level. If the depth of a tree is d, then

$$p = d + 1$$

In gaming each choice is referred as a move, i.e., the decisions made by two adversaries who take turns to make decisions.

- The Minimax Procedure

consider a situation analyzer that converts all judgements about board situations into single overall quantity number. Suppose positive numbers indicate favor to one player and negative numbers indicate favor to the other. Then the degree of favor increases with the absolute value of the number

- Static evaluation: process of computing a number that reflects board quality.
- Static evaluator: procedure that does computation
- Static evaluation score: the number computed.
- Maximizing Player / Maximizer: player hoping for positive numbers
- Minimizing Player / Minimizer: player hoping for negative numbers

A game tree is a representation
That is a semantic tree -
In which
- Nodes denote board configurations
- Branches denote moves

With writers that
- Establish that a node is for the minimizer or the maximizer.
- connect a board configuration with a board configuration description.

With readers that
- Determine whether the node is for the maximizer or the minimizer
- Produce a board configuration's description

The maximizer looks for the move that leads to a larger positive number and assumes that the minimizer will try to

force the play towards situations with strongly negative static evaluations. Also the decisions of maximizer/minimizer must take cognizance of the choices available to the minimizer/maximizer at the next level down.

The procedure by which the scoring information passes up the game tree is called the MINIMAX procedure, because the score at each node is either the minimum or the maximum of the scores at the nodes immediately below.

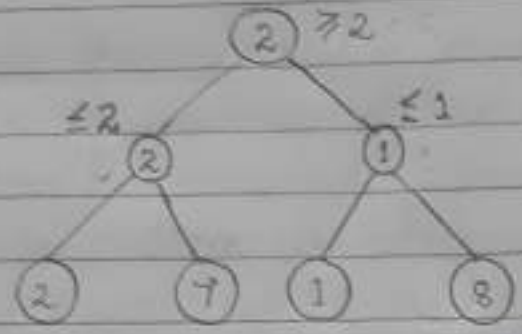
To perform a minimax search using MINIMAX,
- If the limit of search has been reached, compute the static value of the current position relative to the appropriate player. Report the result.
- Otherwise, if the level is a minimizing level, use MINIMAX on the children of the current position. Report the minimum of the result.
- Otherwise, the level is a maximizing level. Use MINIMAX on the children of the current position. Report the maximum of the result.

Demerits: It is dependent on a single, summarizing number, the static value which is unfortunately a poor summary. Also minimaxing is expensive because either the generation of paths or static evalution can require a lot of computation.

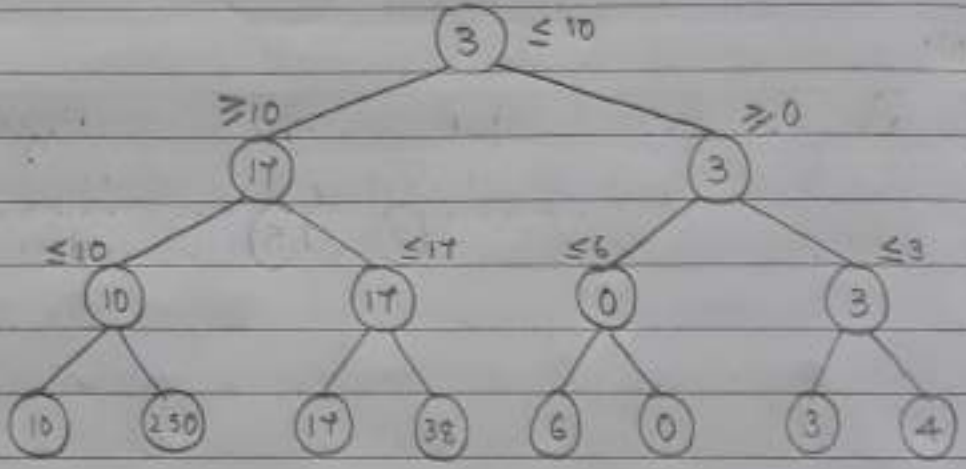
Example: MINIMAX

Minimaxing is a method for determining moves. Minimaxing employs a static evaluator to calculate advantage specifying numbers for the game situations at the bottom of a partially developed game tree. One player works toward the higher numbers seeking the advantage, while the opponent goes for the lower numbers.

For the following example try to maximize.



Example 2: MININAX - Minimizing each node



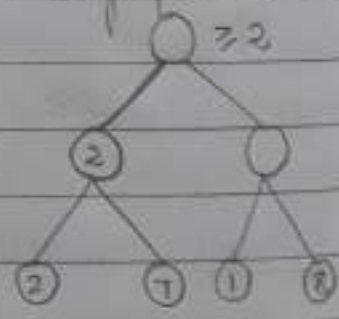
— The Alpha - Beta Procedure: Pruning

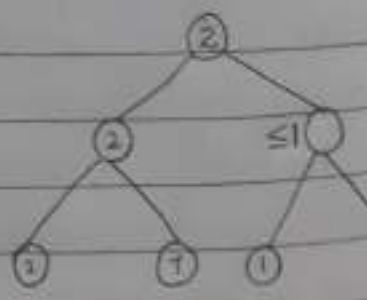
Pruning reduces both the number of tree branches that must generate and the number of static evaluations that must be done, thus cutting down on the work to be done overall.

In this example, there is no need to explore the right side of the tree fully, because there is no way the result could alter the move decision.

Once the movement to the right is shown to be worse than movement to the left, there is no need to see how much worse.

Considering previous example

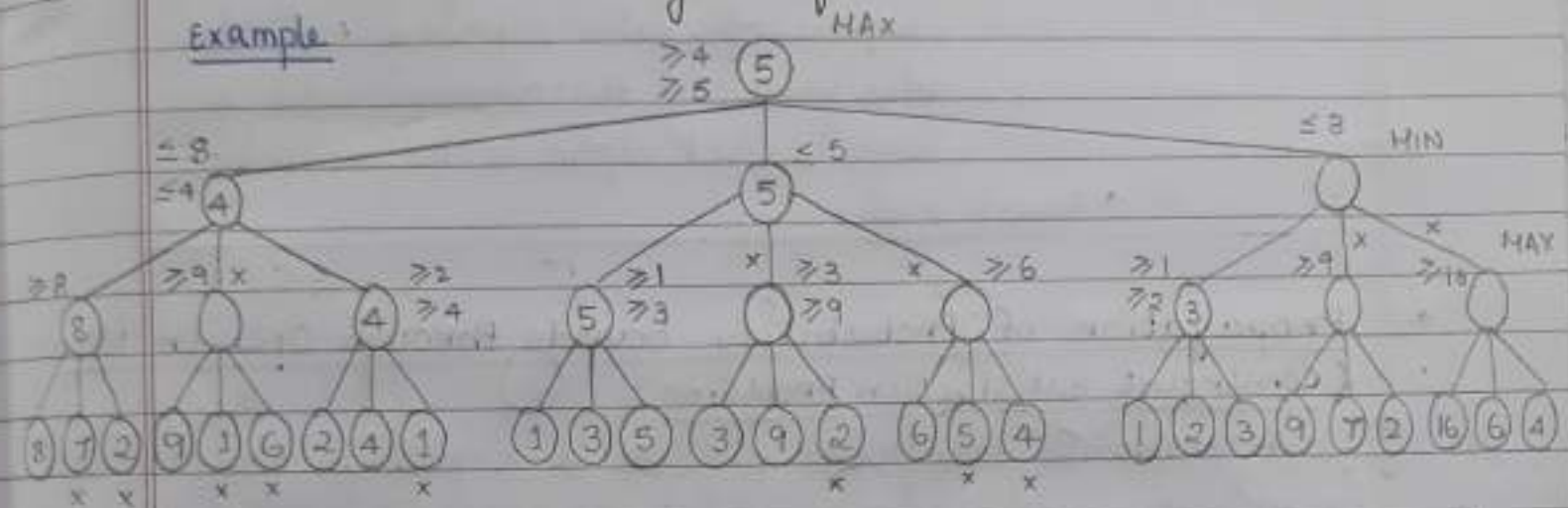




The Alpha-Beta Principle:

If you have an idea that is surely bad, do not take time to see how truly awful it is.

Example:



To perform minimax search with the ALPHA-BETA procedure:

- If the level is the top level, let alpha be $-\infty$ and let beta be ∞.
- If the limit of search has been reached, compute the static value of the current position relative to the appropriate player. Report the result.
- If the level is a minimizing level,
  - Until all children are examined with ALPHA-BETA or until alpha is equal to or greater than beta,
    - Use the ALPHA-BETA procedure, with the current alpha-beta values, on a child; note the value reported.
    - Compare the value reported with the beta value; if the reported value is smaller, reset beta to the new value.
  - Report beta.
- Otherwise, the level is a maximizing level,
  - Until all children are examined with ALPHA-BETA or alpha is equal to or greater than beta,

- Use the ALPHA-BETA procedure, with the current alpha and beta value, on a child; note the value reported.
- Compare the value reported with the alpha value; if the reported value is larger, reset alpha to the new value.

• Report alpha.

- **Propagation of Probability bounds through Opinion Nets: (Constraint Satisfaction Problems)**

A constraint satisfaction problem is a problem that requires its solution within some limitations or conditions also known as constraints. It consists of:

- a set of variables
- a domain for each variable
- set of constraints.

Each state in a CSP is defined by an assignment of values to some or all of the variables. An assignment that does not violate any constraints is called a consistent assignment. A complete assignment is one in which every variable is assigned. A solution to CSP is consistent and complete.

Example 1: The 4-queen problem

Place 4 queens on a 4x4 board such that no two queens reside in the same row, column or diagonal.

- Variables : Q_1, Q_2, Q_3, Q_4
  Domain : 1, 2, 3, 4 (columns) $\{a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_1, c_2, c_3, c_4, d_1, d_2, d_3, d_4\}$
  constraints: Queens can not be on the same row
  Queens can not be on the same column
  Queens can not be on the same diagonal

(1)

	1	2	3	4
a	Q_1	—	—	—
b	\	\	Q_2	—
c	\	/	\	\
d	\	Q_3	\	\	X

	1	2	3	4
a	Q_1	—	—	—
b	\	\	—	Q_2
c	\	Q_3	\	\
d	\	\	Q_4	\	X

$\rightarrow Q_1 = a_1$

$Q_2 = \{b_3, b_4, c_2, c_4, d_2, d_3\}$

$\rightarrow Q_2 = b_3 \ni Q_3 = \{d_2\}$

$\rightarrow Q_3 = b_4 \ni Q_3, Q_4 = \{c_2, d_3\}$

(2)

	1	2	3	4
a	—	Q_1	—	—
b	\	\	\	Q_2
c	Q_3	\	—	\
d	\	\	Q_4	\	✓

$\rightarrow Q_1 = a_2$

then $Q_2 = \{b_4, c_1, c_3, d_1, d_2, d_4\}$

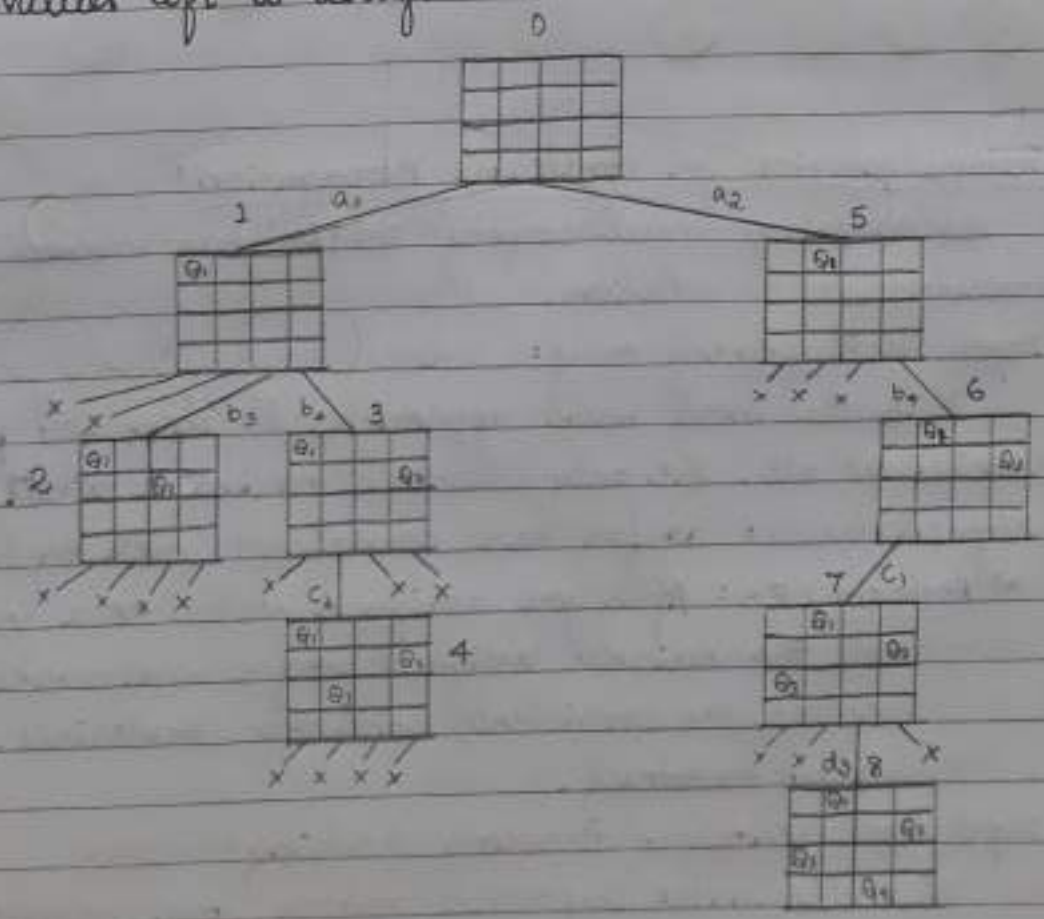
$\rightarrow Q_2 = b_4$

then $Q_3, Q_4 = \{c_1, d_3\}$

This problem can be solved using backtracking to avoid trying every possibility until solution is acquired.

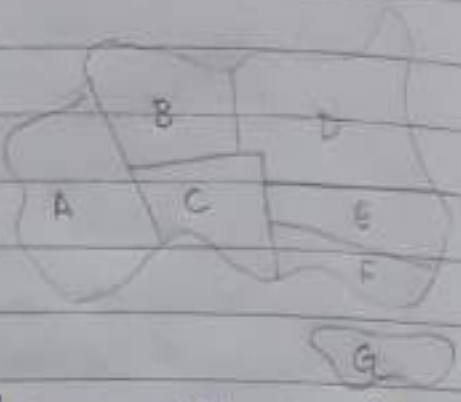
- Backtracking:

It is a form of depth-first search. It chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.

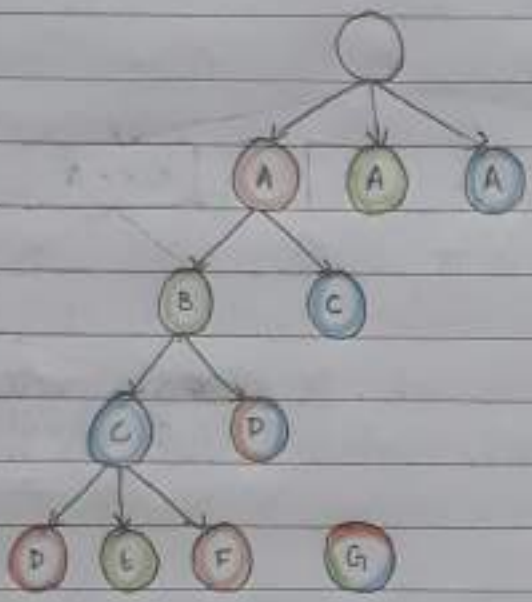


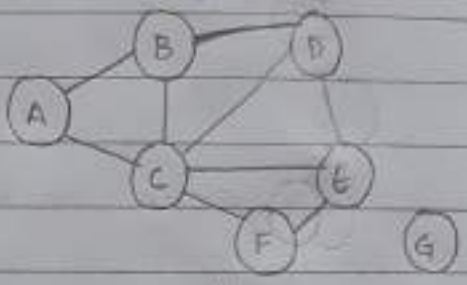
Example 2: Map coloring

Given a map, color it using three colors such that no neighboring territories have the same color.

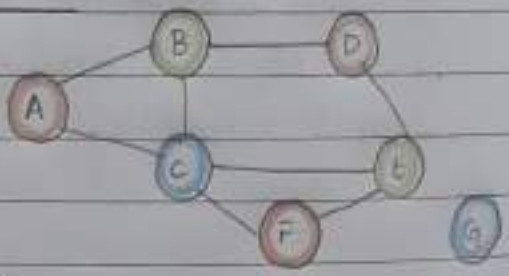


- Variables: A, B, C, D, E, F, G
  Domain: {red, green, blue}
  constraints: adjacent regions must have different colors.





Solution:



- Four strategies for constraint Propagation:

There are four different strategies for pruning branches while searching for a solution.

1. Depth first search only:

It is the most basic approach. It does not prune any branches at all. The only check it makes is that all the assigned values so far are consistent with each other.

a) Perform DFS: After you assign a value to a variable, examine all the variables assigned so far and make sure those values are consistent with the constraints. If they aren't then backtrack.

2. Depth first search + forward checking:

This eliminates impossible options from neighboring variables.

a) Perform DFS

b) Perform Forward Checking: After assigning a value to a variable, consider all of its neighbors. Eliminate any options from its neighbors that are incompatible with the value you just assigned.

3. Depth First Search + Forward Checking + Propagating through singleton domains:

It eliminates impossible options from neighboring variables. Then, if any of those neighboring variables have only one option left, it looks ahead to see what else it can eliminate.

a) Perform DFS

b) Perform Forward Checking:

c) Perform Propagation through singleton domains: Eliminate options from a neighbor from previous step and that neighbor has only one option left, add that neighbor to the list of variables to propagate. Propagate all the variables in the list.

4. Depth First Search + Forward checking + propagation through reduced domains:

It eliminates impossible options from neighboring variables. If it successfully eliminates any options from those variables, it looks ahead to see what else it can eliminate.

a) Perform DFS

b) Perform Forward Checking

c) Perform Propagation through reduced domains: Eliminate any options from a neighbor in previous step, add that neighbor to the list of variables to propagate. Propagate all the variables in the list. To propagate a reduced variable, take note of its remaining options and consider each of its neighbors.

Example 3:

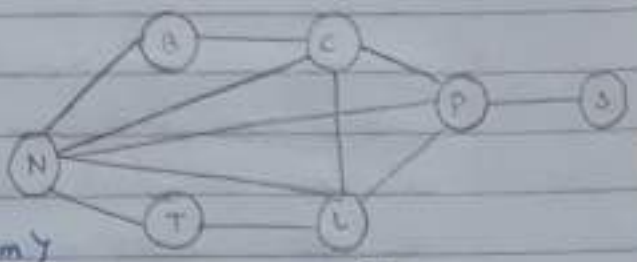
The NIE Time Travel society (NIETTS) has invited seven famous historical figures to each give a lecture at the annual NIETTS convention, and you have been asked to create a schedule for them. Unfortunately, there are only four time slots available and you discover that there are some restrictions on how you can schedule the lectures and keep all the convention attendees happy. For instance, physics students will be disappointed if you schedule Niels Bohr and Isaac Newton to speak during the same time slot, because those students were hoping to attend both of these lectures.

After talking to some students who are planning to attend this year's convention, you determine that they fall into certain groups, each of which wants to be able to see some subset of the time-traveling speakers.

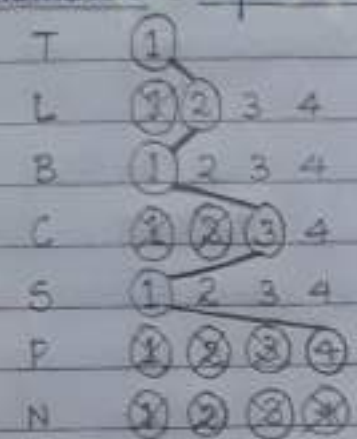
1. The list of lectures consists of Alan Turing, Ada Lovelace, Niels Bohr, Marie Curie, Socrates, Pythagoras and Isaac Newton.
2. Turing has to get home early to help win World War II, so he can only be assigned to the 1pm slot.
3. The course VIII students want to see the physicists: Bohr, Curie and Newton.
4. The course XVIII students want to see the mathematicians: Lovelace, Pythagoras and Newton.
5. The members of the Ancient Greece Club wants to see the ancient Greeks: Socrates and Pythagoras.
6. The visiting Wellesley students want to see the female speakers: Lovelace and Curie.
7. The CMU students want to see the British speakers: Turing, Lovelace and Newton.
8. Finally, you decide that you will be happy if and only if you get to see both Curie and Pythagoras.

- Variables: Turing, Lovelace, Bohr, Curie, Socrates, Pythagoras, Newton (T, L, B, C, S, P, N)

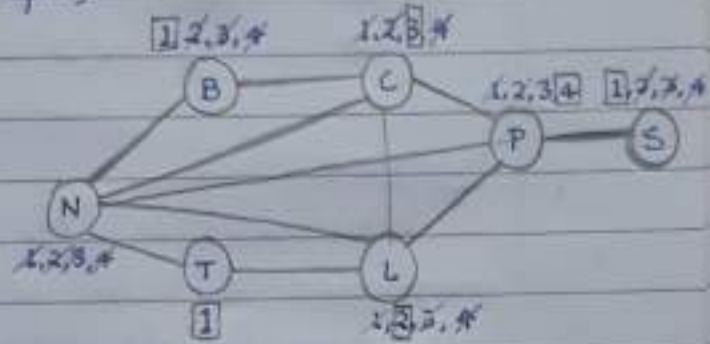
Domain: Slots: {1pm, 2pm, 3pm, 4pm}



Method 1: Depth First Search

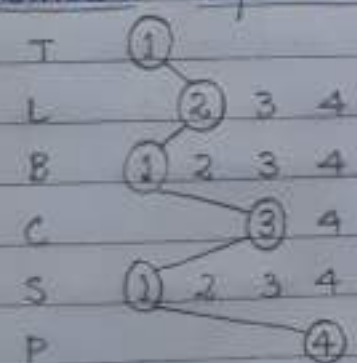


No slot is available for N.

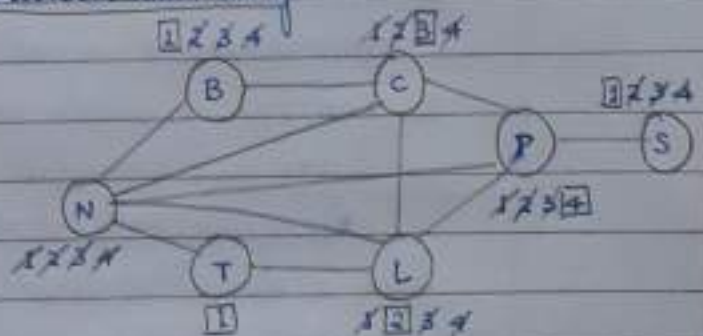


1. Make assignment
2. check for contradiction in the previous assignment

Method 2: Depth First Search + Forward checking

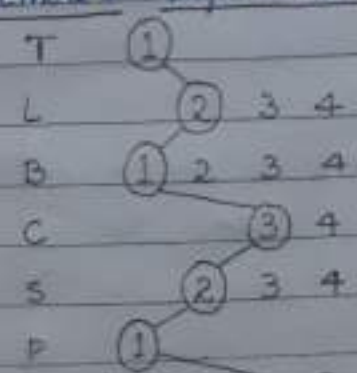


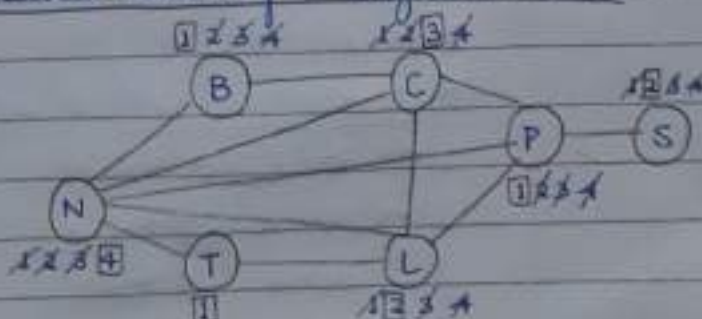
No slot is available for N.



1. Make assignment
2. After each assignment, eliminate values from neighboring domain

Method 3: Depth First Search + Forward checking + Singleton domain:





In this case all the variables are assigned with a slot without any contradiction with the constraints.

1. After DFS + FC check for newly created singleton domains
2. If found, don't assign it but eliminate values from the neighboring domains
3. If this creates a new singleton domain, propagate further

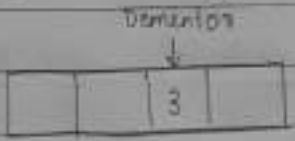
Example 4:

Three students (Ron, Hermione, Malfoy) are looking for train compartments to sit in on the Hogwarts Express train. There are four compartments (1, 2, 3, 4) and each compartment can hold any number of students. The students have following constraints:

1. Malfoy despises Ron and Hermione, so Malfoy must not sit in the same compartment.
2. Hermione wants to keep an eye on Malfoy, so their compartments must be adjacent (and not the same)
3. Ron has a crush on Hermione; but he also thinks she's an obnoxious know-it-all, so their compartments must be adjacent (and not the same).

Alas, a dementor has just arrived in compartment 3, so no one should sit there.

Write the proper adjacent graph based on the rules, using DFS and forward checking and no propagation, assign students in the order of Ron - Malfoy - Hermione.



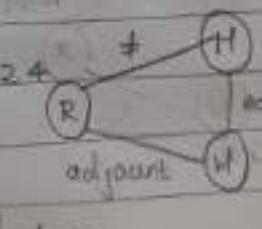
- Variables: Ron, Hermione, Malfoy
  Domain: 1, 2, 4

→ Depth First Search + Forward Checking

→ R = 1 H ≠ 1, 4 ; M ≠ 1
  M = 2 H ≠ 2
  M = 4 H ≠ 2

→ R = 2 H ≠ 2, 4 ; M ≠ 2
  M = 1 H ≠ 1
  M = 4 H ≠ 1

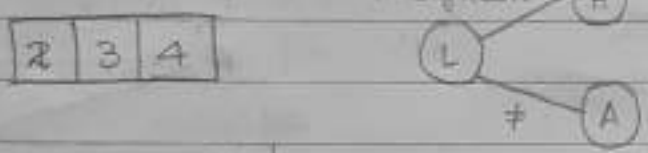
→ R = 4 H ≠ 1, 2, 4



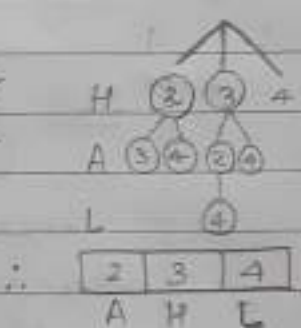
No solution

Example 5:

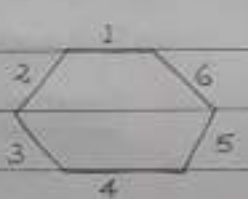
There are three cages. There are three animals: lion, hornbill and antelope. Place them in the cages such that lion and hornbill are adjacent whereas lion and antelope is not.



Variables assigned or propagated	Values eliminated from neighboring domain	Backtracking
H = 2	L ≠ 2,4	
A = 3	L ≠ 3,4	✓
A = 4	L ≠ 3,4	✓
H = 3	L ≠ 3	
A = 2	L ≠ 2	
L = 4		



Example 6:

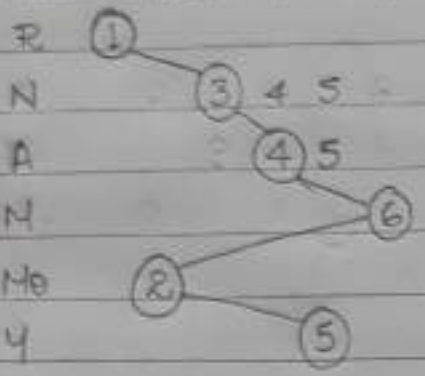


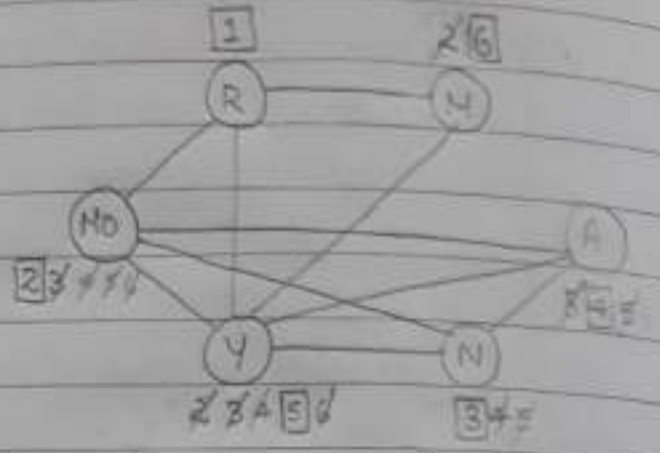
You bought a 6 sided table and want to hold a dinner party. You invited four best friends: Rahul, Namo, Amit and Hamta. Luckily a moose wanders by and accepts your invitation. Counting yourself, you have 6 guests labelled 1 to 6. Your guests have seven seating demands:

1. Hamta wants to sit next to Rahul
2. Amit wants to sit next to Namo
3. Neither Rahul nor Hamta will sit next to Namo or Amit
4. Neither Namo nor Amit will sit next to Rahul or Hamta.
5. The moose is afraid to sit next to Hamta.
6. No two people can sit in the same seat and no one can sit in two seats.
7. Rahul insists on sitting in seat 1.

Based on the constraints given, find the solution.

- Variables : Namta, Rahul, Amit, Name, Hoose, You
Domain ⟨1, 2, 3, 4, 5, 6⟩





- **SLE: Numerical Constraints and Propagation: (crypto Arithmetic)**

Crypt-Arithmetic is the science and art of creating and solving cryptarithm. A cryptarithm is a genre of mathematical puzzle in which the digits are replaced by letters of the alphabets or other symbol.

Example 1 :

$C_3\ C_2\ C_1$

```
  M O O N
+ N O O N
+ S O O N
---------
  J U N E
```

Variables : M, O, N, S, J, U, E

Domain : ⟨0, 1, 2, 3, 4, 5, 6, 7, 8, 9⟩

constraint : no two letters can be the same number.

$M \neq 0, N \neq 0, S \neq 0$

N		1	(2)	3	4	5	6	7	8	9
E	0	1		3	4	5	(6)	7	8	9
O	0	1		3	(4)	5		7	8	9
U	0	1		(3)		5		7	8	9
M		(1)				5		7	8	9
S						(5)		7	8	9
J	0							7	8	(9)

$C_2 = 1$

$C_3 = 1$

$\therefore$

```
  1 1
  M O O N
  1 4 4 2
  N O O N
+ 2 4 4 2
  S O O N
+ 5 4 4 2
---------
  9 3 2 6
```

$9_J\ 3_U\ 2_N\ 6_E$

Example 2:

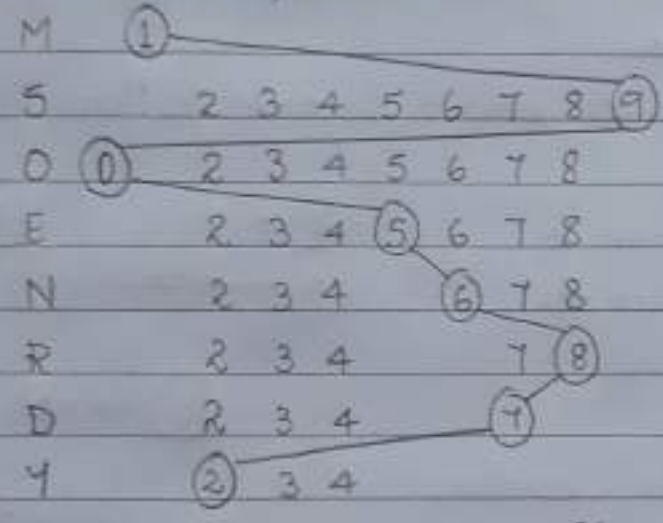
```
  SEND
+ MORE
MONEY
```

Variables: S, E, N, D, M, O, R, Y

Domain: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Constraint: No two letters can have same number.

$S \neq 0, M \neq 0$; $M = 1 \Rightarrow S = 9$



$\therefore$

```
  9 5 6 7
+ 1 0 8 5
-------
1 0 6 5 2
```

Assignment:

Q: Six powerful mages (Ajani, Bolas, Chandra, Dack, Elspeth and Gideon) have enlisted you to help them settle down and find homes in the multiverse. They have five neighboring worlds to choose from, arranged in a row. Some worlds could be empty and some worlds could have more than one occupant. Naturally, the mages are very particular about where they live, so your assignments must satisfy the following requirements:

1	2	3	4	5

1. Ajani is Elspeth's mentor. They must live on the same world.
2. Everyone hates Bolas. No one can live on the same world as him.
3. Chandra wants to spy on Dack secretly - she must live on a world that is adjacent to Dack's world.
4. Gideon is a law mage. He despises the criminals Chandra and Dack so much that Gideon cannot live on the same world or even on a world adjacent to Chandra's or Dack's.

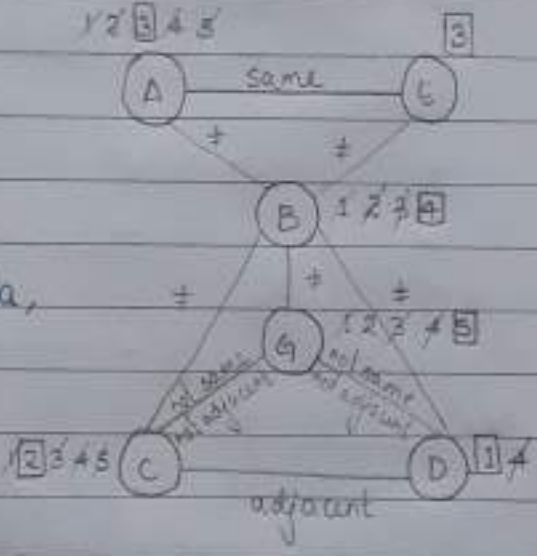
Finally the mages inform you that you can simplify the domains of these variables as follows:

- Dack must live on either world 1 or 4, because he is a wanted criminal everywhere else.
- Elspeth must live on world 3, because she's stuck there and cannot leave.
- Bolas can't live on world 5, because he fears its strange magic.

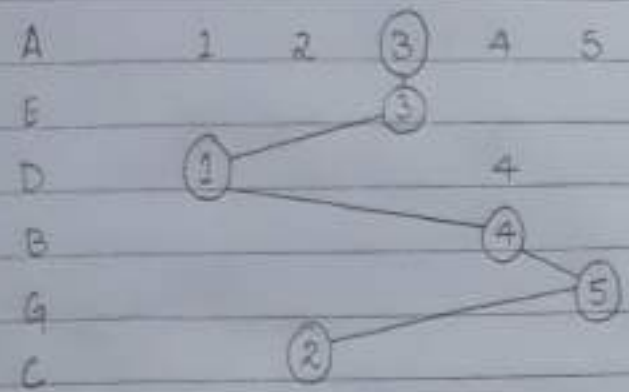
NOTE: constraint propagation is maximally effect when the one with more edges are assigned first.

— Variable: Ajani, Bolas, Chandra, Dack, Elspeth, Gideon
(A, B, C, D, E, G)

Domain: {1, 2, 3, 4, 5}



Variables assigned or propagated	Values eliminated from the neighboring domain	Back tracking
A = 1	B ≠ 1 E ≠ 3	
A = 2	B ≠ 2 E ≠ 3	
A = 3	B ≠ 3	✓
E = 3		✓
D = 1	B ≠ 1, G ≠ 1,2, C ≠ 1,3,4,5	✓
C = 2	B ≠ 2 G ≠ 3	✓
B = 4	G ≠ 4	✓
G = 5	—	✓
B = 4	—	✓
G = 5	—	✓
C = 2	—	✓



Therefore

1	2	3	4	5
D	C	A, E	B	G

Q: Using crypt-arithmetic solve the following:

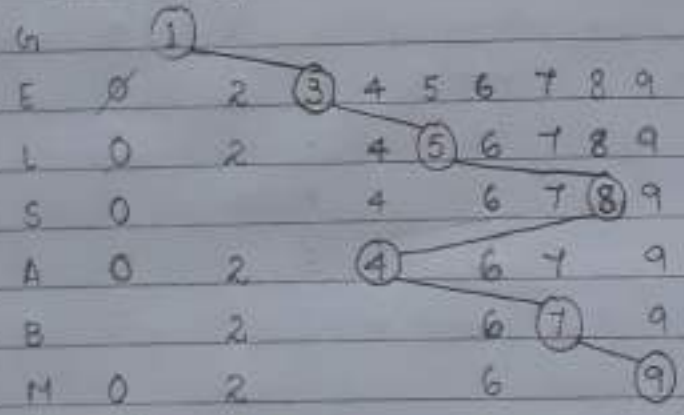
a. BASE + BALL = GAMES

b. TWO + TWO = SOX

— a. Variables: B, A, S, E, L, G, M

Domain: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

B ≠ 0 ; G = 1



∴

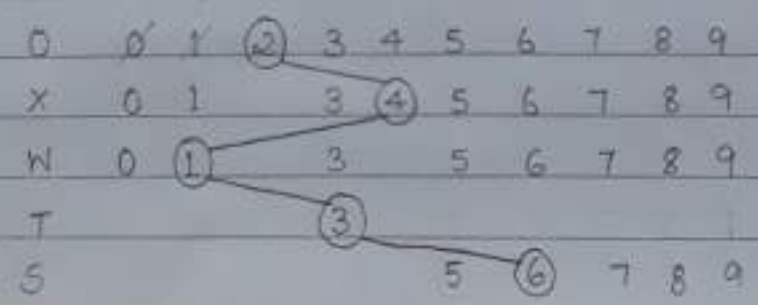
1

E	A	S	E
7	4	8	3
+ B	A	L	L
7	4	5	5
G	A	M	E	S
1	4	9	3	8

— b. Variables: T, W, O, S, x

Domain: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

T ≠ 0 , S ≠ 0 and T <= 4



∴

T	W	O
3	1	2
+ T	W	O
3	1	2
S	O	X
6	2	4

UNIT - 4

Different Learnings

- **Learning by Analyzing Differences:**

– Introduction to learning:

Learning can be described as normally a relatively permanent change that occurs in behavior as a result of experience. Ex: It is possible to learn to open a lock as a result of trial and error.

Once the internal model of what ought to happen is set, it is possible to learn by practicing the skill until the performance converges on the desired model. One begins by paying attention to what needs to be done, but with more practice, one will need to monitor only the trickier parts of the performance.

"Learning denotes changes in a system that enables a system to do the same task more efficiently next time."

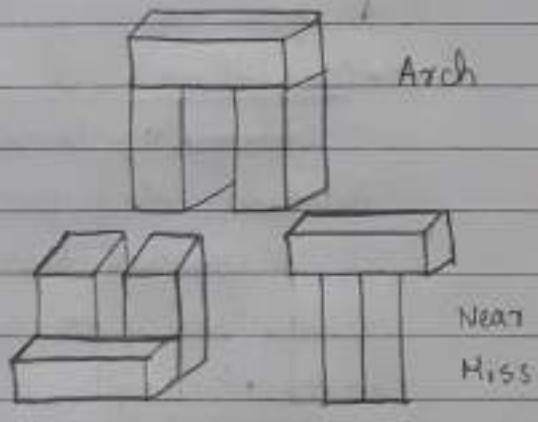
"Learning is constructing or modifying representations of what is being experienced."

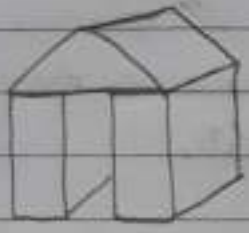
– Learning by Analyzing Differences:

Positive and Negative examples

A sequence of positive examples and near-miss examples for learning about arches.

- The brick must be supported by pillars.
- The pillars must not touch each other
- Top can be a wedge or a brick.





- Supervised Learning

Supervised learning is where we teach or train the machine using data that is well labeled, i.e., some data is already tagged with the correct answer. After that, the machine is provided with a new set of data so that the supervised learning algorithm analyses the training data and produces a correct outcome from labeled data.

Principles

1. You cannot learn if you cannot know:
   - Good teachers help their students by being sure that their students acquire the necessary representations.
2. You cannot learn if you cannot isolate what is important
   - Good teachers help their students by providing not only positive examples but also negative examples and near misses.

- K Nearest Neighbor (KNN):

K Nearest Neighbor algorithm stores all the available cases and classifies the new data or cased based on a similarity measure. It is mostly used to classify a data point based on how its neighbors are classified.

- Overfitting and Underfitting:

Algorithm learns from only a small part of the training dataset, then the model is underfitting.

Algorithm that performs very well on known instances but faulter badly on unseen data or unknown instances are said to be overfitting.

When a model does well in both the training dataset and on the unseen data or unknown instances, it is a good fit.

- KNN working:

KNN algorithm seeks for majority vote between the K most similar instances to a given unseen observation. similarity is defined according to a distance metric between two data points. This can be done by:

- Euclidean distance $\sqrt{(u-v)^2}$
- Manhattan distance
- Hamming distance (based on features)
- cosine distance

- Identification Trees:

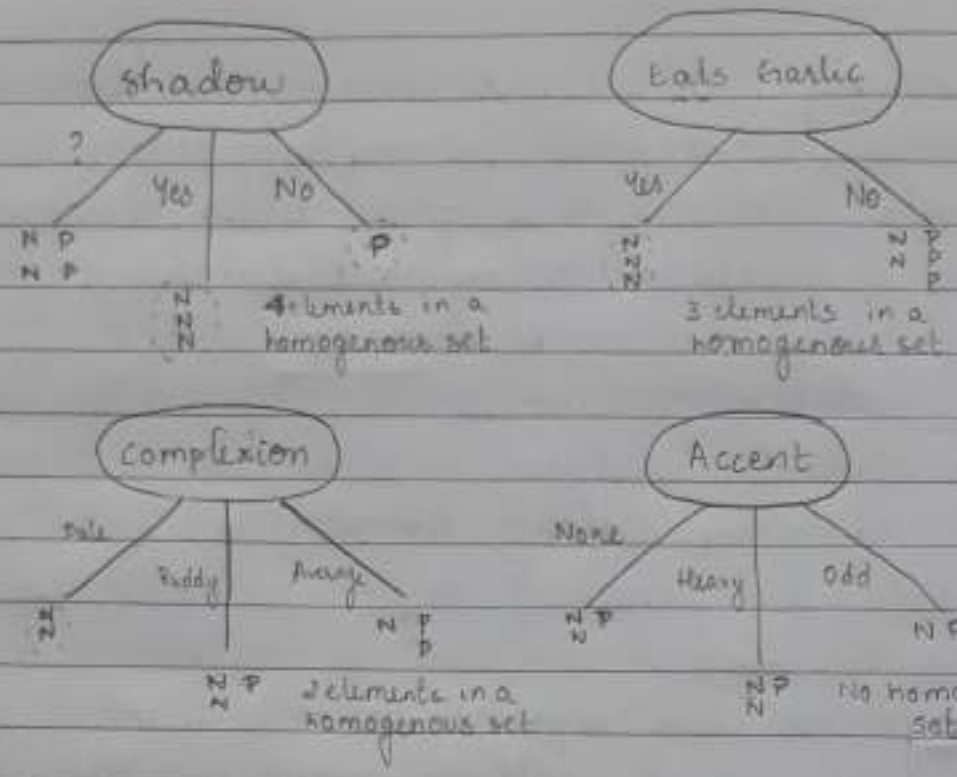
A decision tree is a semantic network in which each node is connected to a set of possible answers. An identification tree is a representation in which each set of possible conclusions is established implicitly by a list of samples of known class.

- Example:

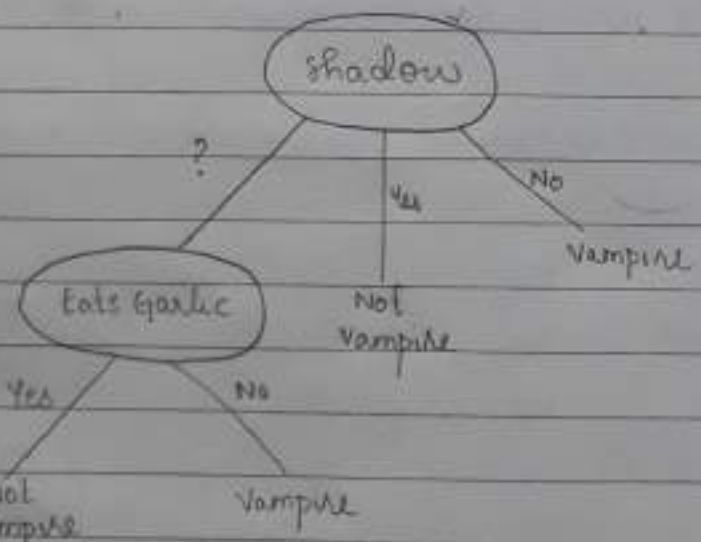
You don't want vampires to be your friend in future. So you made a list of characteristics of several people you have met and they turned out to be vampires or not.

	Casts shadow?	Eats Garlic?	Skin complexion?	Accent?	Vampire?
1.	? (don't know)	Yes	Pale	None	No
2.	Yes	Yes	Ruddy	None	No
3.	? (don't know)	No	Ruddy	None	Yes
4.	No	No	Average	Heavy	Yes
5.	? (don't know)	No	Average	Odd	Yes
6.	Yes	No	Pale	Heavy	No
7.	Yes	No	Average	Heavy	No
8.	? (don't know)	Yes	Ruddy	Odd	No

Now using the above given data we can draw an identification tree such that it satisfies all data in the given data set and satisfy future inputs as well.



We see which one has the highest number of elements in the homogenous set for each attribute. Here, 'shadow' attribute has the highest count for elements in a homogenous set, so we choose this attribute.



For shadow = ? let us consider the next attribute here it is "eats garlic" Now consider only the observations (1,3,5,8) where shadow = ? for the attribute "eats garlic"

From the above tree we can see that, only these two attributes is sufficient to conclude where vampire or not. From this identification tree we can write the rule based approach to decide whether a person is vampire or not

P0: If (OR '?x casts shadow',
'?x eats garlic')
Then '?x is not a vampire'

P1 If (AND '?x doesnot cast shadow',
'?x doesnot eat garlic')
then '?x is a vampire'

In case of any dataset, it is not possible to employ the previous method as we may find no homogenous sets.

Accordingly, for real databases, one needs a way to measure the total disorder or homogeneity, in the subsets produced by each test. To compute a measure of disorder:

$$\text{Disorder} = -\frac{P}{T}\log_2\left(\frac{P}{T}\right) - \frac{N}{T}\log_2\left(\frac{N}{T}\right)$$

where P: total number of positive observations
N: total number of negative observations
T: Total number of observations.

$$D_{test} = \sum\left[D_{set} \times \frac{\text{number of samples in the set}}{\text{Total number of samples}}\right]$$

For attribute: Casts shadow:

→ ? (don't know)

$$-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 0.5 + 0.5 = \underline{1}$$

→ Yes

$$-\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = \underline{0}$$

→ No

$$-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = \underline{0}$$

$$\therefore D_{test} = 1 \times \frac{4}{8} + 0 \times \frac{3}{8} + 0 \times \frac{1}{8} = \underline{0.5}$$

For attribute: eats garlic

→ Yes

$$-\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = 0$$

→ No

$$-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.44 + 0.53 = 0.97$$

$$\therefore D_{test} = 0 \times \frac{3}{8} + 0.97 \times \frac{5}{8} = 0.61$$

For attribute: skin complexion

→ Pale

$$-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

→ Ruddy

$$-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.53 + 0.39 = 0.92$$

→ Average

$$-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.39 + 0.53 = 0.92$$

$$\therefore D_{test} = 0 \times \frac{2}{8} + 0.92 \times \frac{3}{8} + 0.92 \times \frac{3}{8} = 0.69$$

For attribute: Accent

→ None

$$-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.53 + 0.39 = 0.92$$

→ Heavy

$$-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.53 + 0.39 = 0.92$$

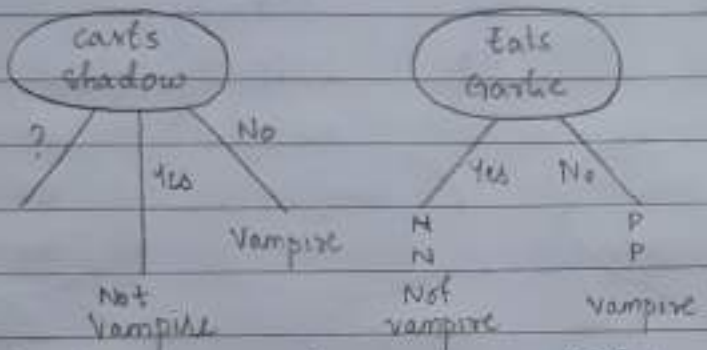
→ Odd

$$-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 0.5 + 0.5 = 1$$

$\therefore$ Dtest = $0.92 \times \frac{3}{8} + 0.92 \times \frac{3}{8} + 1 \times \frac{2}{8} = 0.94$

Now, we have to consider the attribute with least disorder. Here it is attribute carts shadow.

The next attribute to be considered is eats garlic



	Eats Garlic?	carts shadow?	Vampire?
1.	Yes	?	No
2.	~~Yes~~	~~Yes~~	~~No~~
3.	No	?	~~Yes~~
4.	~~No~~	~~No~~	~~Yes~~
5.	No	?	Yes
6	~~No~~	~~Yes~~	~~Yes~~
7.	~~No~~	~~Yes~~	~~No~~
8.	Yes	?	No

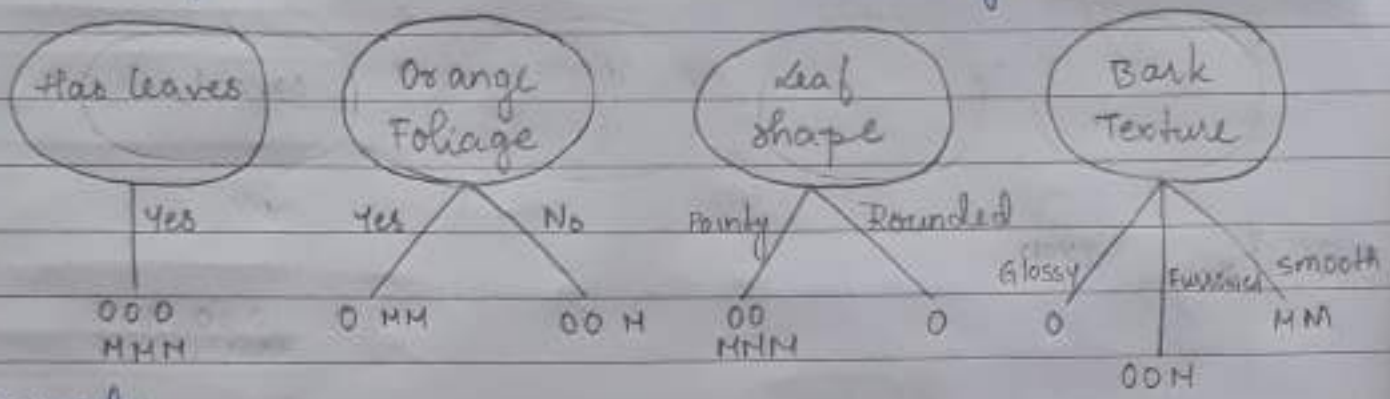
Therefore, disorder for the attribute eats garlic is 0.

- Example 2:

While walking along Charles river in the fall your friend points out six trees and tells you whether they are maple or oak trees. Eager to identify other trees yourself, you observe what appears to be key features. In particular, you recorded whether each tree has leaves, orange foliage, shape of the leaf and bark texture. The resulting data is summarised in the table.

(Although some trees lose their leaves in the fall, all six of the trees still have leaves)

	Tree Type	Has Leaves	Orange Foliage	Leaf Shape	Bark texture
1.	Oak	Yes	Yes	Pointy	Glossy
2.	Oak	Yes	No	Pointy	Furrowed
3.	Oak	Yes	No	Rounded	Furrowed
4.	Maple	Yes	Yes	Pointy	Furrowed
5.	Maple	Yes	Yes	Pointy	Smooth
6.	Maple	Yes	No	Pointy	Smooth



Disorder

For attribute: Has Leaves

→ Yes

$$-\frac{3}{5}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 0.5 + 0.5 = 1$$

$$D_{test} = 1 \times \frac{6}{6} = \underline{1}$$

For attribute: Orange Foliage

→ Yes

$$\frac{-1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.53 + 0.39 = 0.92$$

→ No

$$\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.39 + 0.53 = 0.92$$

$$D_{test} = 0.92 \times \frac{3}{6} + 0.92 \times \frac{3}{6} = \underline{0.92}$$

For attribute : Leaf Shape

→ Pointy

$$\frac{-2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.53 + 0.44 = 0.97$$

→ Rounded

$$\frac{-1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$$

$$D_{pet} = 0.97 \times \frac{5}{6} + 0 = \underline{0.8}$$

For attribute : Bark texture

→ Glossy

$$\frac{-1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$$

→ Furrowed

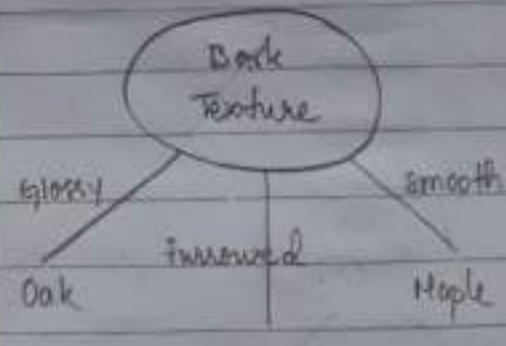
$$\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.39 + 0.53 = 0.92$$

→ Smooth

$$\frac{-0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

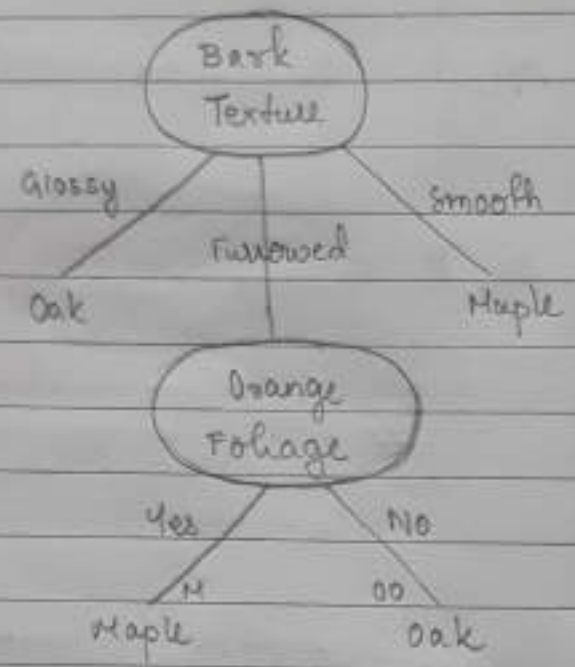
$$D_{test} = 0 \times \frac{1}{6} + 0.92 \times \frac{3}{6} + 0 \times \frac{2}{6} = \underline{0.46}$$

The attribute bark texture has the least disorder.



	Tree Type	Has leaves	Orange Foliage	Leaf shape	Bark Texture
1.	Oak	Yes	Yes	Pointy	Glossy
2.	Oak	Yes	No	Pointy	Furrowed
3.	Oak	Yes	No	Rounded	Furrowed
4.	Maple	Yes	Yes	Pointy	Furrowed
5.	Maple	Yes	Yes	Pointy	Smooth
6.	Maple	Yes	No	Pointy	Smooth

Let us now consider orange foliage.



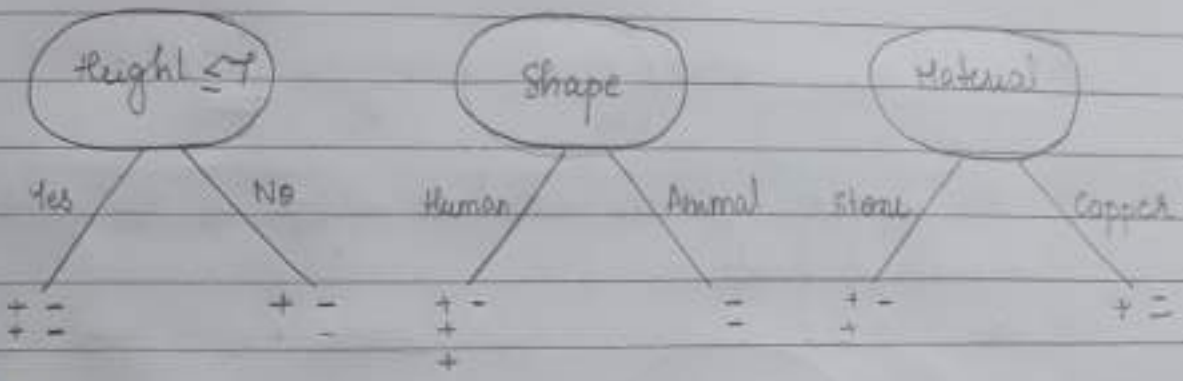
P0: If (OR '?x has glossy bark',
(AND '?x has furrowed bark',
'?x does not have orange foliage'))
then '?x is an oak tree'

P1: If (OR '?x has smooth bark',
(AND '?x has furrowed bark',
'?x has orange foliage'))
then '?x is a maple tree'

- Example 3:

The doctor and his companions: Ramesh, Shakaal, Don have found themselves in a city full of statues. They have to keep their eyes on the statues - some of them are actually quantum-locked Weeping Angels that can only move when they are not being watched. If a Weeping Angel touches one of them, they will be sent back in time and never found again. The Doctor is looking for a way to distinguish Weeping Angels from other statues. Luckily, River's journal has some information about statues from her previous adventures with the doctor.

Statue	Classification	Height in ft	Shape	Material
1.	Angel	4	Human	Stone
2.	Angel	2.5	Human	Stone
3.	Not Angel	4	Human	Copper
4.	Not Angel	3	Animal	Copper
5.	Not Angel	8	Animal	Stone
6.	Angel	305	Human	Copper



Disorder

For attribute: Height less than 6 Ft

→ Yes

$$-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 0.5 + 0.5 = 1$$

→ No

$$-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 0.5 + 0.5 = 1$$

$$\therefore \text{Dest} = 1 \times \frac{4}{6} + 1 \times \frac{2}{6} = 1$$

For attribute: Shape

→ Human

$$-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.31 + 0.5 = 0.81$$

→ Animal

$$-\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

$$\therefore \text{Dest} = 0.81 \times \frac{4}{6} + 0 \times \frac{2}{6} = 0.54$$

For attribute: Material

→ Stone

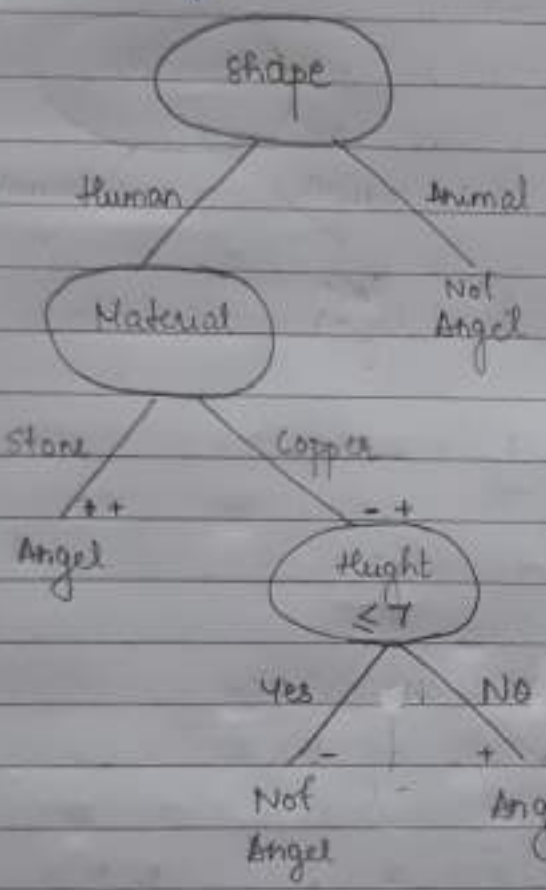
$$-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.39 + 0.53 = 0.92$$

→ Copper

$$-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.53 + 0.39 = 0.92$$

$\therefore D_{test} = 0.92 \times \frac{3}{6} + 0.92 \times \frac{3}{6} = \underline{\underline{0.96}}$

The disorder for the attribute shape is the least. Let us consider it first.



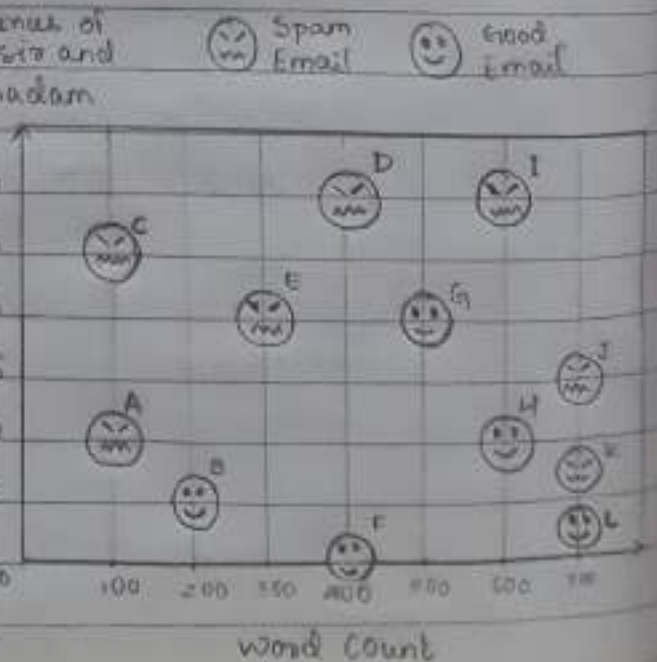
Statue	Classification	Shape	Material
1.	Angel	Human	Stone
2.	Angel	Human	Stone
3.	Not Angel	Human	Copper
4.	~~Not Angel~~	~~Animal~~	~~Copper~~
5.	~~Not Angel~~	~~Animal~~	~~Stone~~
6.	Angel	Human	Copper

Consider attribute: Material

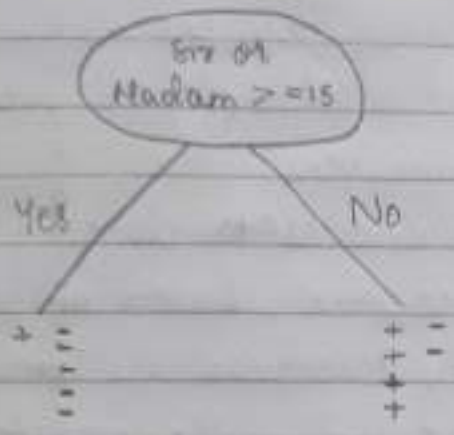
Statue	Classification	Height ≤ 7	Material
1.	~~Angel~~	7: Yes	Stone
2.	~~Angel~~	~~2.5~~: Yes	~~Stone~~
3.	Not Angel	7: Yes	Copper
4.	~~Not Angel~~	3: Yes	Copper
5.	~~Not Angel~~	8: No	~~Stone~~
6.	Angel	305: No	Copper

- **Example 4:**

After receiving yet another "Dear sir or madam" email, you decide to construct a spam filter. For your first attempt you decide to try using a k Nearest Neighbors model. You decide to classify spam according to 2 features: email word count and occurences of the words "sir" or "madam".



Based on the Decision tree (sir or madam >=15), draw the decision tree and calculate the disorder of Decision tree.



	Sir/Madam Occurences	Word Count	Email Type
A	10	100	Spam
B	5	200	Good
C	25	100	Spam
D	30	400	Spam
E	20	300	Spam
F	0	400	Good
G	20	500	Good
H	10	600	Good
I	30	600	Spam
J	15	700	Spam
K	8	700	Spam
L	2	700	Good

Disorder : Attribute sir or Madam Occurences

→ Yes

$$\frac{-1}{6}\log_2\frac{1}{6} - \frac{5}{6}\log_2\frac{5}{6}$$

$$= 0.43 + 0.22 = 0.65$$

→ No

$$-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.39 + 0.52 = 0.92$$

$$\therefore D_{test} = 0.65 \times \frac{6}{12} + 0.92 \times \frac{6}{12} = 0.325 + 0.46 = \underline{0.785}$$

- **SLE: Inductive Logical Programming:**

Inductive logic programming (ILP) is a uses logic programming as a uniform representation for examples, background knowledge and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesised logic program which entails all the positive and none of the negative examples.

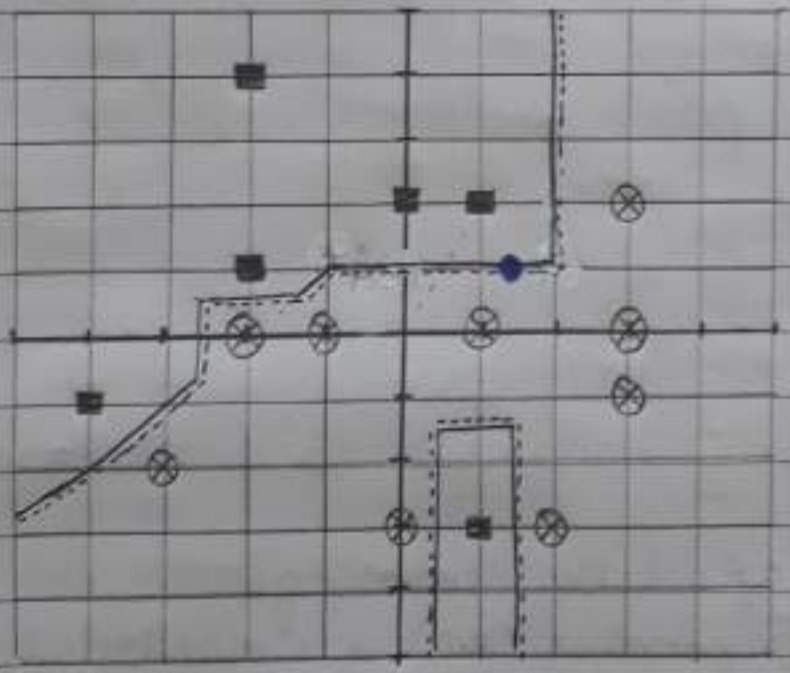
Positive Examples + Negative Examples + Background Knowledge ⇒ Hypothesis

Assignment.

Q: The Mario Bros have hired you to help them identify potential dangers during their frequent expeditions to explore strange new worlds, battle new worlds, battle evil monsters and attempt to rescue some princesses.

You decide to use Nearest Neighbours to classify new creatures that Mario might encounter, based on their positions on a map. In the map, friendly creatures are marked with a square and enemies are marked with monster icons.

a) On the following graph, draw the decision boundaries produced by 1-nearest neighbours.



b) How would the creature ● on the graph be classified with each of the following classifiers?

Nearest Neighbors	creature ●
1NN	Friend
3NN	Friend
5NN	Enemy
9NN	Enemy
15NN	Enemy