Chapter-01 and chapter-02

a) **What does AI mean?**

Artificial Intelligence, here ‘Artificial’ means ‘Man made’ and ‘Intelligence’ means ‘Thinking power’. So, AI means Man made thinking power.

1.

a) **What are the learning outcomes of AI?**

1. **Problem Solving:** AI enables effective problem-solving by analyzing data and generating solutions.
2. **Natural Language Processing:** Enables machines to understand and respond to human language.
3. **Machine Learning:** Allows systems to improve performance through experience and data.
4. **Data Analysis:** Processes vast amounts of data quickly, extracting meaningful insights.
5. **Decision Support:** Provides information to assist human decision-making processes
6. **Robotics:** Empowers machines to perform physical tasks, enhancing productivity.

b) **Define in your own words the following terms: agent, agent program, rationality, autonomy, deterministic and stochastic.**

1. **Agent:** An agent is anything that can be viewed as perceiving its environment through sensors and environment acting upon that environment through actuators.
   Example: A self-driving car navigating through traffic.

2. **Agent Program:** The set of rules or algorithms governing the agent's behavior and decision-making.
   Example: The algorithms guiding a chess-playing computer program.

3. **Rationality:** The ability of an agent to choose actions that maximize the likelihood of achieving its goals.
   Example: A smart thermostat adjusting temperature settings to save energy while maintaining comfort.

4. **Autonomy:** The degree to which an agent can operate independently without direct human intervention.
   Example: A drone surveying agricultural fields without constant human control.

5. **Deterministic:** A system or process where the outcome is entirely predictable from its initial state and inputs.
   Example: A simple calculator performing arithmetic operations.

6. **Stochastic:** Involving randomness or probability, where outcomes are not entirely predictable and may vary.
   Example: Weather forecasting, where predictions involve probabilities due to complex atmospheric systems.

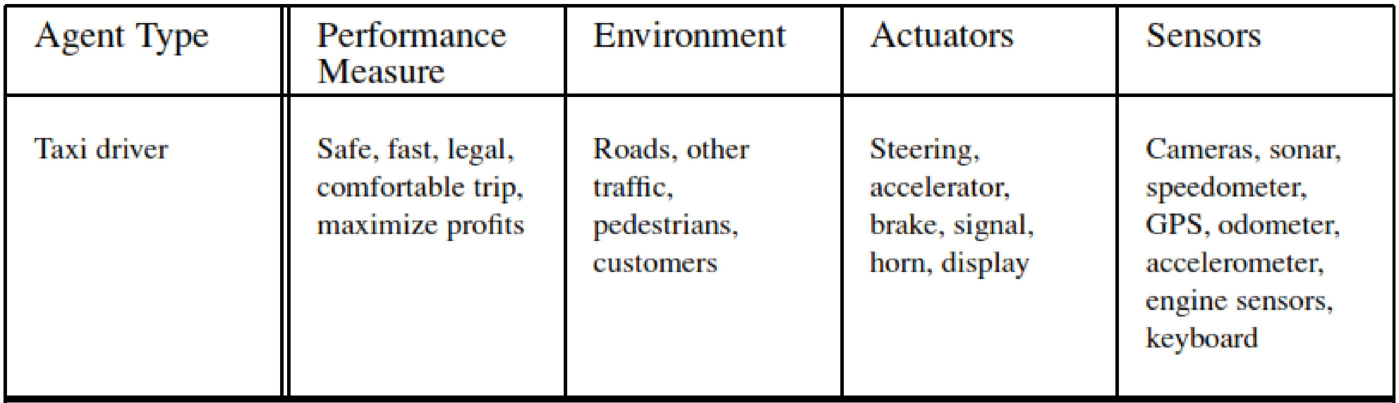
c) **What is PEAS in specifying the task environment? Illustrate and describe the structure of the model-based reflex agent.**

PEAS is a framework used in artificial intelligence to clearly define the parameters and objectives of a given task or problem, providing a structured way to specify an intelligent agent's characteristics and behavior within a particular environment.

PEAS stands for:

- **Performance measure**: This defines how the success of the agent in the task environment will be assessed.
- **Environment:** This describes the external context or the task environment in which the agent operates.
- **Actuators**: These are the mechanisms or components through which the agent affects the environment by taking actions.
- **Sensors:** These are the components that allow the agent to perceive or receive information about the environment.

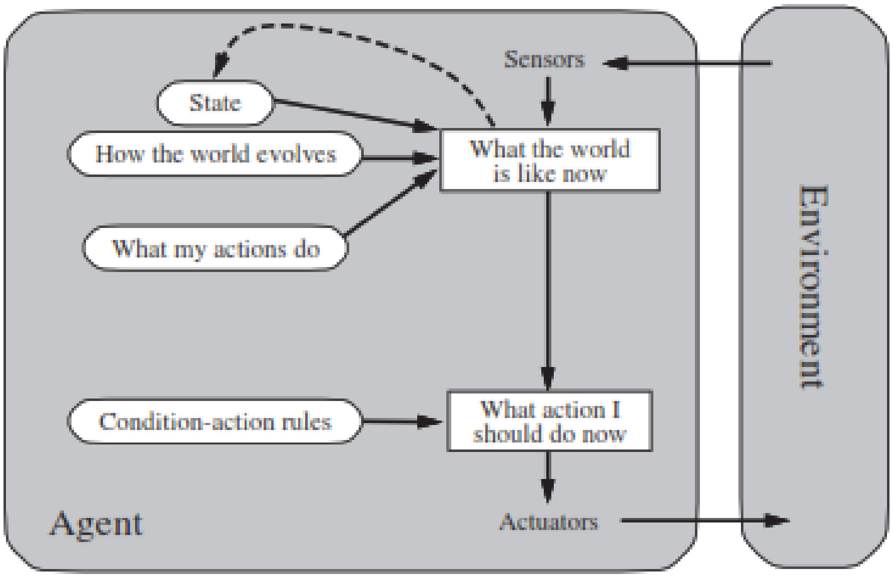
Example: PEAS description of the task environment for an automated taxi.



Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

- **Model-based reflex agents:**



1.

a) Define AI within 4 different paradigms with examples.

1. **Thinking Humanly**: The exciting new effort to make computers think.
   Example: Chatbots that engage in natural language conversation.

2. **Thinking Rationally:** The study of the computations that make it possible to perceive, reason and act.
   Example: Expert systems that use a knowledge base and rules to provide solutions or make decisions in specific domains, like medical diagnosis.

3. **Acting Humanly:** AI strives to mimic human behavior and actions.
   Example: Social robots designed to interact with people.

4. **Acting Rationally:** AI focuses on making optimal decisions to achieve specific goals, regardless of human-like reasoning.
   Example: Game-playing AI, such as Deep Blue for chess or AlphaGo for Go.

b) What is rationality? Differentiate between human agent and AI agent in terms of rationality.

- **Rationality:** The ability of an agent to choose actions that maximize the likelihood of achieving its goals.
  Example: A smart thermostat adjusting temperature settings to save energy while maintaining comfort.
- **Human Agent:-**
  Rationality: Limited by cognitive biases, emotions, and subjective interpretations of information.
- **AI Agent:-**
  Rationality: Strives for optimal decision-making based on programmed algorithms, devoid of emotional or biased influences.

c) "Do not measure the performance of an agent based on its behavior, Measure the performance of an agent based on what is wanted in the environment."-Justify these statements.

Imagine we have a robot that's supposed to clean a room. If we only look at how the robot moves around (its behavior), we might think it's doing a good job. But what really matters is whether the room gets clean, right?

So, the idea is, don't just watch what the robot does; check if it's actually achieving the goal, which is having a clean room. The quote is saying we should judge the robot based on what we want (a clean room) instead of just looking at how it behaves.

In simpler terms, it's like saying, "Don't judge a cleaning robot by how it moves, judge it by how clean the room gets." This way, we focus on the real purpose, not just the actions.

Chapter-03: Solving problem by searching.

1. **What is blind search? How to evaluate an algorithms performance? Compare search strategies in terms of the four evaluation criteria set.**

- **Blind Search:** Blind search, also known as uninformed search, is a search algorithm that explores a problem space without any knowledge about the structure or characteristics of the space.
- **Evaluating Algorithm Performance:** To evaluate the performance of search algorithms, four criteria are commonly used:
  - **Completeness:** Is the algorithm guaranteed to find a solution when there is one?
  - **Optimality:** Does the strategy find the optimal solution?
  - **Time complexity:** How long does it take to find a solution?
  - **Space complexity:** How much memory is needed to perform the search?

- ***Comparison of Search Strategies:***

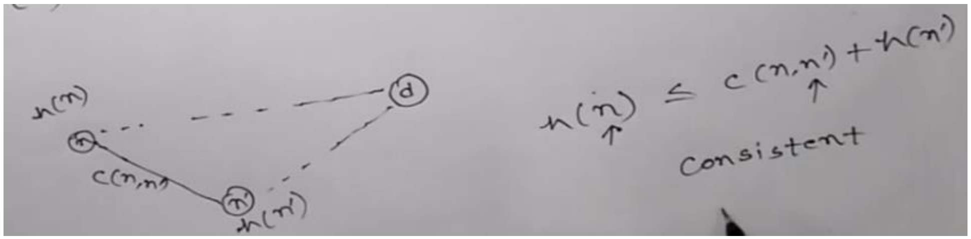
Search strategy	Completeness	Optimality	Time complexity	Space complexity
BFS	Yes	yes	$O(b^d)$	$O(b^d)$
DFS	No	No	$O(b^m)$	O(bm)
UCS	Yes	yes	$O(b^{1+(C*/ε)})$	$O(b^{1+(C*/ε)})$
DLS	Yes	no	$O(b^l)$	O(bxl)
IDDFS	Yes	Yes	$O(b^d)$	O(bd)
Bidirectional	Yes	Yes	$O(b^{d/2})$	$O(b^{d/2})$

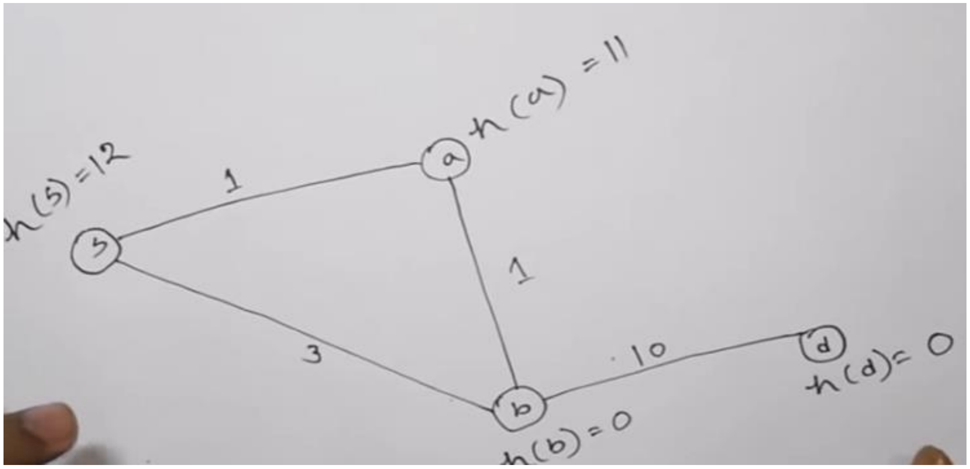
2. **Write the advantage of informed search. How to make heuristic function? Define and illustrate admissible heuristic and consistent heuristic for estimating optimality of A* search algorithm.**

- These are main advantages of informed search strategies:
  - **Faster convergence –**This focused search leads to exponentially faster goal discovery.
  - **Optimal solutions –** Algorithms like A* can guarantee optimality given admissible heuristics. This combined power of speed and accuracy is important for mission-critical applications.
  - **Scalability –** The ability to leverage heuristics lets informed search tackle much bigger problems with large search spaces and higher complexity.
  - **Efficiency –** Reduced nodes expanded means informed search uses computational resources very economically. This enables applications on resource-constrained devices.
  - **Tractability –** Many hard combinatorial problems like game playing become practically solvable only using informed search. The heuristics time complexity.
- **Make a heuristic function:**
  - **Understand the problem:** Grasp the problem's states, actions, and goals.
  - **Identify relevant information:** Determine what info is crucial for estimating the cost to reach the goal.
  - **Ensure admissibility**: Make sure the heuristic never overestimates the true cost to the goal.
  - **Ensure consistency**: The estimated cost to the goal via a successor state should never be less than the estimated cost from the current state to the goal.

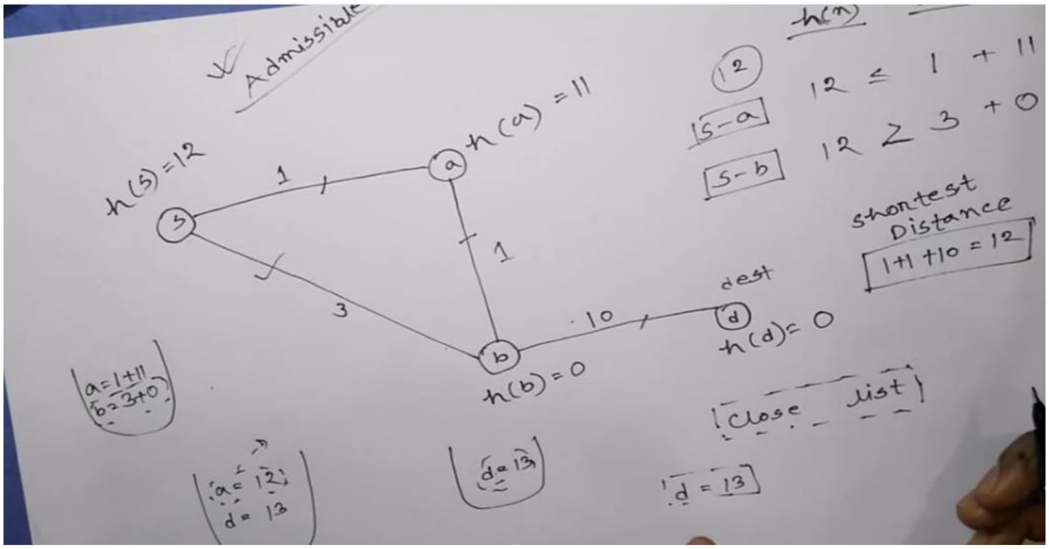
Admissible Heuristic:

- **Definition:** A heuristic that never overestimates the true cost to reach the goal.
- **Illustration of Admissible Heuristic**: h(n) <= actual cost, if this relation maintains for every node, then we say this Admissible Heuristics.





- **Consistent heuristics:**
  - **Definitions:** A heuristic that is admissible and satisfies a consistency condition.
  - **Illustrations:** This graph is not consistent hence it does not provide any optimal solution.



b) Prove that uniform-cost search and BFS with constant step costs are optimal when used with the GRPAH-SEARCH algorithm.

let's illustrate these properties with an example. Consider the following graph:

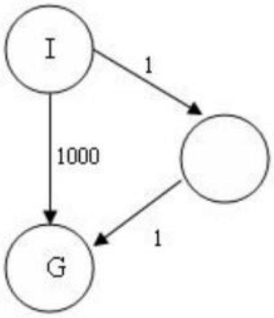
```
    A
   / \
  B   C
 / \   \
D   E   F
```

Let's assume that the step cost between each adjacent pair of nodes is 1.

- **Optimality of UCS**: Suppose we want to find the optimal path from node A to node F. UCS will expand nodes in the following order: A, B, C, D, E, F. It selects and expands nodes based on the lowest path cost. In this case, the optimal path is A → C → F with a total cost of 2.
- **Optimality of BFS with Constant Step Costs:** BFS will explore nodes level by level. In this case, it will expand nodes in the following order: A, B, C, D, E, F. Since all step costs are the same, BFS guarantees that the first time it reaches node F, it has found the shortest path.

c) Show a state space with constant step costs in which GRAPH-SEARCH using iterative deepening finds a suboptimal solution. Compare the four evaluation criteria set of several uninformed search strategies.

The state space that iterative deepening search with GRAPH-SEARCH algorithm will fail to find the optimal solution can be below figure where I is the initial state and G is the goal state**.**



- **Evaluation criteria set of several uninformed search strategies**

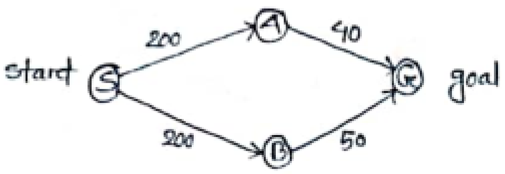
Already noted.

d) What is the Heuristic function of the informed search strategy? How to minimize the total estimated solution cost using the best-first search, A* search algorithm. Show the heuristic must be consistent for the optimal solution in the A* algorithm.

Heuristic function**:** Estimated cost of the cheapest path from node n to a goal node. f(n)=g(n)+h(n)

Minimize the total estimated cost using A* search**:**

By making A* admissible, we can minimize the total estimated solution cost.
For A* search algorithm the equation we use is
F(n)=g(n)+h(n)
Where g(n)= actual cost from start node to n
h(n)=estimated cost from n to goal node.
We know, when:
h(n)>= actual cost, then it is overestimation and when
h(n)<= actual cost then it is underestimation



Here,
g(A)= 200
g(B)= 200
actual cost, A→ G=40
actual cost, B→ G=50

case I: Overestimation (> actual cost)

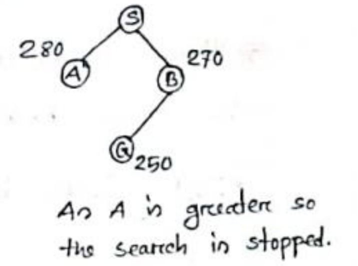
let, h(A) = 80

h(B)=70

now, f(A)= 200+80=280

f(B)=200 + 70=270

f(G)= g(A) + h(G)=250+0=250



As A is greater so the search is stopped.

case II: Underestimation (< actual cost)

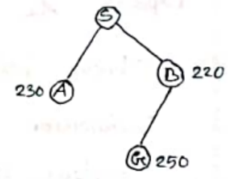
let, h(A) =30

h(B) = 20

f(A)=200+30=230

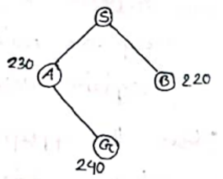
f(B)=200+20=220

f(G)=250



Here, A* doesn't stop as A Is less than G so it explores A then G.

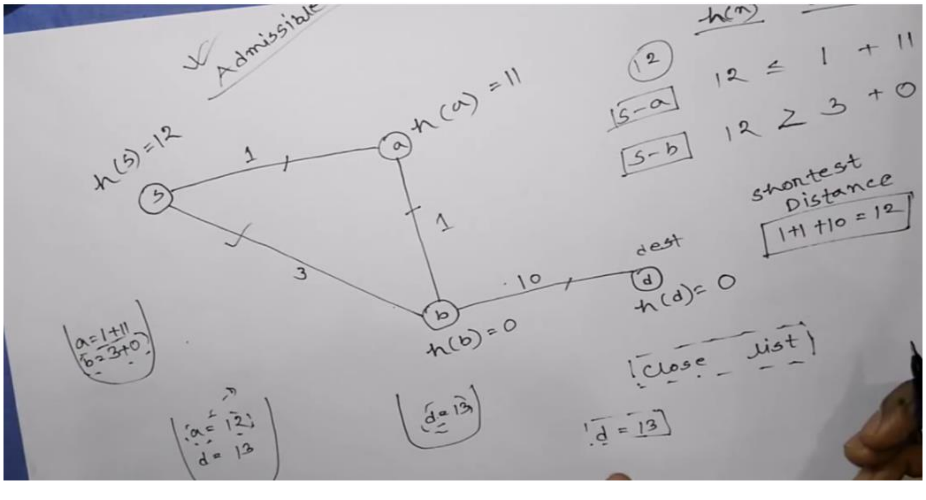
f(G)+g(G)+h(G)=240+0=240



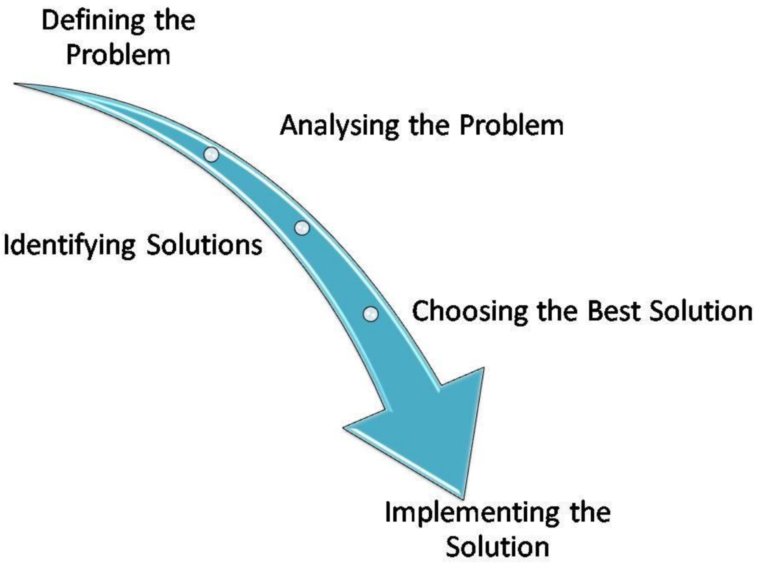
Here, 240 is less than 250 hence estimated cost is minimized.

The consistency condition ensures that the heuristic provides a meaningful estimate of the cost to reach the goal, and it plays a crucial role in guaranteeing the optimality of A*.

Heuristic must be consistent for optimal solution: The below example is not consistent and it does not provide any optimal solution. Hence, Heuristic must be consistent for optimal solution.



d) Explain the steps that a problem-solving agent uses to solve a problem with an example.

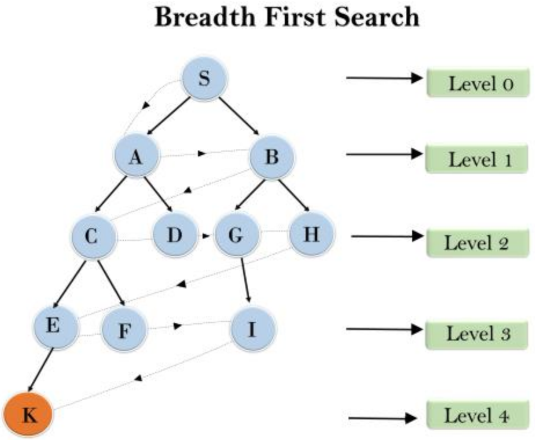


2.

a) Analyze BFS technique in terms of performance measure. Provide reasons to every measure.

It visits all the neighbors of a vertex before moving to the next level neighbors.

For example, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K.



BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

S---> A--->B---->C--->D---->G--->H--->E---->F---->I---->K

- **Performance measure:**
  - **Completeness:** Yes, BFS is complete if the branching factor is finite.
  - **Optimality:** Yes, BFS is optimal if the cost is a non-decreasing function of the depth.
  - **Time complexity:** $O(b^d)$ where b is the branching factor and d is the depth of the shallowest solution.
  - **Space complexity:** $O(b^d)$ for frontier nodes

b) "Exponential time or space complexity is very expensive"-Justify with an example in terms of time and memory requirements.

Exponential time or space complexity is considered expensive because it grows rapidly with the size of the input, making algorithms impractical for larger instances. The Towers of Hanoi and depth-first search examples illustrate the significant impact on both time and memory requirements when dealing with exponential complexities.

➢ **Justification for Exponential Time and space Complexity:**

Example: The Towers of Hanoi

Explanation: The Towers of Hanoi problem involves moving a tower of discs from one peg to another, with the restriction that only one disc can be moved at a time, and a larger disc cannot be placed on top of a smaller one. The minimum number of moves required is 2^n-1 is the number of discs.

Justification: The time complexity and space complexity of a naive recursive solution is exponential O(2^n) as each move leads to two recursive calls. This becomes impractical for large numbers of discs due to the rapid growth in computation time.

c) Compare IDDFS and DFS in terms of performance measure. Which one is preferred when the search space is large and the depth of the solution is unknown? Explain with an example.

Performance of DFS:

- **Completeness:** DFS is not complete if the search space has infinite depth or if there are loops in the graph, as it can get stuck in an infinite loop.
- **Optimality:** DFS does not guarantee optimality; it may find a non-optimal solution if the depth of the goal node is not the shallowest in the search tree.
- **Time complexity:** O(b^m), where b is the branching factor and m is the maximum depth of the search tree. In the worst case, DFS may explore all nodes.
- **Space complexity:** The space complexity of Depth First Search (DFS) is O(bm) in the worst case, where b is the branching factor, and m is the maximum depth of the search tree

Performance of IDDFS:

- **Completeness:** IDDFS is complete as it will eventually find a solution if one exists, exploring deeper levels in each iteration.
- **Optimality:** IDDFS is optimal when the path cost is a non-decreasing function of the depth because it explores shallower levels first.
- **Time complexity:** O(b^d), where b is the branching factor and d is the depth of the shallowest goal node.

Space complexity: O(bd), where b is the branching factor and d is the maximum depth reached in any iteration

Preferred for Large Search Spaces with Unknown Depth:

IDDFS Preferred when the depth of the solution is unknown and the search space is large.

Explanation: IDDFS combines the benefits of DFS with a systematic and controlled exploration of increasing depths. It gradually increases the depth limit, providing a balance between depth-first efficiency and breadth-first completeness.

Example: 8-Puzzle Problem:

Scenario: Solving an 8-puzzle where the optimal solution depth is unknown.

➢ Preference for IDDFS: IDDFS would be preferred due to its ability to perform a series of depth-first searches with increasing depth limits. It avoids the drawbacks of deep exploration while maintaining completeness. This is crucial when the optimal depth is uncertain, and an exhaustive DFS may be impractical.

3.

a) **Why do we use a local search strategy to address the optimization problem? What are the key advantages of local search algorithm?**

Local search strategies are used for optimization problems due to their efficiency in exploring large solution spaces, adaptability to different problem types, iterative improvement process, capability to handle non-convex spaces.

- **Advantages:**
  - Use very little memory.
  - Can find reasonable solutions in large or finite state.

b) **Show how the last configuration of 4-queens on a 4x4 board has fewer conflicts that the first configuration using a local search strategy in where conflicts mean there are no two queens on the same row, column or diagonal.**

Initial state:

	1	2	3	4
1	Q			
2		Q		Q
3				
4	Q			

- Queen at (1,1) conflicts with the queen at (2,2) diagonally.
- Queen at (2,2) conflicts with the queen at (2,4) Horizontally.
- Queen at (1,1) conflicts with the queen at (4,1) vertically.

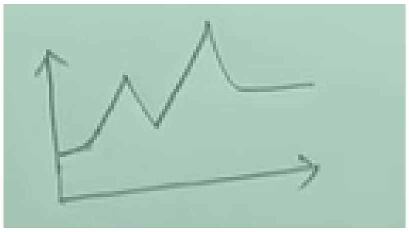
Final state:

	1	2	3	4
1		Q		
2				Q
3	Q			
4			Q	

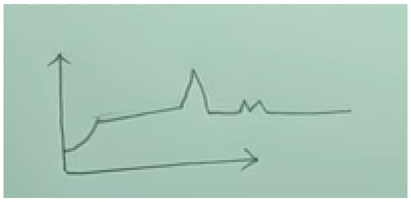
- No queen conflict with others.

c) **What are the reasons and problems of the hill-climbing algorithm for getting stuck? How to escape these problems using the Simulated-annealing search algorithm?**

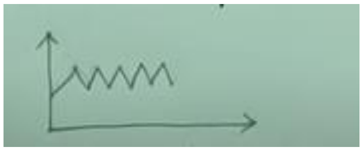
- **Hill-climbing often gets stuck for the following reasons:**
- **Local maxima:** Algorithm can't go higher.



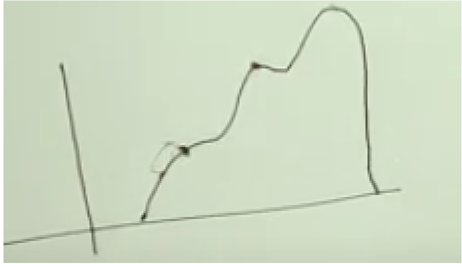
- **Plateaux:** Area where the evaluation function is flat.



- **Ridges:** Search may oscillate slowly.

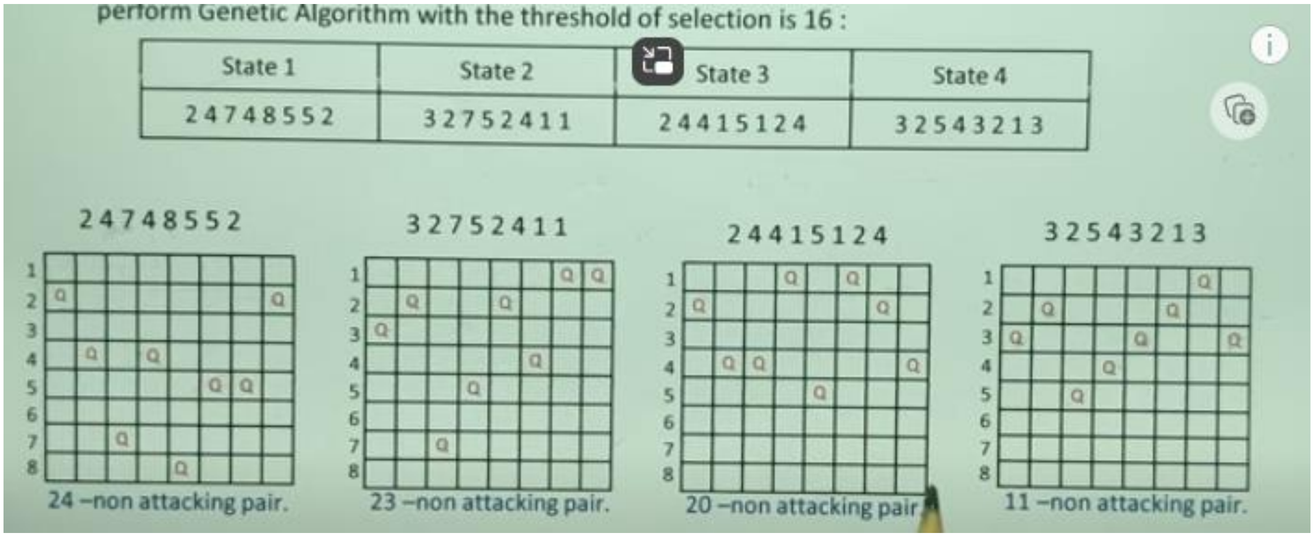


- Escape these problems using the Simulated-annealing search algorithm:
  - Hill-Climbing always goes uphill, looking for the best solution nearby. Sometimes, it might get stuck on a hill and miss a better solution on a different hill. Simulated Annealing is like Hill-Climbing, but with a trick. It sometimes goes downhill a bit, allowing it to explore other hills. This helps avoid getting stuck and find a better overall solution.



d) Illustrate and explain the genetic algorithm using digit strings representation of 8-queen states.

Steps in the Genetic Algorithm:



perform Genetic Algorithm with the threshold of selection is 16 :

State 1	State 2	State 3	State 4
24748552	32752411	24415124	32543213

24748552

1								
2	Q							Q
3								
4		Q		Q				
5						Q	Q	
6								
7			Q					
8					Q			

24 –non attacking pair.

32752411

1							Q	Q
2		Q			Q			
3	Q							
4						Q		
5				Q				
6								
7			Q					
8								

23 –non attacking pair.

24415124

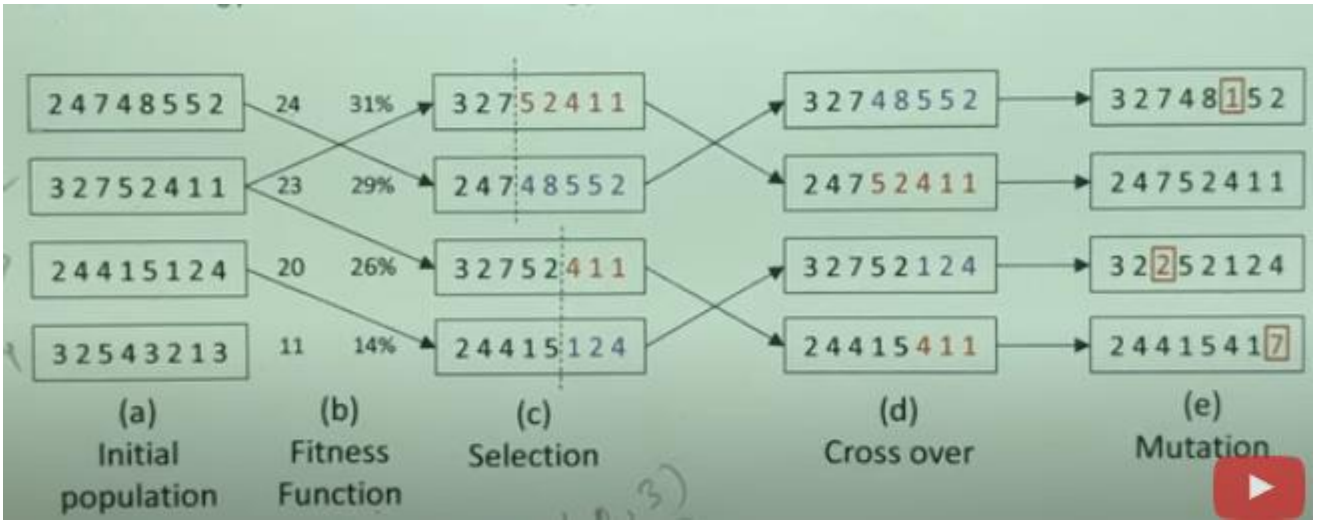
1				Q		Q		
2	Q						Q	
3								
4		Q	Q					Q
5					Q			
6								
7								
8								

20 –non attacking pair

32543213

1							Q	
2		Q				Q		
3	Q				Q			Q
4				Q				
5			Q					
6								
7								
8								

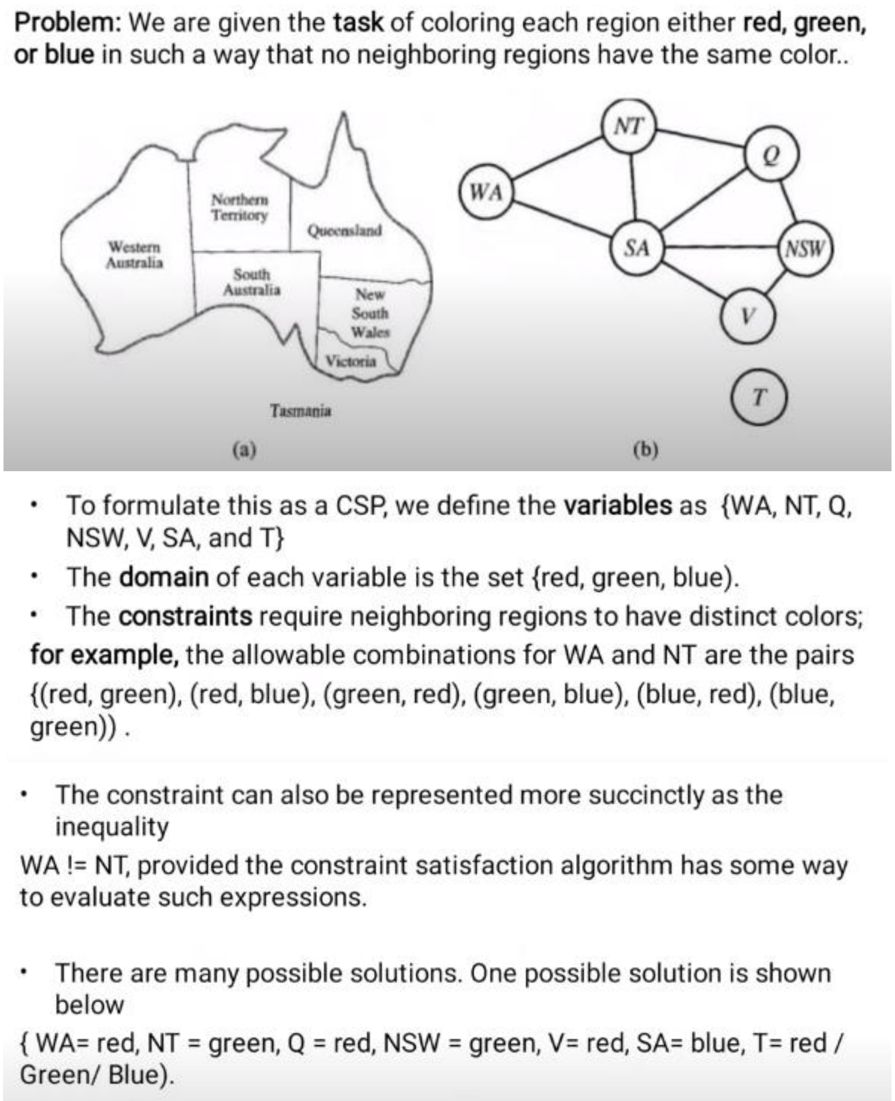
11 –non attacking pair.



a) **Define constraint satisfaction problems. Represent the map coloring problem with a constraint graph.**

A problem is solved when each variable has a value that satisfies all the constraints on that variable is called CSP.

Map coloring:



Problem: We are given the **task** of coloring each region either **red, green, or blue** in such a way that no neighboring regions have the same color..

- To formulate this as a CSP, we define the **variables** as {WA, NT, Q, NSW, V, SA, and T}
- The **domain** of each variable is the set {red, green, blue).
- The **constraints** require neighboring regions to have distinct colors;

for example, the allowable combinations for WA and NT are the pairs {(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)) .

- The constraint can also be represented more succinctly as the inequality

WA != NT, provided the constraint satisfaction algorithm has some way to evaluate such expressions.

- There are many possible solutions. One possible solution is shown below

{ WA= red, NT = green, Q = red, NSW = green, V= red, SA= blue, T= red / Green/ Blue).

.

b) Define robot. Briefly describe different types of robot hardware.

A robot is a programmable machine designed to perform tasks autonomously or semi-autonomously, typically in the physical world.

Robot hardware:

1. **Actuators:** Motors, servo motors.
2. **Sensors:** Vision sensors, infrared sensors, ultrasonic sensors, touch sensors.
3. **Power Supply:** Batteries.
4. **Processors/Controllers**: Microcontrollers, CPUs.
5. **Communication Modules:** Wireless modules, sensor interfaces.
6. **Frames/Chassis:** Structural components, wheels, tracks, legs.

Chapter-07 and chapter-08.

4. Define logical agent. Write down a simple algorithm of a generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

- **Logical agents** are intelligent agents that use formal logic as a foundational framework for reasoning and making decisions.
- **Algorithm for a Generic Knowledge-Based Agent:**

```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

5. What is propositional logic? Drives a propositional logic from Wumpus world is a cave consisting of rooms connected by passageways.

- **Propositional Logic:** It is a declarative statement that is either true or false but not both.
- <u>.Here are the key components of the propositional logic for the Wumpus World:</u>
  - **Symbols and Variables:**
    - ✓ P(i, j): The proposition that there is a pit at location (i, j).
    - ✓ W(i, j): The proposition that the Wumpus is at location (i, j).
    - ✓ G(i, j): The proposition that there is gold at location (i, j).
    - ✓ S(i, j): The proposition that the agent is at location (i, j).
    - ✓ Breeze(i, j): The proposition that there is a breeze at location (i, j).
    - ✓ Stench(i, j): The proposition that there is a stench at location (i, j).
    - ✓ Glitter(i, j): The proposition that there is a glitter at location (i, j).
    - ✓ Arrow: The proposition that the agent has an arrow.
    - ✓ WumpusAlive: The proposition that the Wumpus is alive.
  - **Logical Rules:**
    - **Axioms:**
    - **Percept Axioms:**

- ✓ Breeze(i, j) ⇔ ∨(P(i+1, j), P(i-1, j), P(i, j+1), P(i, j-1))
- ✓ Stench(i, j) ⇔ ∨(W(i+1, j), W(i-1, j), W(i, j+1), W(i, j-1))
- ✓ Glitter(i, j) ⇔ G(i, j)

- **Location Axioms:**
  - ✓ S(i, j) ⇔ ¬P(i, j) ∧ ¬W(i, j)

  Safe Locations:
  - ✓ Safe(i, j) ⇔ S(i, j) ∨ ¬(Breeze(i, j) ∨ Stench(i, j))
- **Action Axioms:**
  - ✓ Grab ⇔ Glitter(i, j) ∧ S(i, j)
  - ✓ Shoot ⇔ Arrow ∧ WumpusAlive ∧ ∃(i, j)(S(i, j))
  - ✓ Forward ⇔ S(i, j) ∧ ¬WumpusAlive
  - ✓ TurnRight
  - ✓ TurnLeft
  - ✓ Climb ⇔ S(1,1) ∧ ¬WumpusAlive ∧ ¬Gold(i, j)
- **Wumpus Fate:**
  - ✓ WumpusAlive ⇔ ¬(Shoot ∧ ¬∃(i, j)(W(i, j)))

Chapter-22: NLP

a) **What is neural network? Describe the multilayer perceptron neural network.**

A neural network is like a computer brain that learns stuff.

A Multilayer Perceptron (MLP) is a type of neural network characterized by having three or more layers of nodes: an input layer, one or more hidden layers, and an output layer. Here's a brief overview of each layer:

- ➢ **Input Layer:** This layer receives the initial data or features and passes them to the hidden layers. Each node in this layer represents a feature of the input.
- ➢ **Hidden Layers:** These layers process the input data through weighted connections and activation functions. The "hidden" nature comes from the fact that the intermediate computations in these layers are not directly observable from the outside. Multiple hidden layers allow the network to learn complex representations.
- ➢ **Output Layer:** This layer produces the final output of the network. The number of nodes in this layer depends on the problem type (e.g., binary classification, multi-class classification, regression).

b) **What is a language model? Describe the N-gram character model.**

A language model is like a computer program that understands and talks in human language.

The N-gram character model is a type of language model that predicts the next letter or character in a sentence by looking at the letters that came before it. It learns from a lot of examples to figure out which letters often go together.

Let's say we have the sentence "I love to thought" In a big dataset, the N-gram character model might learn that when it sees the sequence "I love to," the next character is often "t"

So, if you input "I love to," the model might predict the next character as "t" because it learned from lots of examples where those letters follow each other

✪ Difference between Uninformed and Informed search.

Parameters	Informed Search	Uninformed Search
Known as	It is also known as Heuristic Search.	It is also known as Blind Search.
Using Knowledge	It uses knowledge for the searching process.	It doesn’t use knowledge for the searching process.
Performance	It finds a solution more quickly.	It finds solution slow as compared to an informed search.
Completion	It may or may not be complete.	It is always complete.
Cost Factor	Cost is low.	Cost is high.
Examples of Algorithms	• Greedy Search • A* Search	• Depth First Search (DFS) • Breadth First Search (BFS)

Chapter-13

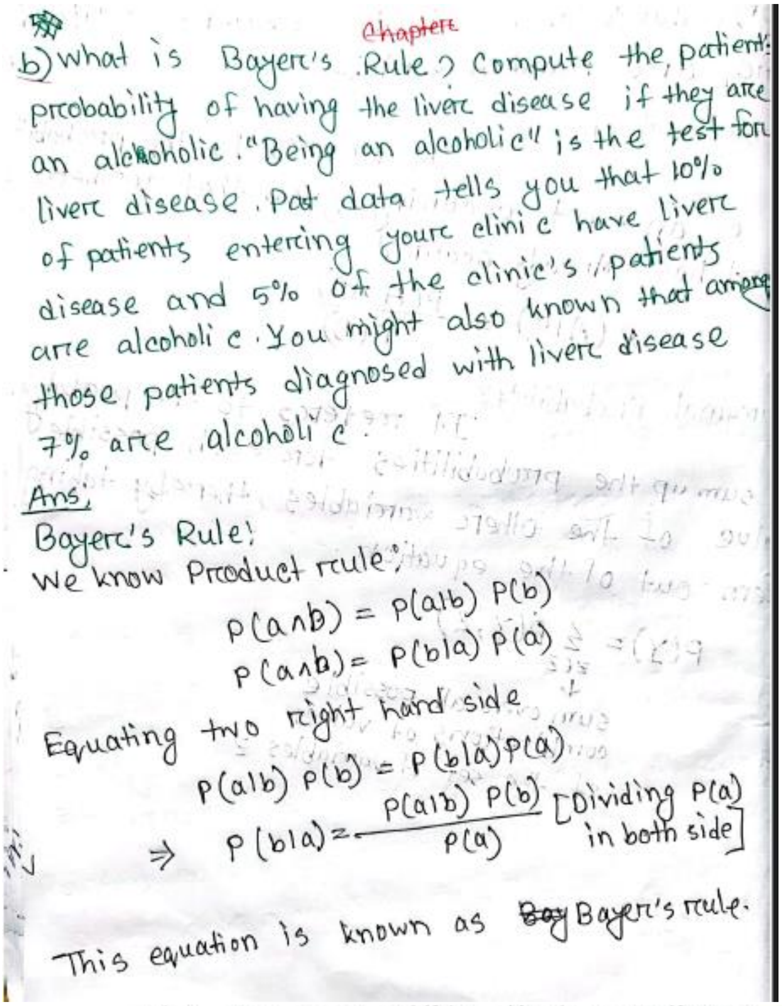
✪ Define the following terms in your own words. Conditional probability, Marginal Probability, Normalization and Independent probability.

Conditional Probability: The chance of something happening given that something else has already happened.
Example: What's the chance of having ice cream (A) given that it's sunny (B)? Written as P(A | B).

Marginal Probability: The chance of one thing happening by itself, without considering other stuff.
Example: What's the chance of picking a red card (A) from a deck, regardless of anything else? Written as P(A).

Normalization: Making sure all chances add up to 100% (or 1).
Example: If the chances of rain, sun, and clouds are 30%, 50%, and 20%, normalization makes sure they add up to 100%.

Independent Probability: Things happening on their own, not affecting each other.
Example: Tossing a coin twice. The first toss doesn't change the chance of getting heads on the second toss. They're independent.



Chapter

b) What is Bayer's Rule? Compute the patient's probability of having the liver disease if they are an alcoholic. "Being an alcoholic" is the test for liver disease. Pat data tells you that 10% of patients entering your clinic have liver disease and 5% of the clinic's patients are alcoholic. You might also known that among those patients diagnosed with liver disease 7% are alcoholic.

Ans;

Bayer's Rule:

We know Product rule;

$$P(a \wedge b) = P(a|b)\,P(b)$$

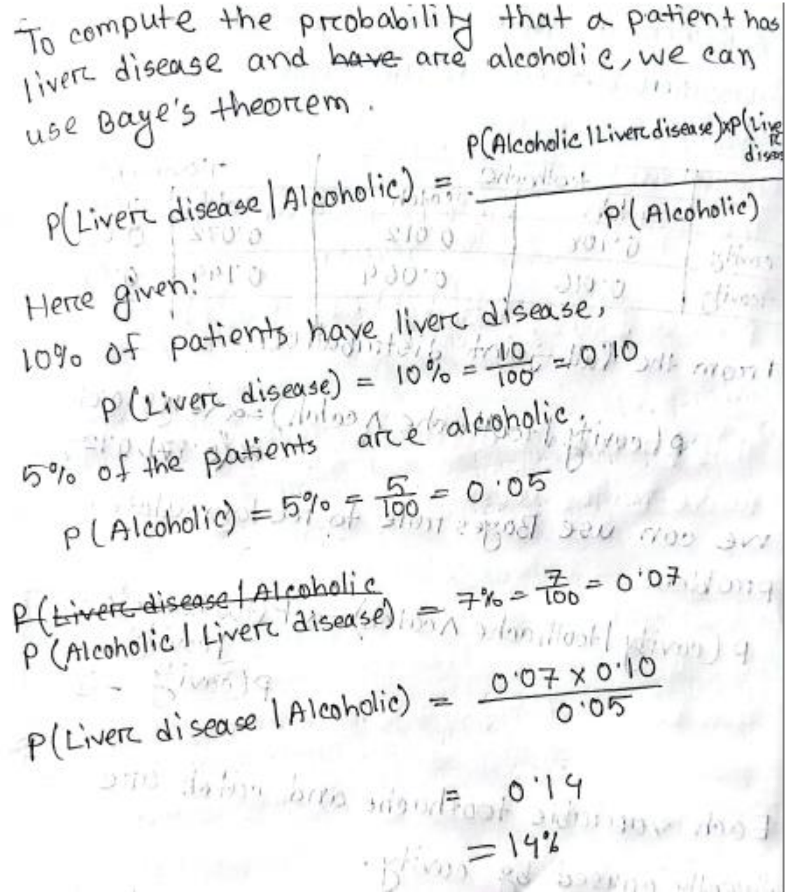
$$P(a \wedge b) = P(b|a)\,P(a)$$

Equating two right hand side

$$P(a|b)\,P(b) = P(b|a)\,P(a)$$

$$\Rightarrow P(b|a) = \frac{P(a|b)\,P(b)}{P(a)}$$ [Dividing P(a) in both side]

This equation is known as ~~Bay~~ Bayer's rule.



To compute the probability that a patient has liver disease and ~~have~~ are alcoholic, we can use Baye's theorem.

$$P(\text{Liver disease}|\text{Alcoholic}) = \frac{P(\text{Alcoholic}|\text{Liver disease}) \times P(\text{Liver disease})}{P(\text{Alcoholic})}$$

Here given:

10% of patients have liver disease,

$$P(\text{Liver disease}) = 10\% = \frac{10}{100} = 0.10$$

5% of the patients are alcoholic.

$$P(\text{Alcoholic}) = 5\% = \frac{5}{100} = 0.05$$

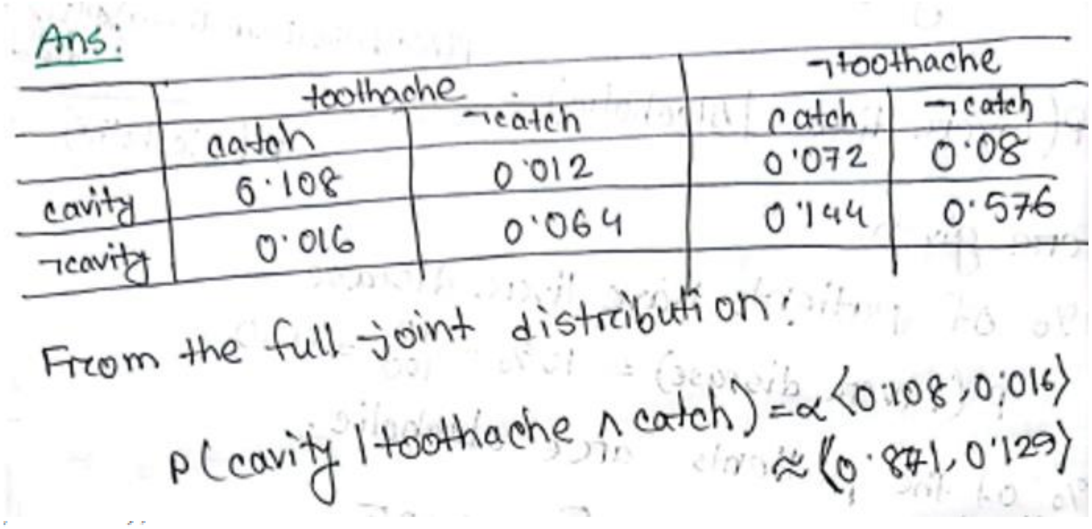
~~P(Liver disease | Alcoholic~~
$$P(\text{Alcoholic}|\text{Liver disease}) = 7\% = \frac{7}{100} = 0.07$$

$$P(\text{Liver disease}|\text{Alcoholic}) = \frac{0.07 \times 0.10}{0.05}$$

$$= 0.14$$

$$= 14\%$$

c) Design a naïve Bayes Model, Bayesian classifier based on the dentistry example.



Ans:

	toothache		¬toothache	
	aatch	¬catch	catch	¬catch
cavity	6·108	0'012	0'072	0'08
¬cavity	0'016	0'064	0'144	0'576

From the full joint distribution:

$$p(cavity \mid toothache \wedge catch) = \alpha \langle 0{\cdot}108, 0{;}016 \rangle \approx \langle 0{\cdot}871, 0{'}129 \rangle$$

$$\mathbf{P}(Cavity \mid toothache \wedge catch) = \alpha \, \langle 0.108, 0.016 \rangle \approx \langle 0.871, 0.129 \rangle \, .$$

We can try using Bayes' rule to reformulate the problem:

$$\mathbf{P}(Cavity \mid toothache \wedge catch) \\ = \alpha \, \mathbf{P}(toothache \wedge catch \mid Cavity) \, \mathbf{P}(Cavity) \, . \qquad (13.16)$$

irrelevant.[5] Mathematically, this property is written as

$$\mathbf{P}(toothache \wedge catch \mid Cavity) = \mathbf{P}(toothache \mid Cavity)\mathbf{P}(catch \mid Cavity) \, . \qquad (13.17)$$

This equation expresses the **conditional independence** of *toothache* and *catch* given *Cavity*. We can plug it into Equation (13.16) to obtain the probability of a cavity:

$$\mathbf{P}(Cavity \mid toothache \wedge catch) \\ = \alpha \, \mathbf{P}(toothache \mid Cavity) \, \mathbf{P}(catch \mid Cavity) \, \mathbf{P}(Cavity) \, . \qquad (13.18)$$

The general definition of **conditional independence** of two variables X and Y, given a third variable Z, is

$$\mathbf{P}(X, Y \mid Z) = \mathbf{P}(X \mid Z)\mathbf{P}(Y \mid Z) \, .$$

In the dentist domain, for example, it seems reasonable to assert conditional independence of the variables *Toothache* and *Catch*, given *Cavity*:

$$\mathbf{P}(Toothache, Catch \mid Cavity) = \mathbf{P}(Toothache \mid Cavity)\mathbf{P}(Catch \mid Cavity) \, . \qquad (13.19)$$

$$\mathbf{P}(X \mid Y, Z) = \mathbf{P}(X \mid Z) \quad \text{and} \quad \mathbf{P}(Y \mid X, Z) = \mathbf{P}(Y \mid Z)$$

assertion in Equation (13.19), we can derive a decomposition as follows:

$$\begin{aligned} &\mathbf{P}(Toothache, Catch, Cavity) \\ &= \mathbf{P}(Toothache, Catch \mid Cavity)\mathbf{P}(Cavity) \quad \text{(product rule)} \\ &= \mathbf{P}(Toothache \mid Cavity)\mathbf{P}(Catch \mid Cavity)\mathbf{P}(Cavity) \quad \text{(using 13.19).} \end{aligned}$$