Discuss the properties of environment. How does the vacuum cleaner perceive its the environment? what sensing mechanisms are the employed and their role in detecting dirt, obstacles, and other relevant feature. What are the primary actuators used in the vacuum cleaner, and how do they facilitate its the vaccuum cleaner, and how do they facilitate its movement and cleaning action.

Vacuum Cleaner problem :-

- Observable :- partially (Can Sense Current room only)
- Deterministic :- Same action always leads to Same result.
- Episodic :- Each room is a Separate task.
- Static :- Environment doesn't change on its own.
- Discrete :- finite number of locations and Action.
- Single - Agent :- No other agents involved.

How it Senses :-

- Detects room ID (A or B)
- Checks dirt status (Dirty / Clean)
- Advanced models detect obstacles, Stairs, Walls.

Sensors used :-

Sensor	Role
Dirt Sensor	Detect dirt in the current area.
Bump Sensor	Detect collision with object.
Cliff Sensor	prevents falling off edges
Infrared	Detects distance to walls.

Decision - making process :-

1) Sense the environment (location + dirt)
2) Decide based on logic / rules.
3) Act by sucking dirt or moving.
4) Repeat until all areas are clean.

Conclusion :-

The Vacuum Cleaner is an intelligent agent that perceives, acts, and learn (in smart version) to clean efficiently using basic sensors and logic in a simple environment.

2) Explore the Search Space using Search Strategies and formulate the problem Components for the 8-queens problems using the following information: place 8 queens on a Chessboard Such as that the none of the queen's attack any of the others.

Goal:- place 8 queens on 8x8 Chessboard Such that no two queen attack each other.

problems Components (formulation):-

Component	Description
Initial State	Empty Chessboard (no queens placed)
State Space	All possible placements of queens (with Conflicts)
Goal test	All 8 queens placed with no attacks (valid board).
path Cost	Not relevant here (all steps = 1)

Search Strategies:-

Strategy	Description
Backtracting	place queens row by row, backtrack if Conflicts

Constraint Satisfaction Model as CSP. Use forward Check + MRV (Minimum Remaining Value)

Conclusion:-

The 8-queens problems is a classic CSP (Constraint problem). Using backtracking searching with pruning or heuristic algorithm.

Solve the water jug problem: You are given 2 jugs, a 4-gallon one and 3-gallon one. Neither has any measuring marker on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into 4-gallon jug?

Goal:-

A 4-gallon jug.

A 3-gallon jug.

A pump to fill jugs.

Allows actions: fill, Empty, Transfer between jugs.

Assumptions:-

- You can fill a jug completely from the pump.
- You can empty a jug entirely on the ground.

Result :-

Now, 4-gallon Jug has exactly 2 gallons of water.

State Transitions :-

You can also write the states as (x, y) where.

- x = amount in 4-gallon jug.
- y = amount in 3-gallon jug.

- Show how BFS & DFS work on the search tree for given state space graph.

Breadth - first Search (BFS):

BFS explores node level by level from the root.

A → B → C → D → E → F

Explanation :-

- Start at A
- visit A's children → B, C
- visit B's child → D
- visit C's children → E, F

Depth - first Search (DFS):-

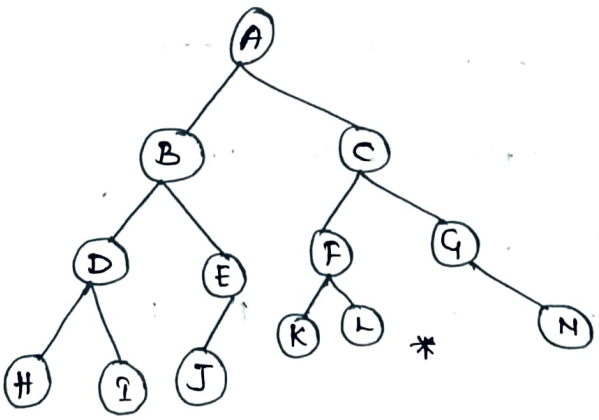
DFS explores as far as possible along a before backtracking.

A → B → D → C → E → F

Explanation:-

- Start at A
- Go to B
- Go to D (B's child)
- Backtrack to A → Go to C.
- Go to E. then F

Discuss uninformed Searching Strategies BFS and DFS with its advantages and disadvantages using the following graph to reach the goal L.



1) Breadth - first Search (BFS)

A → B → C → D → E → F → G → H
→ I → J → K → L

Goal L is found at this point.

* Completeness: BFS guarantees that it will find the shallowest solution (i.e, shortest path to L).
* Optimality: BFS always finds the optimal solution if all step costs are equal.

2) Depth - first Search (DFS)

A → B → D → H → I → J → E → C → F →
K → L

Goal L is found here, but later than in BFS.

* Low memory Requirement: Stores only current path and unexpanded Sibling.
* fastin Some cases: Can find Solution without exploring entire tree (if lucky).

A Customer wants to travel from the one location to another using OLA Cab booking mobile application. The Customer Can Selection, prime and so on based on his Comfort to reach the destination. Identify that type of problem-Solving agent Can be used and write the pseudocode for the above problem.

problem-Solving Agent Type:-

Why?

- The Customer has a goal: reach a Specific destination.
- The agent select action (cab type/routes) to achieve this goal.
- the agent evaluates possible outcome of action (Comfort, price, ETA. etc).

Agent Architecture involved:-

- percept: user input (pickup, drop location, preference).
- Action: Select cab type, assign driver, navigate route.
- Goal: Reach the destination efficiently.

pseudocode for Goal-Based Agent:-

```
function BookCab(pickup, destination,
                 userpreferences):
    goal <- Reach destination
    CabTypes <- [mini, micro, Sedan, Shared,
                 prime]
    availableCabs <- GetAvailableCabs(pickup)
    filteredCabs <- []
if filteredCabs is empty:
    return "No cabs available. Try
            again later".

bestCab <- null
bestScore <- ∞

for Cab in filteredCabs:
    route <- findRoute(pickup, destination)
    time <- Estimate(route)
    Cost <- Estimatefare(cab.type, route)
    Score <- Evaluate(cab, time, Cost,
                      User preferences)

    if Score < bestScore:
        bestScore <- Score
        bestCab <- Cab

return "Cab booked" + bestCab.type + "ETA:"
        bestCab.eta.
```