

Report No.: 01

**Submitted by:**

Sakar KC

BEI IV/I

Roll no.: 08

Subject: Artificial Intelligence

**Submitted To:**

Department of Electronics, Communication and Information Engineering

Date: 2079-04-30

# LAB 1: VACUUM WORLD

## OBJECTIVES:

1. To get familiar with problem solving.
2. To understand the concept of agent.

## THEORY:

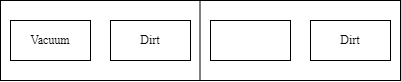
## **Problem Solving:** It is the area of finding answers for unknown situation. Its process are:

1. Understanding
2. Representation
3. Formulation
4. Solving

Its types are:

1. Simple: It can be solved using deterministic approach
2. Complex: Lack of full information

Example:

1. **Vacuum World**
   1. Understanding:
      1. n rooms and 1 vacuum cleaner.
      2. Dirt can be in any room. It has to be cleaned.
      3. Vacuum cleaner is present in any one room at a time.
      4. Goal: clean all rooms
   2. Representation**:**
   3. ****Formulation**:**
      1. Possible action: move left, move right, and clean dirt
   4. Solving**:**

## **vacuumWorldIMPLEMENTATION:**

Python Code: https://github.com/sakarkc2122/practicalAI

**DISCUSSION:**

Firstly, we learned the concepts of intelligent agents and its type. We understand that agent function maps from percept histories to actions. Thus, we take the vacuum world problem to understand the agent more clearly. In the code: ‘01vacuumWorld’, I have used the rational agent approach, i.e, simple reflex agent. The function ‘vacuumWorld()’ is the agent function in the program. The function like ‘clean()’, ‘moveRight()’, and ‘moveLeft()’, are the actuator in this program. Finally, condition-action (if-then) rules are figured out for the problem and then implemented in the ‘vacuumWorld()’.

Algorithm:

1. At first, problem instances (initial state & goal state) are defined.
2. Find the room with a vacuum
   1. If the vacuum is in 1st room, clean the room and move right and clean until it reaches the last room.
   2. If the vacuum is in the last room, clean the room and move left and clean until it reaches the first room.
   3. If vacuum is in between first and last
      1. If (index of room with a vacuum - 0) is less than (number of room - room with vacuum), then move left and clean the room.
      2. Else, move right and clean the room.

**CONCLUSION:**

After a brief study and analysis of the agents and their type, I implemented a simple reflex agent for the vacuum world problem. I have used the rational agent approach. Thus, this program gives the number of states that a vacuum takes to clean all the rooms (n rooms). The output also provides every next state after any one actuator acts. Thus it can be used for real-world vacuum cleaner robots. The performance measure of this program is the number of optimal state vacuums taken to clean all the rooms.

# LAB 2: DEPTH-FIRST SEARCH & BREAD-FIRST SEARCH

## OBJECTIVES:

1. To get familiar with searching.
2. To understand the concept of basic searching strategies, like depth-first search & breadth-first search.

## THEORY:

**Depth-First Search (DFS):**

It proceeds down a single brah of the tree at a time. It expands the root node, then the leftmost child of the root node, and so on. It always expands a node at the deepest level of the tree. Only when the search hits a dead end, search backtrack is done and expands node at higher levels.

It uses stack to keep track of nodes, i.e LIFO approach.

Its time complexity is O(bm) and

Its space complexity is O(bm)

where b is the maximum branching factor of the search tree and m is the maximum depth of the state space.

**Breadth-First Search (BFS):**

It proceeds level by level down the search tree. Starting from the root node, (initial state) explores all children of the root node left to right; if no solution is found, expand the root node's first (leftmost) child. Then expand the second node and its children similarly.

BFS forms a queue where nodes are expanded in FIFO manner.

Its time complexity is O(bd+1) and

Its space complexity is O(bd+1)

where b is the maximum branching factor of the search tree and d is the level of goal node in the search table.

## **IMPLEMENTATION:**

Python Code: https://github.com/sakarkc2122/practicalAI

**DISCUSSION:**

Firstly, we learned the concepts of searching. Its different terminologies like: search space, problem instance (initial state & goal state), problem space, searching strategies, search tree, etc. Among many searching strategies, we implemented the basic DFS & BFS in code. *‘02DFS.py’* and *‘02BFS.py’* are the two programs for DFS and BFS respectively in the repository. The programs starts with a defined graph and its output shows in what order nodes in the graph are visited.

**CONCLUSION:**

After a brief study of searching techniques, DFS and BFS are the basic searching technique to get started. BFS and DFS will get the result eventually if the searched node is present in the graph. But if we choose them in the wrong application, they will take high time and memory management. BFS & DFS are used in path-finding algorithms, like BFS in the Ford-Fulkerson algorithm and DFS, in solving puzzles with only one solution, such as a maze or a sudoku puzzle.

# LAB 3: PROPOSITIONAL LOGIC

## OBJECTIVES:

1. To get familiar with searching.
2. To understand the concept of basic searching strategies, like depth-first search & breadth-first search.

## THEORY:

