

**Artificial Intelligence Systems**

Lab Report # 02

**Submitted by:** Ali Hamza

**Registration No:** B23F0063AI106

**Submitted to:** Dr. Mohsin

Date: 25th February , 2025

# Lab Task:

Develop a simple reflex agent program in Python for the **vacuum-cleaner** **world problem**. Your agent must defines following things

* States
* Goal State
* Goal Test
* Actions
* Transition Mode
* Path Cost

**Some Pre-defined (Model ) Knowledge for agent**

* **Actions** : Only 0 or 1 where 0 means CLEAN and 1 means DIRTY
* **States :** Your agent have only 3 states (A/B/C)
* **Goal state:** {“A” : 0 , “B” : 0 , “C “: 0}
* **States Sequence :** A , B and C are rooms Connected in following order : A→B and B→C
* **Path Cost :** +1 for every Action (Either from dirt to clean OR moving from one room to another)

**Inputs of Agent**

1. Enter LOCATION (Initial Vaccum placement ) A/B/C in captial letters.
2. Enter Status of Current Location 0/1 accordingly.
3. Vacuum Cleaner senses the status of the other rooms before performing any action, also known as Environment sensing. So Give Status of other rooms as input. (0/1)

**Output of agent**

For each possible initial state(as input ), the program returns a sequence of actions that leads to the goal state, along with the path cost.

*"""  
Reflex Agent Program for Vacuum Cleaner World Problem  
  
States:  
A, B, and C represent the rooms. Each room can be either clean (0) or dirty (1).  
The agent will perform actions in these rooms to reach a goal state.  
  
Goal State:  
The goal state is when all rooms (A, B, and C) are clean, i.e., {"A": 0, "B": 0, "C": 0}  
  
Actions:  
0: CLEAN (clean the current room if it's dirty)  
1: DIRTY (move to the next room if the current room is clean)  
  
States Sequence:  
A -> B -> C are rooms connected in this order. The agent can move from one room to another.  
The agent starts in a given room (A/B/C) and moves through the rooms in sequence A -> B -> C.  
  
Path Cost:  
The path cost is +1 for every action performed (either cleaning or moving from one room to another).  
  
Inputs:  
1. LOCATION: The initial location of the vacuum cleaner (A, B, or C).  
2. Status of the current room: (0 for clean, 1 for dirty).  
3. Status of the other rooms: Status of rooms A, B, and C (0 for clean, 1 for dirty).  
  
Output:  
The program returns a sequence of actions that leads to the goal state, where all rooms are clean.  
It also outputs the path cost, which is the total number of actions taken by the agent.  
  
Agent's Behavior:  
- If the current room is dirty, the agent will clean it (Action: 0).  
- If the current room is clean, the agent will move to the next room in the sequence (Action: 1).  
- The agent senses the status of other rooms before performing any action, so it knows if it needs to move or clean.  
  
Goal Test:  
The agent keeps performing actions until all rooms (A, B, and C) are clean, i.e., the state becomes {"A": 0, "B": 0, "C": 0}.  
  
The agent's task is to clean the rooms and then report the sequence of actions it performed to reach the goal state, along with the total path cost.  
"""  
  
# Reflex Agent Program for Vacuum Cleaner World Problem  
  
# Goal State: When all rooms are clean (i.e., {"A": 0, "B": 0, "C": 0})*goal\_state = {"A": 0, "B": 0, "C": 0}  
  
  
*# Function to check if the goal state is achieved  
def* goal\_test(state):  
 *return* state == goal\_state  
  
  
*# Function to clean a room (sets room status to clean, i.e., 0)  
def* clean(room, state):  
 print(f"Cleaning room {room}...")  
 state[room] = 0  
 *return* state  
  
  
*# Function to move to the next room  
def* move\_to\_next\_room(current\_room):  
 *if* current\_room == "A":  
 *return* "B"  
 *elif* current\_room == "B":  
 *return* "C"  
 *else*:  
 *return None # No more rooms after C  
  
  
# Function to handle the agent's behavior based on its location and room status  
def* agent\_perform\_action(current\_room, state, actions, path\_cost):  
 *if* state[current\_room] == 1: *# If current room is dirty* state = clean(current\_room, state)  
 actions.append(f"Clean {current\_room}")  
 path\_cost += 1 *# Cleaning action adds 1 to path cost  
 else*:  
 next\_room = move\_to\_next\_room(current\_room)  
 *if* next\_room:  
 actions.append(f"Move to {next\_room}")  
 path\_cost += 1 *# Moving to the next room adds 1 to path cost* current\_room = next\_room  
 *return* current\_room, state, actions, path\_cost  
  
  
*# Function to get the status of rooms from the user  
def* get\_user\_input():  
 initial\_location = input("Enter initial location (A/B/C): ").strip().upper()  
 status\_A = int(input("Enter status of room A (0 for clean, 1 for dirty): "))  
 status\_B = int(input("Enter status of room B (0 for clean, 1 for dirty): "))  
 status\_C = int(input("Enter status of room C (0 for clean, 1 for dirty): "))  
 *return* initial\_location, [status\_A, status\_B, status\_C]  
  
  
*# Main function to execute the vacuum cleaner agent's task  
def* vacuum\_cleaner\_agent(initial\_location, status):  
 *# Initialize state based on input* state = {"A": status[0], "B": status[1], "C": status[2]}  
  
 current\_location = initial\_location  
 actions = []  
 path\_cost = 0  
  
 *# Continue cleaning until the goal state is reached  
 while not* goal\_test(state):  
 current\_location, state, actions, path\_cost = agent\_perform\_action(  
 current\_location, state, actions, path\_cost  
 )  
  
 *# Return the sequence of actions and total path cost  
 return* actions, path\_cost  
  
  
*# Function to display the results of the agent's actions  
def* display\_results(actions, path\_cost):  
 print("\nSequence of Actions:")  
 *for* action *in* actions:  
 print(action)  
  
 print(f"\nTotal Path Cost: {path\_cost}")  
  
  
*# Main driver function  
if* \_\_name\_\_ == "\_\_main\_\_":  
 initial\_location, status = get\_user\_input()  
  
 *# Run the vacuum cleaner agent* actions, path\_cost = vacuum\_cleaner\_agent(initial\_location, status)  
  
 *# Display the results* display\_results(actions, path\_cost)

# 