**Lab Exercise 2**

**Implementation of Agent Problem**

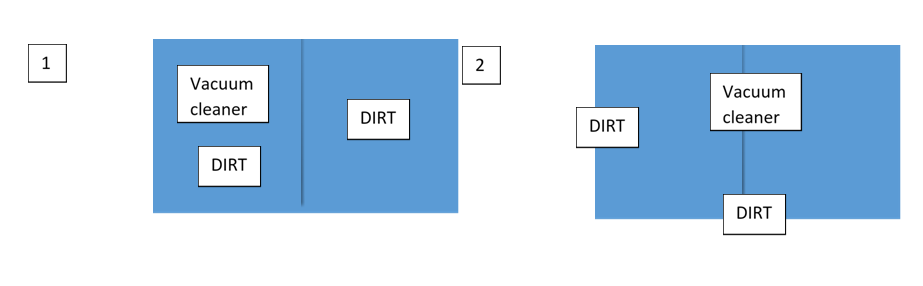
Anjali Agrawal

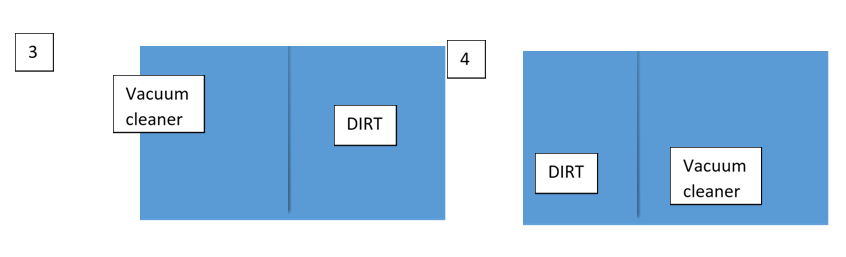
RA1811030010096

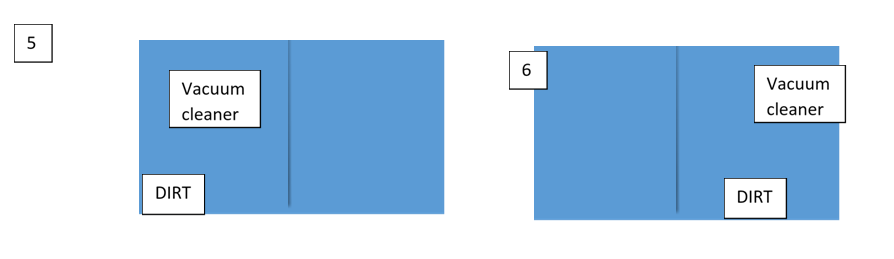
**Title:** Intelligent Agents - Vacuum Cleaner Problem

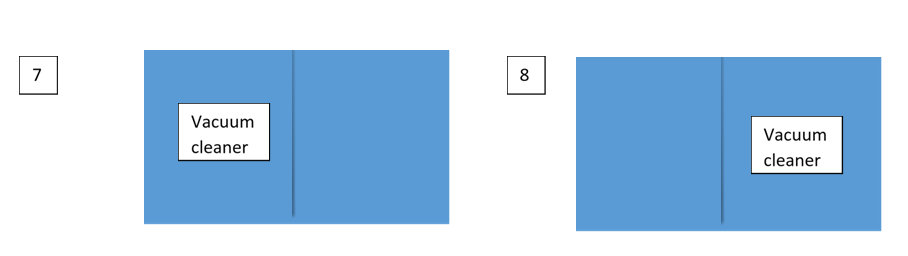
**Problem Description:** Vacuum cleaner problem is a well-known search problem for an [agent](https://www.includehelp.com/ml-ai/artificial-intelligence-based-agent.aspx) which works on [Artificial Intelligence](https://www.includehelp.com/ml-ai/introduction-to-artificial-intelligence.aspx). In this problem, our vacuum cleaner is our [agent](https://www.includehelp.com/ml-ai/artificial-intelligence-based-agent.aspx). It is a goal based agent, and the goal of this agent, which is the vacuum cleaner, is to clean up the whole area. So, in the classical vacuum cleaner problem, we have two rooms and one vacuum cleaner. There is dirt in both the rooms and it is to be cleaned. The vacuum cleaner is present in any one of these rooms. So, we have to reach a state in which both the rooms are clean and are dust free.

**Solution:**There are eight possible states possible in our vacuum cleaner problem. These can be well illustrated with the help of the following diagrams:

****

****

****

****

Here, states 1 and 2 are our initial states and state 7 and state 8 are our final states (goal states). This means that, initially, both the rooms are full of dirt and the vacuum cleaner can reside in any room. And to reach the final goal state, both the rooms should be clean and the vacuum cleaner again can reside in any of the two rooms.

The vacuum cleaner can perform the following functions: move left, move right, move forward, move backward and to suck dust. But as there are only two rooms in our problem, the vacuum cleaner performs only the following functions here: move left, move right and suck.

Here the performance of our agent (vacuum cleaner) depends upon many factors such as time taken in cleaning, the path followed in cleaning, the number of moves the agent takes in total, etc. But we consider two main factors for estimating the performance of the agent. They are:

1. Search Cost: How long the agent takes to come up with the solution.
2. Path cost: How expensive each action in the solution is.

By considering the above factors, the agent can also be classified as a utility-based agent.

**Python Code:**

#INSTRUCTIONS

#Enter LOCATION A/B in capital letters

#Enter Status O/1 accordingly where 0 means CLEAN and 1 means DIRTY

def vacuum\_world():

# initializing goal\_state

# 0 indicates Clean and 1 indicates Dirty

goal\_state = {'A': '0', 'B': '0'}

cost = 0

location\_input = input("Enter Location of Vacuum") #user\_input of location vacuum is placed

status\_input = input("Enter status of " + location\_input) #user\_input if location is dirty or clean

status\_input\_complement = input("Enter status of other room")

print("Initial Location Condition" + str(goal\_state))

if location\_input == 'A':

# Location A is Dirty.

print("Vacuum is placed in Location A")

if status\_input == '1':

print("Location A is Dirty.")

# suck the dirt and mark it as clean

goal\_state['A'] = '0'

cost += 1 #cost for suck

print("Cost for CLEANING A " + str(cost))

print("Location A has been Cleaned.")

if status\_input\_complement == '1':

# if B is Dirty

print("Location B is Dirty.")

print("Moving right to the Location B. ")

cost += 1 #cost for moving right

print("COST for moving RIGHT" + str(cost))

# suck the dirt and mark it as clean

goal\_state['B'] = '0'

cost += 1 #cost for suck

print("COST for SUCK " + str(cost))

print("Location B has been Cleaned. ")

else:

print("No action" + str(cost))

# suck and mark clean

print("Location B is already clean.")

if status\_input == '0':

print("Location A is already clean ")

if status\_input\_complement == '1':# if B is Dirty

print("Location B is Dirty.")

print("Moving RIGHT to the Location B. ")

cost += 1 #cost for moving right

print("COST for moving RIGHT " + str(cost))

# suck the dirt and mark it as clean

goal\_state['B'] = '0'

cost += 1 #cost for suck

print("Cost for SUCK" + str(cost))

print("Location B has been Cleaned. ")

else:

print("No action " + str(cost))

print(cost)

# suck and mark clean

print("Location B is already clean.")

else:

print("Vacuum is placed in location B")

# Location B is Dirty.

if status\_input == '1':

print("Location B is Dirty.")

# suck the dirt and mark it as clean

goal\_state['B'] = '0'

cost += 1 # cost for suck

print("COST for CLEANING " + str(cost))

print("Location B has been Cleaned.")

if status\_input\_complement == '1':

# if A is Dirty

print("Location A is Dirty.")

print("Moving LEFT to the Location A. ")

cost += 1 # cost for moving right

print("COST for moving LEFT" + str(cost))

# suck the dirt and mark it as clean

goal\_state['A'] = '0'

cost += 1 # cost for suck

print("COST for SUCK " + str(cost))

print("Location A has been Cleaned.")

else:

print(cost)

# suck and mark clean

print("Location B is already clean.")

if status\_input\_complement == '1': # if A is Dirty

print("Location A is Dirty.")

print("Moving LEFT to the Location A. ")

cost += 1 # cost for moving right

print("COST for moving LEFT " + str(cost))

# suck the dirt and mark it as clean

goal\_state['A'] = '0'

cost += 1 # cost for suck

print("Cost for SUCK " + str(cost))

print("Location A has been Cleaned. ")

else:

print("No action " + str(cost))

# suck and mark clean

print("Location A is already clean.")

# done cleaning

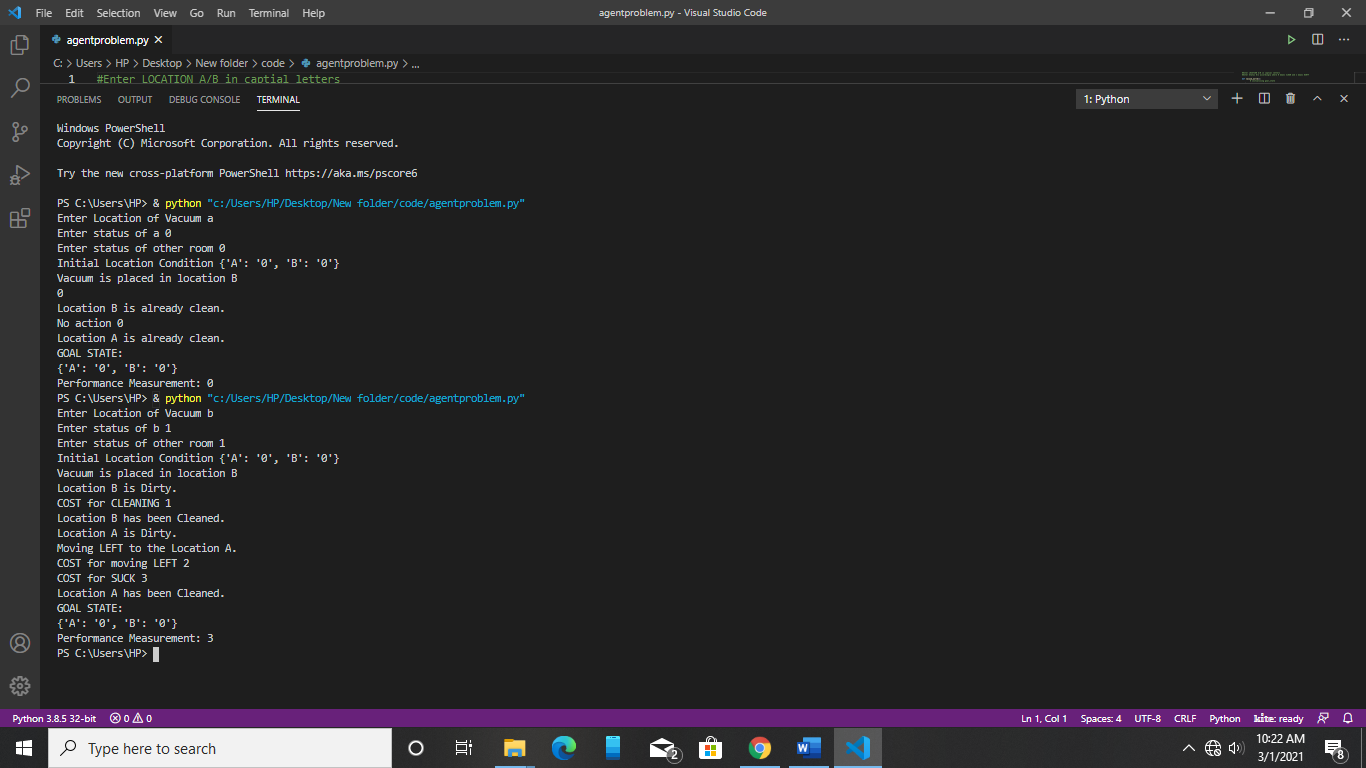
print("GOAL STATE: ")

print(goal\_state)

print("Performance Measurement: " + str(cost))

vacuum\_world()

**Input and output:**



**Result:**

The code was run successfully and the problem was solved using AI