**PMAS Arid Agriculture University**

**University Institute of Information Technology**

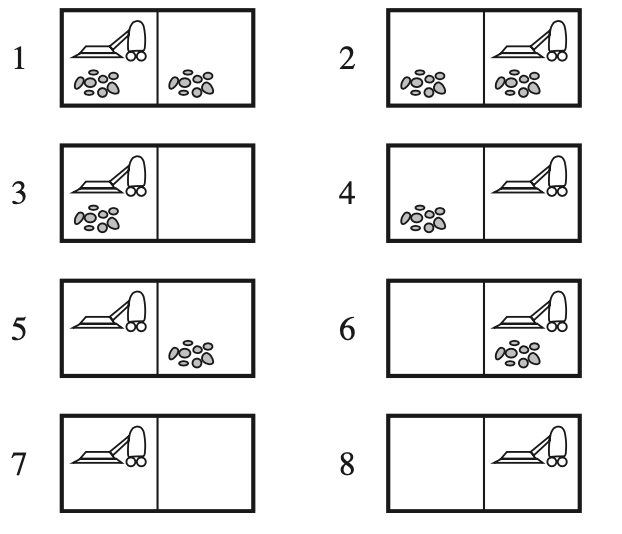
**CS-632 Artificial Intelligence**

**Handout 01**

# Vacuum World Problem

**Vacuum cleaner problem** is a well-known search problem for an [agent](https://www.includehelp.com/ml-ai/artificial-intelligence-based-agent.aspx" \t "_blank) which works on [Artificial Intelligence](https://www.includehelp.com/ml-ai/introduction-to-artificial-intelligence.aspx" \t "_blank). In this problem, our vacuum cleaner is our [agent](https://www.includehelp.com/ml-ai/artificial-intelligence-based-agent.aspx" \t "_blank). 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.

So, there are eight possible states possible in our **vacuum cleaner problem**. These can be well illustrated with the help of the following diagram:



First, suppose that the agent's sensors give it enough information to tell exactly which state it is in (i.e., the world is completely accessible) and suppose it knows exactly what each of its actions does.

Then it can calculate exactly what state it will be in after any sequence of actions. For example, if the initial state is 5, then it can calculate the result of the actions sequence {right, suck}.

This simplest case is called a **single-state problem.**

Now suppose the agent knows all of the effects of its actions, but world access is limited. For example, in the extreme case, the agent has no sensors so it knows only that its initial state is one of the set {1,2,3,4,5,6,7,8}. In this simple world, the agent can succeed even though it has no sensors. It knows that the action {right} will cause it to be in one of the states {2,4,6,8}. In fact, it can discover that the sequence {right, suck, left, suck} is guaranteed to reach a goal state no matter what the initial state is.

In this case, when the world is not fully accessible, the agent must reason about sets of states that it might get to, rather than single states. This is called the **multiple-state problem.**

The case of ignorance about the effects of actions can be treated similarly.

**Example** Suppose the *suck* action sometimes deposits dirt when there is none. Then if the agent is in state 4, sucking will place it in one of {2,4}. There is still a sequence of actions that is guaranteed to reach the goal state.

However, sometimes ignorance prevents the agent from finding a guaranteed solution sequence. Suppose that the agent has the nondeterministic suck action as above and that it has a position sensor and a local dirt sensor. Suppose the agent is in one of the states {1,3}. The agent might formulate the action sequence {suck, right, suck}. The first action would change the state to one of {5,7}; moving right would then change the state to {6,8}. If the agent is, in fact, in state 6, the plan will succeed, otherwise it will fail. It turns out there is no fixed action sequence that guarantees a solution to this problem.

The agent can solve the problem if it can perform sensing actions during execution. For example, starting from one of {1,3}: first suck dirt, then move right, then suck *only if there is dirt there*. In this case the agent must calculate a whole tree of actions rather than a single sequence, i.e., plans now have conditionals in them that are based on the results of sensing actions.

***Code:***

*#INSTRUCTIONS*

*#Enter LOCATION A/B in captial 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()*

***Output:***

***Enter Location of Vacuum***

*A*

***Enter status of A***

*1*

***Enter status of other room***

*1*

*Initial Location Condition{'A': '0', 'B': '0'}*

*Vacuum is placed in Location A*

*Location A is Dirty.*

*Cost for CLEANING A 1*

*Location A has been Cleaned.*

*Location B is Dirty.*

*Moving right to the Location B.*

*COST for moving RIGHT2*

*COST for SUCK 3*

*Location B has been Cleaned.*

*GOAL STATE:*

*{'A': '0', 'B': '0'}*

*Performance Measurement: 3*