11/10/24.

PAGE NO :
DATE :

Vacuum Cleaner Agent

Algorithm

1. Initialize goal state as {'A': '0', 'B': '0'}
   0 = Clean, 1 = Dirty
2. Set cost = 0
3. Get location input
4. Get Status_input complement.
6. If location_input == 'A'
   a. If Status input == '1':
      (i) Clean room A
      (ii) Update goal_state['A'] = '0'
      (iii) Increment cost by 1.
   b. If Status_input_complement == '1':
      (i) Move from B
      (ii) Increment cost by 1
      (iii) Clean room B
      (iv) Update goal state ['B'] = '0'.
      (v) Increment cost by 1
7. Else if location input == 'B'
   (a) If Status_input == '1'
      (i) Clean room B
      (ii) Update goal_state['B'] = '0'
      (iii) increment cost by 1

(b) If Status_input_complement == 'B'
(i) Move to room A
(ii) Increment cost by 1
(iii) Clean room A
(iv) Update goal_State ['A'] = '0'
(v) Increment cost by 1

8. Print the final goal State
9. print the total performance cost

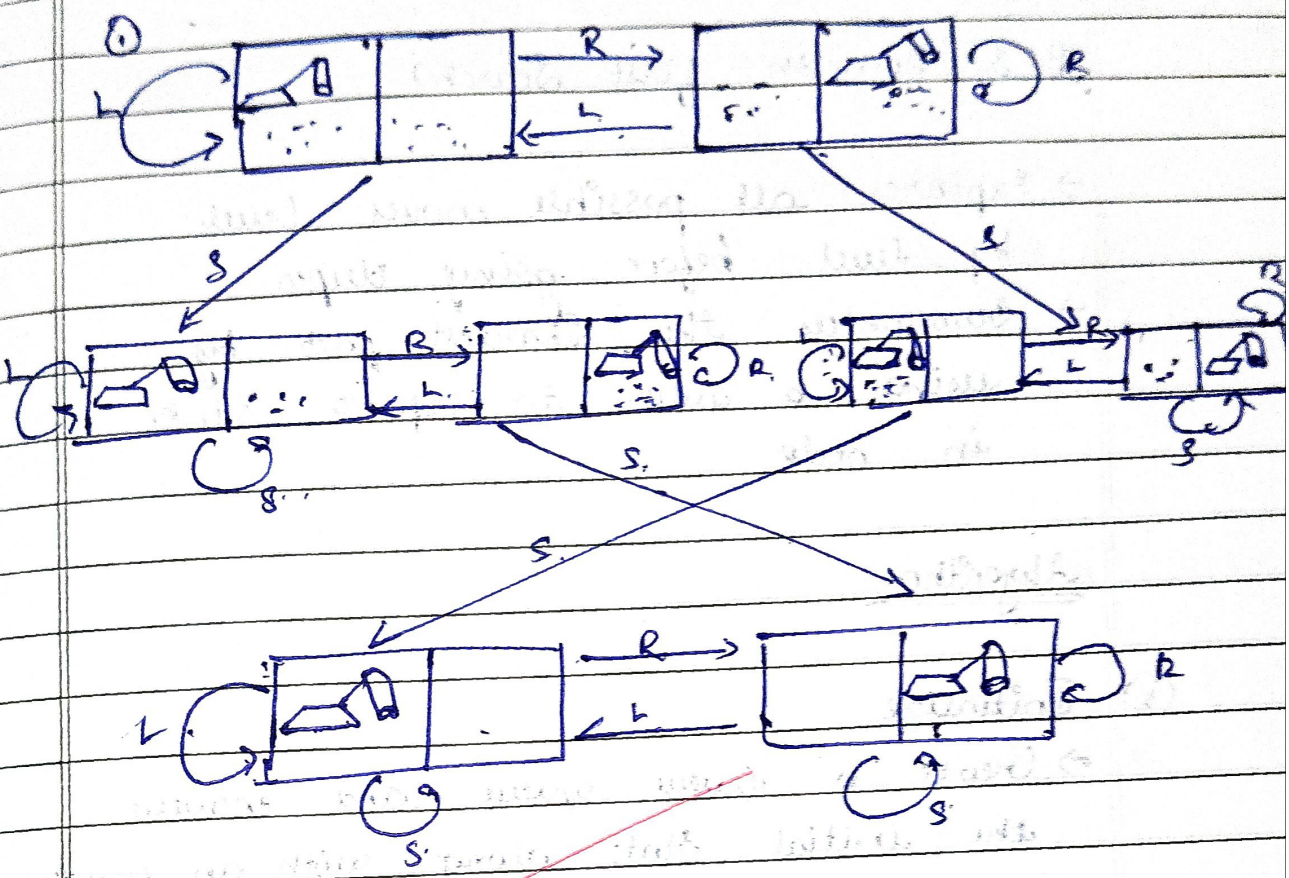
OUTPUT

Enter location of vacuum (A or B) : A
Enter Status of Room A : 0
Enter Status of the other room : 1

Initial room condition : {'A' : '0', 'B' : '1'}
Vacuum is placed in location B
Location A is already clean
Location B is Dirty, Moving to B:
Cost for moving to B --
Cleaning B --
cost for cleaning B : 2
Final room Condition : {'A' : '0', 'B' : '0'}
performance measurement (Total cost) : 2

PAGE NO :
DATE :

State Space Diagram



Evaluation steps

1. Completeness :-
   The implementation handles all scenerios for cleaning both rooms based on their statuses

2. Time complexity
   O(1) as operations are constant regardless of input size

3. Space complexity
   O(1) for storing the state of the rooms and cost, fixed and independent of input size.