SL.NO:CAA170967

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING

(An Autonomous Institution, Affiliated to Anna University, Chennai)
Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

THEORY EXAMINATIONS

Register Number	205001085		
Name of the Student	V. Sabarivasan		
Degree and Branch	BE CSE	Semester	V
Subject code and Name	UCS1504 Artificial Intelligence		
Assessment Test No.	I	Date	

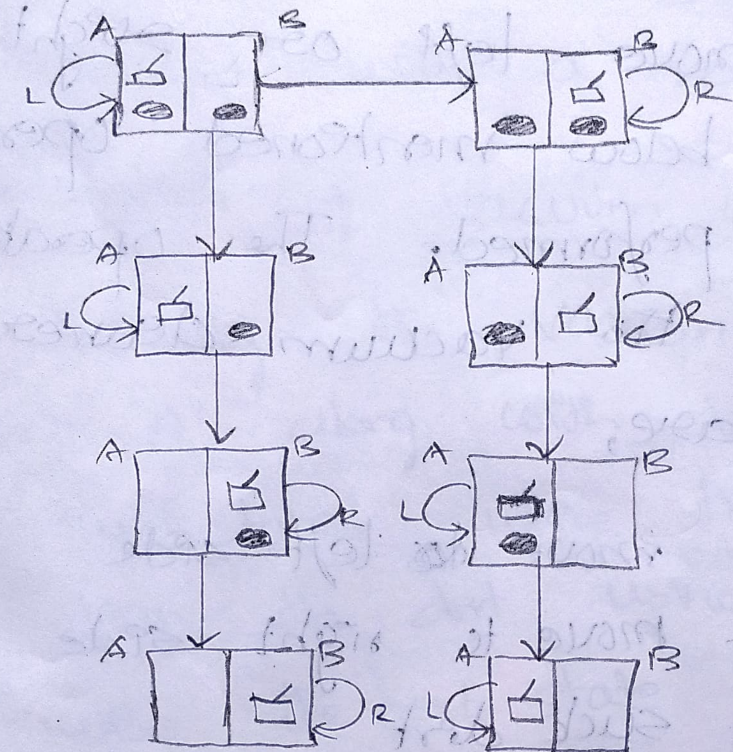
Details of Marks Obtained									
Part A		Part B				Part C			
Question No.	Marks	Question No.	(a) Marks	(b) Marks	Total Marks	Question No.	(a) Marks	(b) Marks	Total Marks
1	½	7	1			10	9		
2	2					11			
3	0	8	2			12	9		
4	0								
5	2	9	6			13			
6	2								
Total (A)		Total (B)				Total (C)			33½

Grand Total (A+B+C)				Marks (in words)			
Signature of Faculty							

SSN

PART-C

(10) State space representation.



Initial state:

The vacuum cleaner is in left room 'A' where there is presence of dirt. This is the initial state

Successor function:

The vacuum cleaner can either move left or right, so that the below mentioned operators can be performed. The operations in which the vacuum cleaner can perform are,

→ LEFT - move ~~to~~ left side

→ RIGHT - move to right side

→ SUCK - suck dirt

→ NOOPERATION - if there is no dirt, no operations.

Goal test:

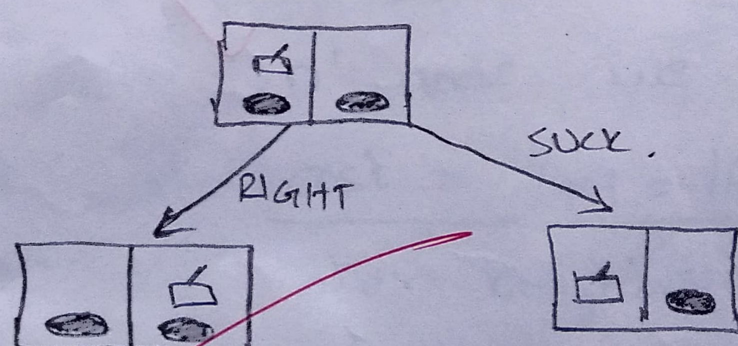
The goal of the vacuum world is to ensure that both rooms 'A' and 'B' are clean.

Path cost:

Each operation takes a cost of 1, ~~the~~ which means each step is increased with a cost of 1.

Explanation of vacuum world problem:

→ Initially the vacuum cleaner is in room 'A' along with dirt. So it leads to two states where the 1st state is to clean the dirt using 'SUCK' operation otherwise ~~the~~ the next state will be moving to room 'B' by 'RIGHT' operation.

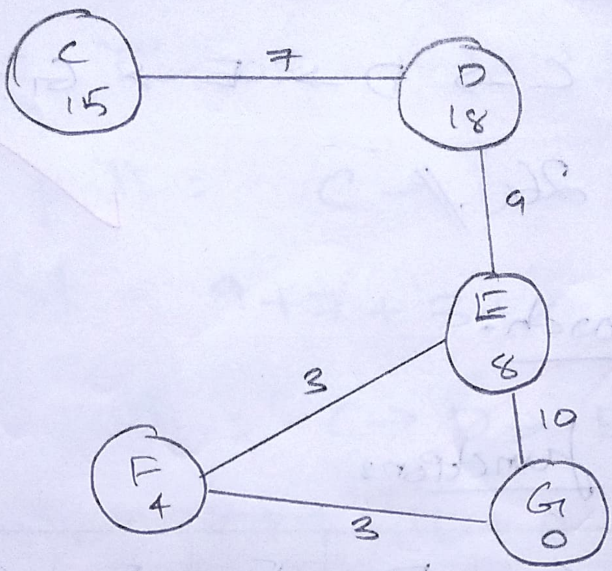


(State A) (State B)

→ For state A, the vacuum cleaner can perform 'SUCK' operation and in state B, the vacuum cleaner can perform 'RIGHT' operation to move to room 'B'.

⇒ If the room is already cleaned or if there is no dirt, then 'NOOPERATION' is performed where the vacuum cleaner remains still due to no dirt presence.

(12)



?!!

h(n)

(i) <u>Greedy Search</u>

⇒ <u>Inital</u> state is at 'C'. First we can reach node 'D' from 'C' at <u>cost = 7</u>. <u>(C → D)</u>.

⇒ Then from 'D' node, we can move to node 'E' at <u>cost = 7+9 = 16</u> since there are no other neighbours of 'D'.
<u>(C → D → E)</u>

⇒ From node 'D', it has the choice either to move to 'G' or to 'F'. But since 'G' is our goal state, the next step will be moving to node 'G' from E at cost = 16 + 10 = 26

∴ Path: C → D → E → G //

Cost: 26 //

(ii) A* search:

Heuristic function:

n	C	D	E	F	G
h(n)	15	18	8	4	0

⇒ Starting from node 'C', it moves to node 'D' since 'D' is the only neighbour of node 'C'. (C → D)

* Cost = 7 + 18 = 25 //

⇒ From node 'D', it moves to node 'E' since 'E' is the only neighbour of node 'D'. (C → D → E)

* Cost = 7 + 9 + 8 = 24 //

⇒ From node E, it can either move to node 'F' or node 'G'.

(i) For path: C → D → E → F

Cost = 9 + 7 + 3 + 4 = 23 //

(ii) For path: C → D → E → G

Cost = 7 + 9 + 10 + 0 = 26 //

∴ Path: C → D → E → F is chosen.

⇒ From node 'F', it moves to node 'G' which is our goal state.

∴ Path: C → D → E → F → G. //

Cost: 7 + 9 + 3 + 3 + 0 = 22 //

Result:

A* search gives optimal solution when compared to Greedy search.

PART-B

(9) **Depth Limited Search:**

⇒ DLS is similar to Depth First search Algorithm where in this algorithm, an additional factor called 'limit' is added.

⇒ DLS algorithm solves the infinite loop problem caused in DFS search.

⇒ DLS algorithm is defined as it behaves like Depth first search Algorithm to a certain limited level in a tree structure.

→ It has two failure factors. They are,

(i) Cutoff failure value.

(ii) Absence failure value.

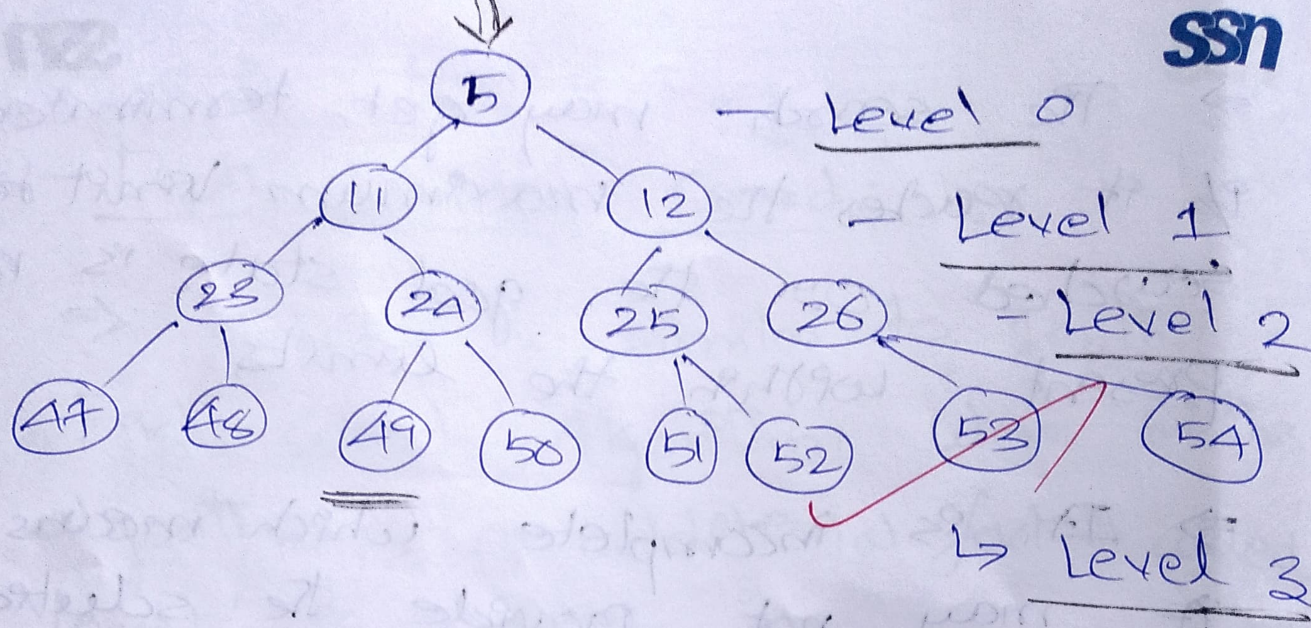
⇒ The search may get terminated if it reaches the maximum limit to be searched or the goal state is not present within the limits.

⇒ It is incomplete which means it may not provide the solution which we needed.

⇒ Hence, it is not optimal.

⇒ Time Complexity: $O(b^l)$ where 'b' is no. of nodes and 'l' is the maximum depth limit.

⇒ Space complexity: $O(bl)$ which is linear space complexity.



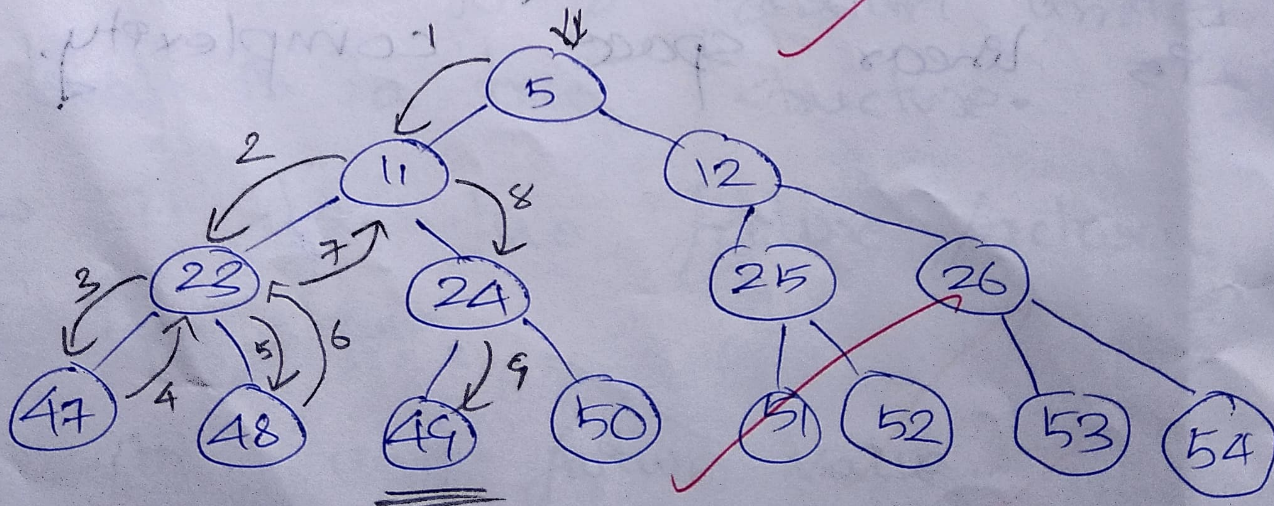
Order of traversal :-

i) $5 \xrightarrow{1} 11 \xrightarrow{2} 23 \xrightarrow{3} 47$

ii) $47 \xrightarrow{4} 23 \xrightarrow{5} 48$

iii) $48 \xrightarrow{6} 23 \xrightarrow{7} 11 \xrightarrow{8} 24$

iv). $24 \xrightarrow{9}$ 49 (good state)



(8) Water jug problem

(i) Initial State:-

There are 3 jugs of capacity 8, 5 and 3 galloons and are filled initially with 8, 0 and 0 gallons respectively.

(ii) Goal state:

Goal state is to get exactly 4 gallons of water in each of the 8-gallon and 5 gallon jugs.

(iii) Actions:

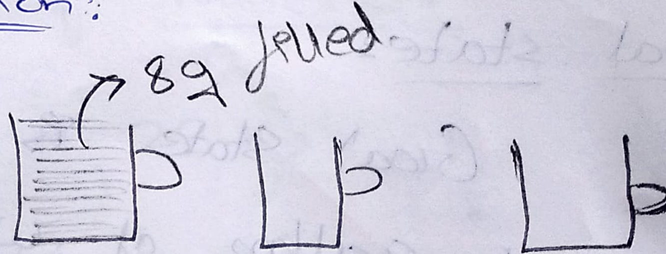
The action which we can take place is pour water from a jug to another jug within the specific capacity of each jug. If a jug is already full, ~~the~~ water cannot be poured into the jug anymore.

(iv) Path cost:

The cost is calculated by number of times a jug is used to pour water on another jug.

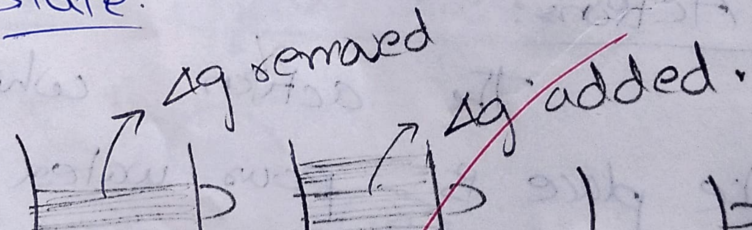
(i.e) cost = 1 for using a jug to pour water from it to another jug.

(v) Solution:



Capacity: 8g 5g 3g

Goal state:



Capacity: 8g 5g 3g.

⇒ The solution is to pour 4 gallons of water from jug with 8 gallons capacity to the jug with 5 gallons capacity. In this way, the final goal state is reached as <u>cost = 1</u>.

<u>PART - A</u>

① ~~a) Partially / Fully observable.~~

~~b) Deterministic / Stochastic~~

~~c)~~

a) Fully observable

b) Deterministic

c) sequential.

~~d)~~ Static

~~e)~~ Discrete.

(2) PEAS :

Performance: Hitting the tennis ball correctly.

Environment: Tennis court.

Actuators: Tennis bat which hits the ball.

Sensors: Eyes which focus on the tennis ball.

---

(3) branching factor } $b = 10$.

Depth $d = 5$.

$\therefore$ Total no. of nodes $= 10 \times 5 = 50$ //.

(5) Algorithm for tree search:

~~function Tree-Search (problem, source):~~

~~function Search (problem, source, goal):~~

~~if source == goal: return true~~

~~x = successor (source)~~

~~while (x != goal):~~

~~x = successor (x)~~

~~return false~~

```
function T-Search (problem, source, goal):
    if (source == goal): return True
    x = successor (source)
    while (x != goal)
        x = successor (x)
        if (x == goal): return True
    return False.
```

6 Drawbacks of hill climbing search:

(i) ~~Local minimum and global main~~

(i) Local maximum / Global maximum problem.

(ii) Flat maximum.

(iii) Ridges

Overcome the drawbacks:

⇒ Using steepingascend hill climbing search algorithm, can solve the above drawbacks.

(4) Requirements of total Turing Test:

i) Problem agent.

ii) Goal state.

iii) Path cost.

iv) Actions.

v) Initial state.

PART-B

(7)

Type of agent:

Utility based reference agent.

Explanation:

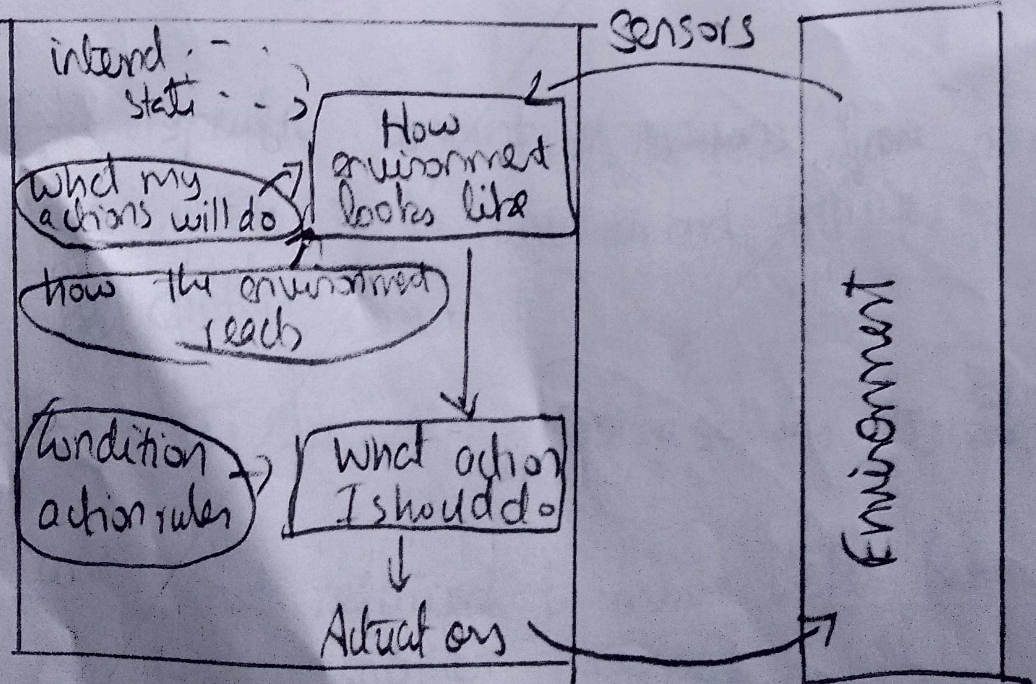
In this type of agent, it keeps track of the current state of the world using an internal state.

7. This type of agent is model based agent :

This agent ~~give~~ takes information from the environment using sensors.

It then takes input on what the actions will do and how the environment ~~(illegible)~~ reacts.

It finally applies the condition action rule to the actuators and performs the action.



Δ Structure of model based agent.

Example : Vacuum cleaner

It takes input on dirtiness of room through sensors

Actions are move right/left, suck

Actuators perform the actions.

Part-A

1.
a) Partially observable
b) Deterministic
c) Sequential
d) Dynamic
e) Discrete

2. "Playing a tennis match",

P Performance Measure : How ~~accurate and~~ fast the ball is hit
highest score

E Environment : Ground, net

A Actuators : ~~Bat~~, legs, hands (bat) ~~..~~

S Sensors : Eyes ~~..~~