Artificial Intelligence

Name: Bhukya Vasanth kumar
Roll: B180441CS
S-7 CSE (A batch)

(1) PEAS description

(a) Internet book-shopping Agent

- Performance Measure: price, quality of book, authors, book review
- Environment: Vendors, shippers, web
- Actuators: follow URL link, display to user, fill the form
- sensors: HTML pages (text, graphics, scripts)

characteristics:

observable/not : NOT
Deterministic / stochastic : PARTLY
Episodic / sequential : STOCHASTIC
static / Dynamic : SEMI
Discrete / continous : DISCRETE
single Agent / multi Agent : MULTI

(b) performing high jump

- performance Measure: safety, altitude
- Environment: wall
- Actuators: jumping apparatus
- sensors: camera, height sensors

characteristics:

observable/not : YES
Deterministic / stochastic : DETERMINISTIC
Episodic / sequential : EPISODIC
static / Dynamic : STATIC
Discrete / continous : DISCRETE
single agent / multiagent : SINGLE

(2) (a) FALSE.

perfect rationality means, the ability to make good decisions, given the sensor information is received.

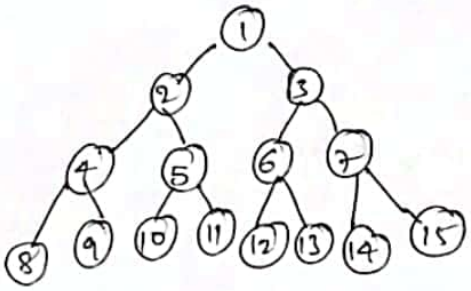
Ex: vacuum cleaner is Rational but doesn't serve state of square adjacent to it.

(b) TRUE.

A pure reflex agent ignores previous percepts, so cannot obtain an optimal state estimate in partially observable environment in the world.

Ex: correspondence chess is played by sending moves. If one player's move → current percept. a reflex agent cannot keep track of board state and would have to respond to say, 'b2', in same way, regardless of position in which it was played.

(3) (a) State space where start state = 1, each state has 2 successors. 2k, 2k+1.



(b) Goal state = 11.

order in which nodes will be visited:

(*) Breadth First search (BFS): 1→2→3→4→5→6→7→8→9→10→11

(*) Depth limited search : 1→2→4→8→9→5→10→11 (limit = 3)

(*) Iterative deepening search: 1; 1→2; 2→3;

1→2→4→5→3→6→7;

1→2→4→8→9→5→10→11

(4) using only four colours, you have to colour a planar map in such a way that no two adjacent regions have same colour.

→ Initial state: No regions coloured

Actions (3rd ed.) / successors (2nd ed): Assign to a colour to an uncoloured region.

Transition Model (3rd ed): Previously uncolored region has assigned colour.

Goal Test: All regions can be colored, and no two adjacents have same color.

cost function: Number of assignments.

(5) state space of problem can be described as set of ordered pairs of integers (x,y).

Where,
x represents quantity of water in 4- Gallon Jug $x=0,1,2,3,4$
y represents quantity of water in 3- Gallon Jug $y=0,1,2,3$.

Start state: (0,0)
Goal state: (2,0)
Production Rules:-

Rule	state	Process
1	$(x,y \mid x<4)$	$(4,y)$ Fill 4-gallon Jug
2	$(x,y \mid y<3)$	$(x,3)$ Fill 3-gallon Jug
3	$(x,y \mid x>0)$	$(0,y)$ Empty 4-gallon Jug
4	$(x,y \mid y>0)$	$(x,0)$ Empty 3-gallon Jug
5	$(x,y \mid x+y>=4 \wedge y>0)$	$(4, y-(4-x))$ Pour water from 3 gallon jug into 4 gallon jug until 4 gallon is full
6	$(x,y \mid x+y<=3 \wedge x>0)$	$(x-(3-y),3)$ Pour water from 4 gallon jug into 3 gallon jug until 3 gallon is full
7	$(x,y \mid x+y<=4 \wedge y>0)$	$(x+y,0)$ Pour water from 3 to 4 gallon jug
8	$(x,y \mid x+y<=3 \wedge x>0)$	$(0,x+y)$ Pour water from 4 to 3 gallon jug
9	$(0,2)$	$(2,0)$ Pour 2 gallon water from 3 to 4 gallon jug

Initialization :

Start state : (0,0)

Apply Rule 2

(x,y) | y<3) → (x,3) {Fill 3 gallon jug}

state = (x,3)

Iteration 1:

current state : (x,3)

Apply Rule 7, (x,y) | x+y <= 4 ^ y > 0) → (x+y,0)

{ Pour all water from 3 to 4 gallon jug }

state = (3,0)

Iteration 2:

current state = (3,0)

Apply Rule 2, (x,y) | y < 3) → (3,3) {Fill 3-gallon jug}

state = (3,3)

Iteration 3:

current state = (3,3)

Apply, Rule 5, (x,y) | x+y >= 4 ^ y > 0) → (4, y-(4-x))

{ Pour water from 3 to 4 gallon jug until 4 gallon jug is full. }

state = (4,2)

Iteration 4:

current state = (4,2)

Apply Rule 3, (x,y) | x > 0) → (0,y)

{ Empty 4-gallon jug }

state = (0,2)

Iteration 5:

current state = (0,2)

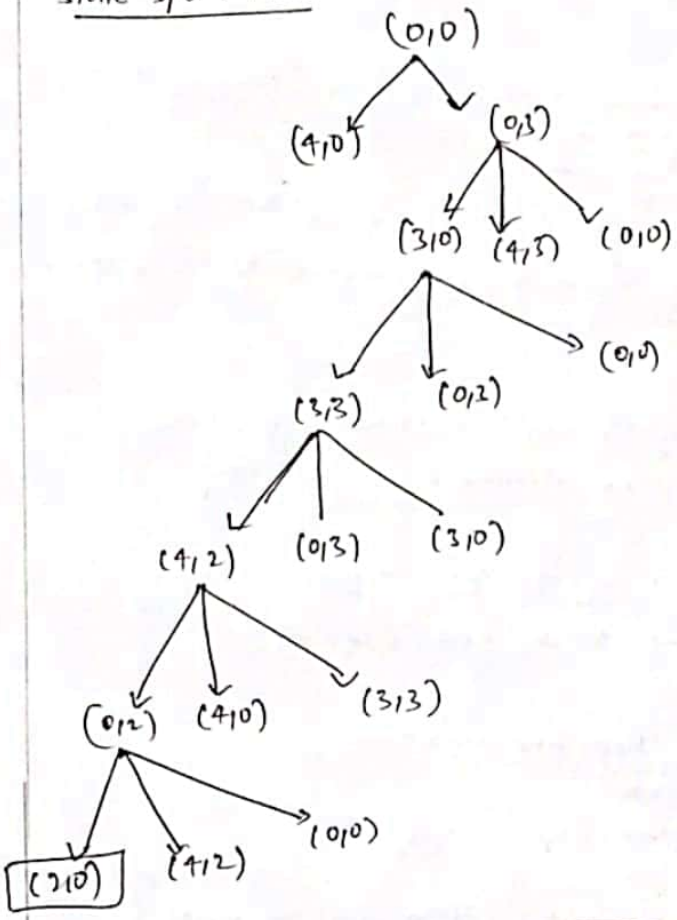
Apply Rule 9, (0,2) → (2,0)

{ Pour 2 gallon from 3 to 4 gallon jug }

state = (2,0)

GOAL ACHIEVED.

State space Tree:



(6) Breadth first search is complete whenever branching factor is finite, even if step costs are allowed.
(zero)

→ TRUE.
Because, if there exists a goal it occurs at finite depth d and will be found in $O(b^d)$ steps.

→ What matters in DFS is depth of goal, not cost.

(7) let there be a domain, where every state has single successor and there is single goal at depth, n.
DFS will find goal in n steps,
Iterative deepening search will take $1+2+\dots n = \frac{n(n+1)}{2}$
$= O(n^2)$ steps.

(8) Best first search with evaluation function as

$f(n) = (2-w)\,g(n) + w\,h(n)$.

(a) If $w > 1$, $wh(n)$ will overestimate distance of goal, making it inadmissible, incomplete. $\therefore$ $\boxed{w \leq 1}$

(b) To determine values of w to be optimal, $f(n) = (2-w)\left(g(n) + \frac{2}{2-w}h(n)\right)$.

$\therefore$ $\boxed{\text{for } 0 \leq w < 1,}$ Algorithm is guaranteed to be optimal.

(c) $w = 0$, gives $f(n) = 2g(n)$.
This behaves exactly like uniform-cost search - the factor of 2 makes no difference in ordering of nodes

$w = 1$, gives A* search $\because$ $f(n) = g(n) + f(n)$.

$w = 2$, gives $f(n) = 2h(n)$ → Greedy Best first search

$\therefore$
- $w = 0$ → uniform best first search
- $w = 1$ → A* search
- $w = 2$ → Greedy Best first search

(9) Farmer, Cabbage, Goat, Wolf (should go from west to east).
Assume ↑

State description:
(<side for farmer>, <side for wolf>, <side for goat>, <side for cabbage>)

Initial state:
(w, w, w, w) — All participants on one side of river let it be w, west.

Final (Goal) state:
(e, e, e, e) — All participants end on east bank of river.

loss (dead) state:
- (e, w, w, e) — Wolf eates goat
- (w, e, e, w) — Wolf eats goat
- (e, e, w, w) — Goat eats cabbage
- (w, w, e, e) — Goat eats cabbage

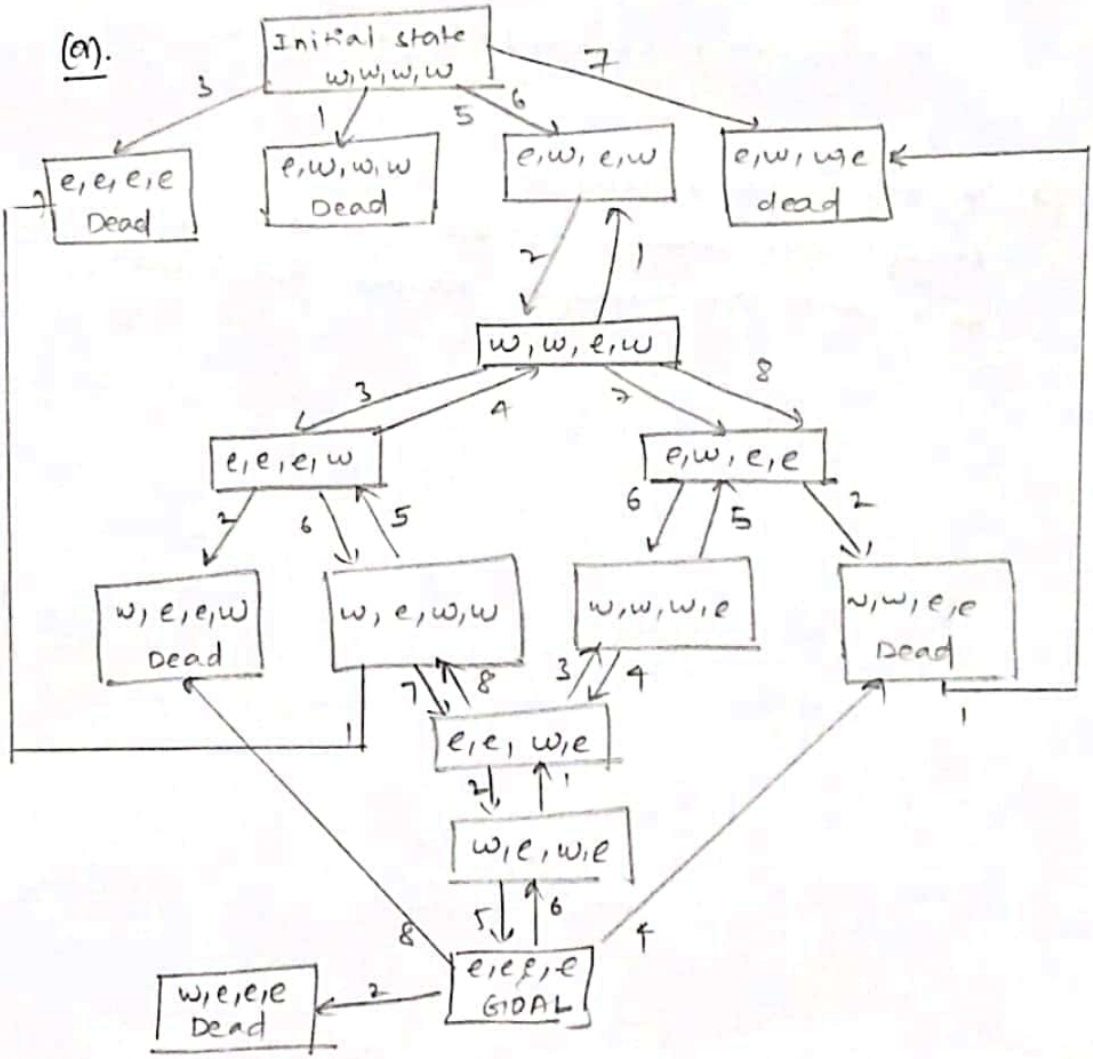
(e, w, w, w) – Goat eats cabbage, wolf eats goat
(w, e, e, e) – Goat eats cabbage, wolf eats goat

(b)

Transitions cost: unit cost for each step.

1. (w, locwolf, locgoat, loccabbage) => (e, locwolf, locgoat, loccabbage)
2. (e, locwolf, locgoat, loccabbage) => (w, locwolf, locgoat, loccabbage)
3. (w, w, locgoat, loccabbage) => (e, e, locgoat, loccabbage)
4. (e, e, locgoat, loccabbage) => (w, w, locgoat, loccabbage)
5. (w, locwolf, w, loccabbage) => (e, locwolf, e, loccabbage)
6. (e, locwolf, e, loccabbage) => (w, locwolf, w, loccabbage)
7. (w, locwolf, locgoat, w) => (e, locwolf, locgoat, e)
8. (e, locwolf, locgoat, e) => (w, locwolf, locgoat, w).

(a).



(c) Heuristic search is a search that incorporates heuristic to guide direction pursued by search.

Soln.

Consider a problem in which only 2 operations can be applied to each state. Assume each Transition leads to unique state.

Starting from initial state, after 1 move, There are 2 possible states, after 2 moves, 4 more states.

∴ The nth move adds 2^n states.

∴ 10th move adds 210 states / 1024 states.

After only 20 moves, there are 2m million states.

This approached which is A* solution can be used.

(d) Yes, this solution is better than 9.(b).

Because, main advantage of heuristic approach is that offers quick solution that is easy to understand and implement.