UCS1504 – Answer key
CAT 1
Part – A

4. NLP, Knowledge representation, automated reasoning, ML, Vision, and Motor control

5. Tree Search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

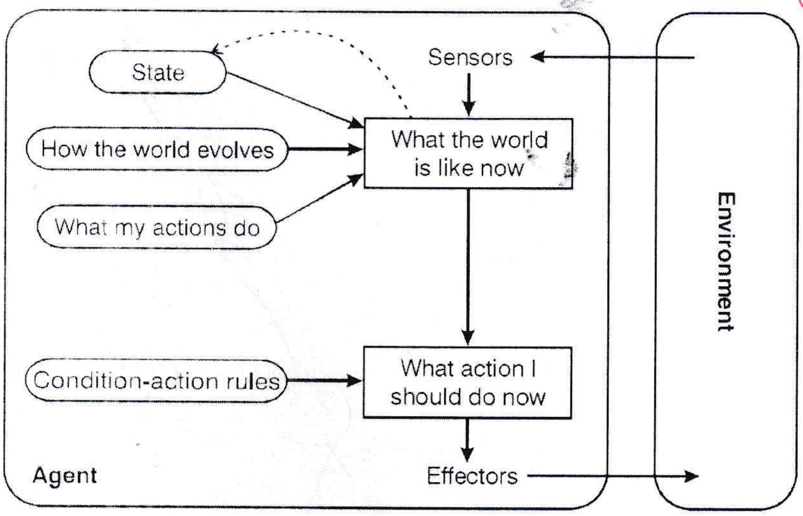
6. Drawbacks of Hill Climbing
- Local maximum
- Plateau
- Ridge

① partially, stochastic, episodic, dynamic, continuous.

Part –B

7. An intelligent agent is a combination of *Agent Program* and *Architecture*
Intelligent Agent = Agent program + Architecture
Agent program is a function that implements the agent mapping from percepts to actions.
Architecture is a computing device used to run the agent program
Model based reflex agents
An agent which combines the current percept with the old internal state to generate updated description of the current state. At what occurrence this method will be useful?
When the sensors do not provide access to the complete state of the world
In some problems, it is necessary to consider the previous state description.
The current percept is combined with the old internal state and it derives a new current state is updated in the state description also. This updation requires two kinds of knowledge in the agent program.
(i) How the world evolves independently of the agent?
(ii) How the agent's own action affect the world?

2+2+2 → Exam
structure
Alg



3. $N(IDS) = (d)b + (d-1)b^2 + \ldots + (1)b^d$

$N(IDS) = 50 + 400 + 3000 + 20000 + 10000 = 1234$

```
function REFLEX-AGENT-WITH-STATE(percept) return action

    static : state, a description of the current world state,
                 rules, a set of condition – action rules

    state ← UPDATE-STATE(state, percept)

    rule ← RULE-MATCH(state, rules)

    action ← RULE-ACTION[rule]

    state ← UPDATE-STATE(state, action)

    return action
```

8. State, initial state, goal state, goal test, path cost, search cost, total cost, successor function or operators

marks

2

State: (i, j, k) = (8, 5, 3)
Initial state: (8, 0, 0)
Goal state: (4, 4, 0)
Operators:

- Partial transformation of water from one jug to another
- Pour complete water from one jug to another

2

Rules:

- Transferring from 8-liter jug to remaining two jugs and vice versa
- Pouring required water to fill the other jug completely
- Emptying the jug: pouring water to the ground or transferring to another jug completely

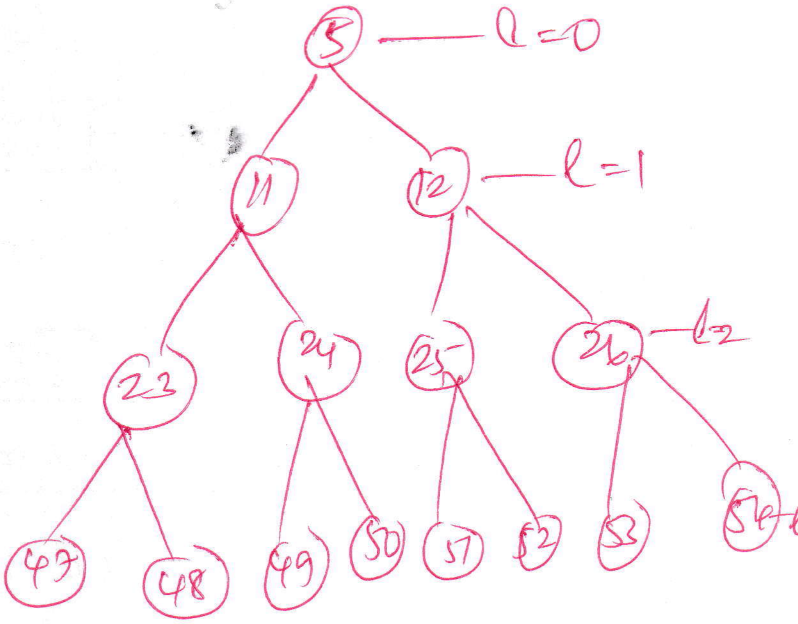
Solution:

2

8 gallons	5 gallons	3 gallons
8	0	0
3	5	0
3	2	3
6	2	0
6	0	2
1	5	2
1	4	3
4	4	0

Ans:

3 marks



Ans : 5 11 23 47 48 24 49

9. Depth Limited Search

function DEPTH-LIMITED-SEARCH(*problem*, *limit*) **returns** a solution, or failure/cutoff
  return RECURSIVE-DLS(MAKE-NODE(INITIAL-STATE[*problem*]), *problem*, *limit*)

(3) marks

function RECURSIVE-DLS(*node*, *problem*, *limit*) **returns** a solution, or failure/cutoff
  cutoff_occurred? ← false
  if GOAL-TEST[*problem*](STATE[*node*]) **then return** SOLUTION(*node*)
  else if DEPTH[*node*] = *limit* **then return** *cutoff*
  else for each *successor* **in** EXPAND(*node*, *problem*) **do**
    result ← RECURSIVE-DLS(*successor*, *problem*, *limit*)
    if *result* = *cutoff* **then** *cutoff_occurred?* ← true
    else if *result* ≠ *failure* **then return** *result*
  if *cutoff_occurred?* **then return** *cutoff* **else return** *failure*

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complete?	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

Part C

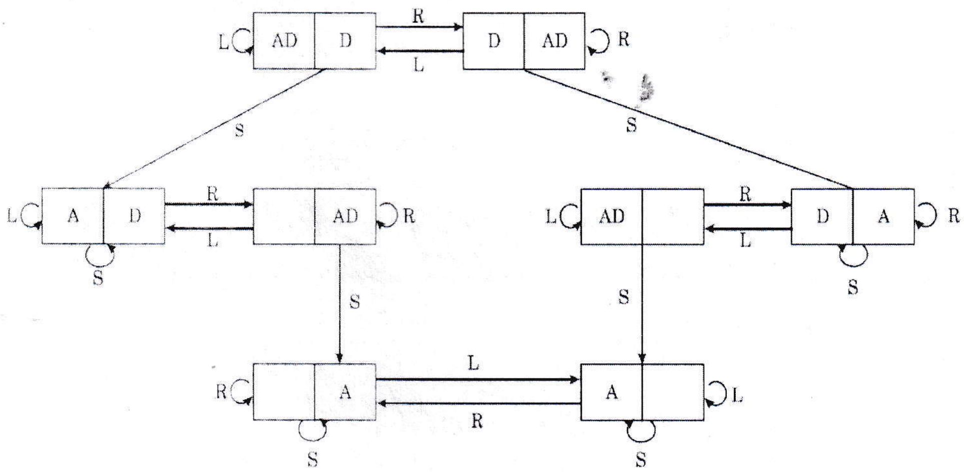
(6) marks

10. Vacuum world problem: Let the world contain just two locations. Each location may or may not contain dirt and the agent may be in one location or the other. The agent has three possible actions in this version of the vacuum world i.e. {Left, Right, Suck}

For the given problem it is possible to construct 8 states, where the goal is to clean up all the dirt, and the goal state is equivalent to state set {1,8}

State space with sensors

(5) marks



11. Genetic Algorithm (6) marks + application (4) marks.

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
    inputs:population, a set of individuals
           FITNESS-FN, a function that measures the fitness of an individual
    repeat
        new_population←empty set
        loop for i from 1 to SIZE(population) do
            x ← RANDOM-SELECTION(population, FITNESS-FN)
            y ← RANDOM-SELECTION (population, FITNESS-FN)
            child ← REPRODUCE(x, y)
            if (small random probability) then child ← MUTATE(child)
            add child to new_population
        population ← new_population
    until some individual is fit enough, or enough time has elapsed
    return the best individual in population, according to FITNESS-FN
```

```
function REPRODUCE(x, y) returns an individual
    inputs: x, y, parent individuals
    n ← LENGTH(x)
    c ← random number from 1 to n
    return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c+1, n))
```

Complexity is on the order of O(gnm))

12. A* search

The evaluation function is the addition of greedy search *h(n)* and uniform cost search g*(n)*

(ie) $f(n) = h(n)+g(n)$

$f(n)$ = cost of the cheapest solution through n

$h(n)$ = estimated cost of the cheapest path from node n to the goal state

$g(n)$ = actual path cost from the start node to node n

Greedy Best First Search

$f(n) = h(n)$

algs for both searches, BFS to tree search.

5 marks 1 function

2 define the parameters

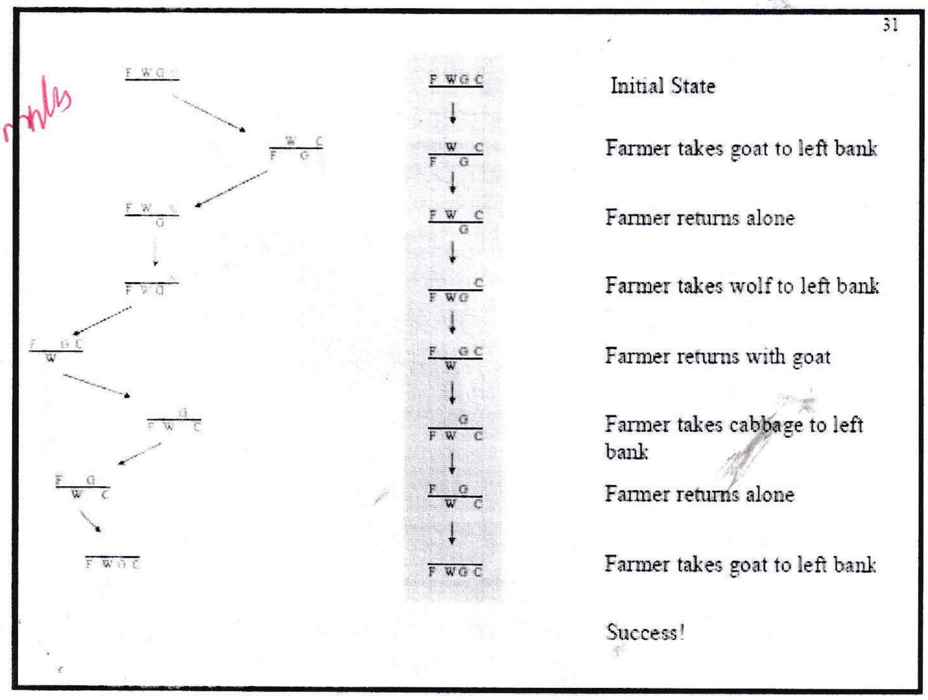
3 algs → Greedy BFS

A*

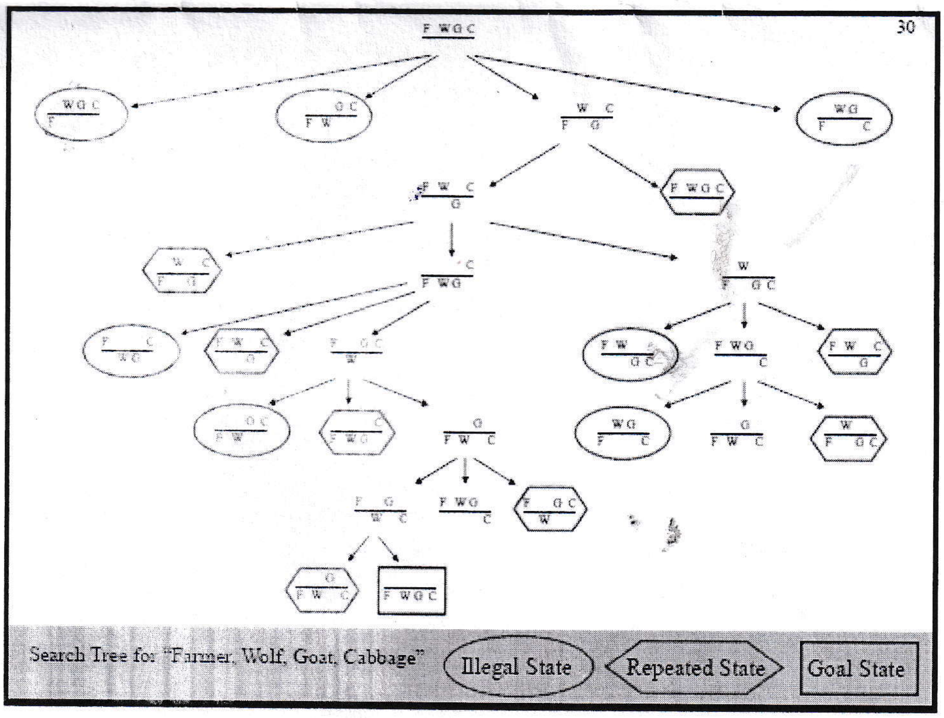
Best First search.

12. State: (farmer, fox, chicken, grain)
State: (farmer, wolf, goat, cabbage)

State
IS
GS
rules
(4) marks

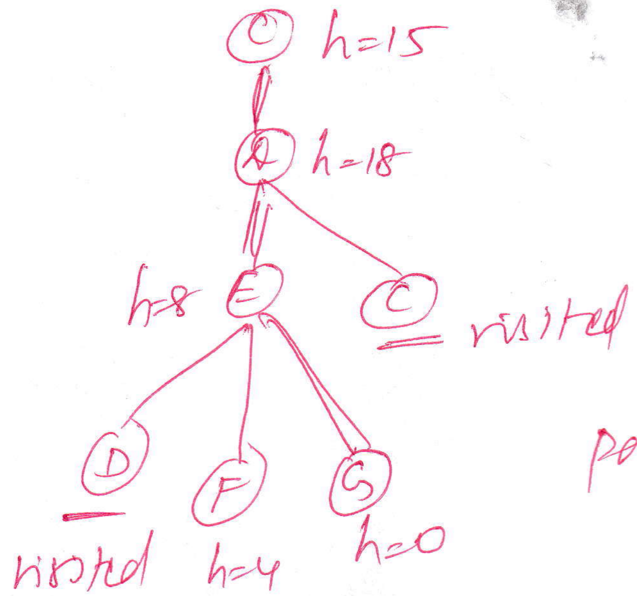


Solution
(6) marks



(12) $f(n) = h(n)$

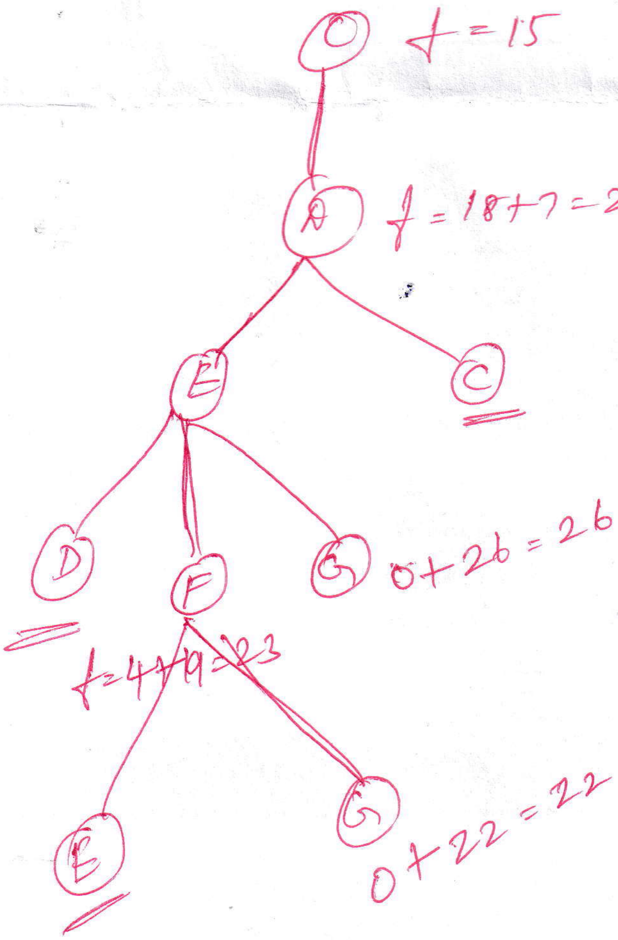
(8) marks.



path = C-D-E-G.

7+9+10 = 26

$f(n) = h(n) + g(n)$



path = C-D-E-F-G

7+9+3+3 = 22