Answer 1

Constraint satisfaction Problem V/s Problem solving using search

>> In constraint satisfaction Problem, we have a finite set of variables, nonempty domain of possible values for each variable and a finite set of constraints.
We wish to find an assignment that does not violate the constraints.
For Problem solving using search, we have start state, goal state and some permissible actions. We wish to reach goal state following any path.

>> In constraint satisfaction, first constraints are discovered and propagated as far as possible throughout the system. Then, if there is still not a solution, search begins.
Thus, it is a two step process.
In the latter approach, there is no propagation. We keep following path in the hope that it will lead to the goal state.

Cryptoarithmetic problem:

```
   EAR
 + EAR
 -----
  DRUM
```

Assumptions:-

1.) The values for all the alphabets are distinct.
2.) No alphabet has value 0 and value is single-digit.

```
    C2 C1
    E  A  R
 C3 E  A  R
 ----------
  D R  U  M
```

C_i's denote the carry generated.

$\boxed{\text{Initial state}}$

M is even
$\because D \neq 0$ and all values are single digit, carry ≤ 1
$\Rightarrow C_3 = 1$
$\Rightarrow D = 1$
$U = (2A + C_1) \% 10$
$R = (2E + C_2) \% 10$

let $R = 2$
$\Rightarrow M = 4$
$C_1 = 0$
$U = (2A) \% 10$
$2 = (2E + C_2) \% 10$
M is even $\Downarrow$
$E = 0$ or 1

For $C_2 = 0$ $E = 6$ [$\because$ For $C_2 = 1$ not possible]
$\because$ U is even
and $C_2 = 0 \Rightarrow A < 5$
$A = 3, 4$
3: U conflicts with E
4: conflicts with M
No value possible for A.

let $R = 3$
$\Rightarrow M = 6, C_1 = 0$
$\because$ U is even
$A = 2$ not possible (no value for E)
$A = 4$ not possible (no value for E)
$A = 6$ conflicts with U
$A = 7$
$\Rightarrow U = 4$
E no value of E possible
$A = 8$
U conflicts with M
$A = 9$
No value of E is possible.
Hence,
No value possible for A.

let $R = 4$
on Next page

Initial state

M is even
$D = 1$
$C_3 = 1$
$U = (2A + C_1) \% 10$
$R = (2E + C_2) \% 10$

$R = 2$
Not possible

$R = 3$
Not possible

$R = 4$
$\Rightarrow M = 8, C_1 = 0$
$A = 2$ not possible
(U conflicts with R)

let $A = 3$
$\Rightarrow U = 6, C_2 = 0$
$10 + 4 = 2E + 0$
$\Rightarrow E = 7$

All alphabets are assigned values. No conflicts.
we stop here.

Finally,
$A = 3$
$E = 7$
$U = 6$
$D = 1$
$M = 8$
$R = 4$
$C_1 = 0$
$C_2 = 0$
$C_3 = 1$

$$\begin{array}{rrrr} & 0 & 0 & \\ & 7 & 3 & 4 \\ 1 & 7 & 3 & 4 \\ \hline 1 & 4 & 6 & 8 \end{array} \qquad \begin{array}{r} EAR \\ +\ EAR \\ \hline DRUM \end{array}$$

Answer 3.

Underestimation and overestimation of a heuristic function

A heuristic functions gives an estimated cost of reaching goal state from current state.

let h = actual cost of getting from current state to goal state.

h^* = estimate of h [our heuristic function].

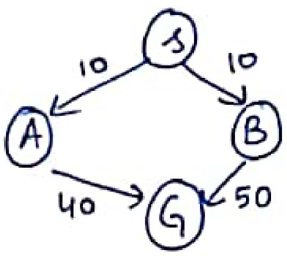
Underestimation : when $h^* \leq h$

our heuristic function underestimates the cost of getting from current state to goal state.
In this case, we always get the optimal solution.

Overestimation : when $h^* > h$

our heuristic function overestimates the cost of getting from current state to goal state.
In this case, we cannot guarantee finding the cheapest path solution unless we expand the entire graph until all paths are longer than the best solution.

Example :



For A* algorithm
$f(n) = g(n) + h(n)$

$h^*(A) = 30$, $h^*(B) = 20$
Underestimation

$S \to A$ $f(A) = 40$
$S \to B$ $f(B) = 30$ ✓
$S \to B \to G$ $f(G) = 60 < 40$
$S \to A \to G$ $f(G) = 50$ ✓
optimal solution obtained

$h^*(A) = 80$, $h^*(B) = 20$
Overestimation

$S \to A$ $f(A) = 90$
$S \to B$ $f(B) = 30$ ✓
$S \to B \to G$ $f(G) = 80$ ✓
optimal solution is not reached

Conditions under which A^* algorithm gives optimal solution

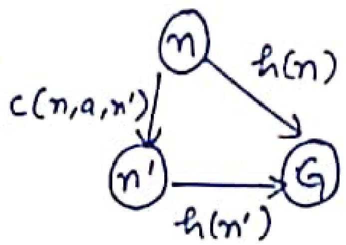
1.) A^* returns optimal solution if $h(n)$ is admissible – it never overestimates true cost to nearest goal.

$h(n) \leq h^*(n)$.

As shown in previous example for $h(n) > h^*(n)$ optimal solution is not reached.

2.) A^* return optimal solution if $h(n)$ is consistent – for every node n, every successor n' of n generated by action a,

$$h(n) \leq c(n,a,n') + h(n')$$

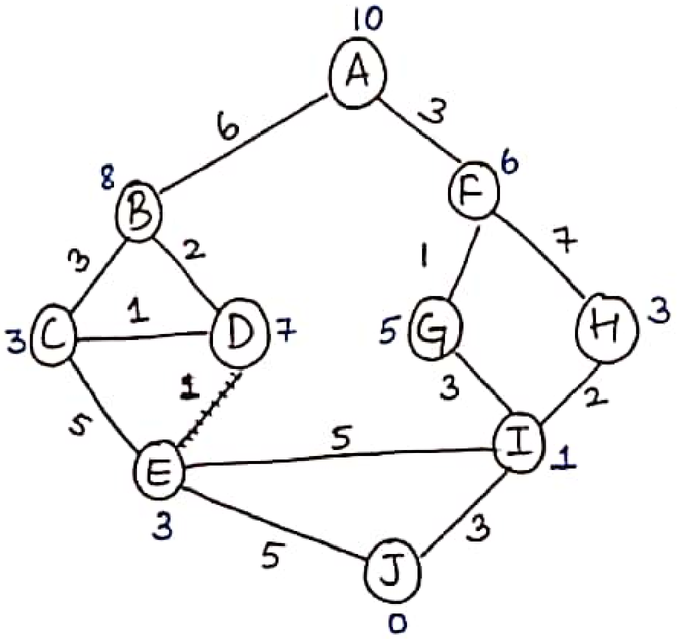


let n' is successor of n, then

$$\begin{aligned} g(n') &= g(n') + h(n') \\ &= g(n) + c(n,a,n') + h(n') \\ &> g(n) + h(n) \\ &= f(n) \end{aligned}$$

$f(n') \geq f(n)$

$\Rightarrow$ $f(n)$ is non decreasing along any path – which is desirable.



The numbers on edge represent distance between the nodes.

The numbers written on node represent heuristic value $h(n)$, where $n = A, B, \ldots, J$

start state : A goal state : J

(a) Greedy best first search algorithm

Evaluation function $f(n) = h(n)$

Start state : A

Step1: A is expanded to get B and F.

$h(B) > h(F)$.

Step2: F is expanded to get G and H.

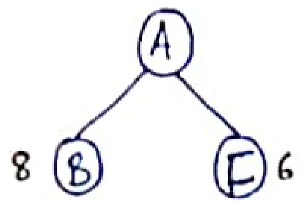
$h(G) > h(H)$

Step3: H is expanded to get I.

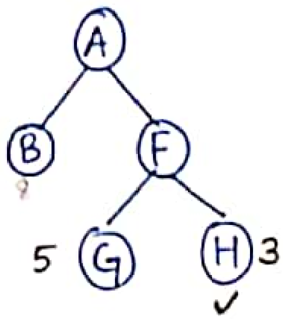
Step4: I is expanded to get E and J.

$h(E) > h(J)$.

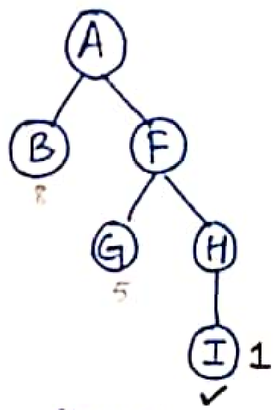
Goal state J is reached.



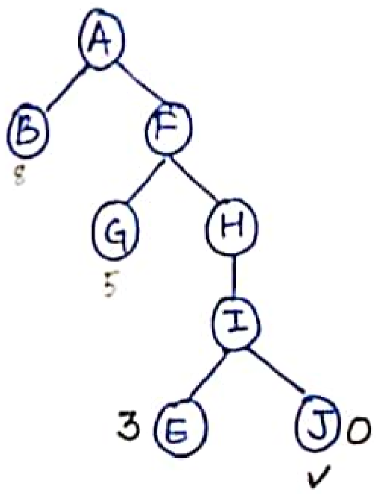
step 1:



step2:



Step 3



step4.

Greedy best first search on graph given

(b) A^* algorithm

n	$h(n)$
A	10
B	8
C	3
D	7
E	3
F	6
G	5
H	3
I	1
J	0

Start state : A

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

Step 1

A 10

$A \to B$ $f(B) = 8+6 = 14$

$A \to F$ $f(F) = 6+3 = 9$ ✓

$f(B) > f(F)$

Path $A \to F$ is selected.

Step 2

$A \to F \to G$ $f(G) = (\overset{3}{\cancel{4}} + 1) + 5 = 9$ ✓

$A \to F \to H$ $f(H) = (\overset{3}{\cancel{4}} + 7) + 3 = 13$

Out of $A \to B$, $A \to F \to G$ and $A \to F \to H$,

Path $A \to F \to G$ has smallest value of f.

Path $A \to F \to G$ is selected.

Step 3

$A \to F \to G \to I$ $f(I) = 7 + 1 = 8$ ✓

Out of $A \to B$, $A \to F \to H$ and $A \to F \to G \to I$,
path $A \to F \to G \to I$ is selected as it has the smallest value of f.

Step 4

$A \to F \to G \to I \to E$ $f(E) = 12+3 = 15$

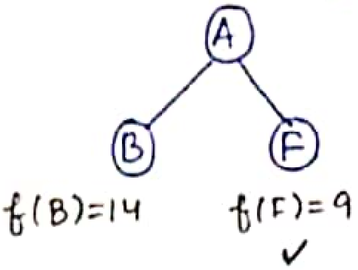
$A \to F \to G \to I \to J$ $f(J) = 10+0 = 10$

Out of paths $A \to B$, $A \to F \to H$, $A \to F \to G \to I \to E$ and $A \to F \to G \to I \to J$, path $A \to F \to G \to I \to J$ is selected, as it has the smallest value of f.

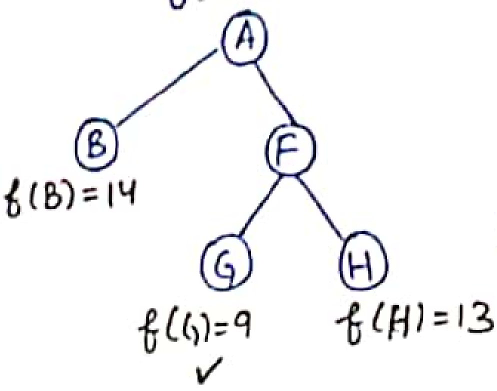
We have reached goal state J.

$\therefore$ We stop here.

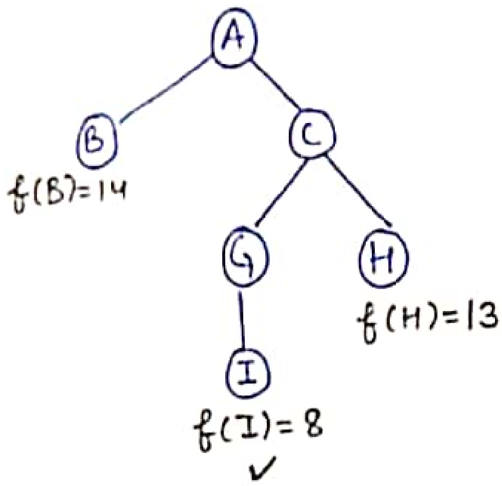
Diagrametically,



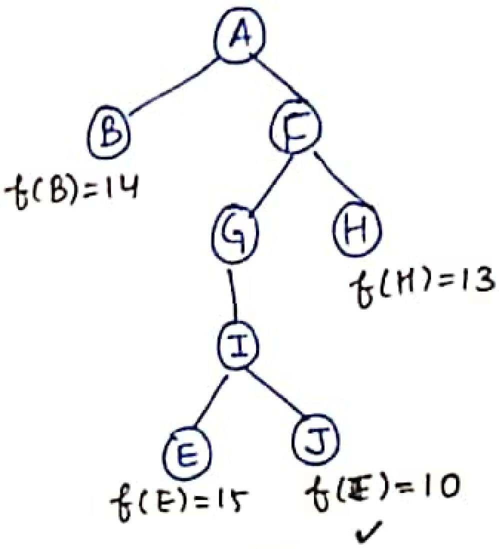
Step 1.



step2



step3



step4

A* algorithm on graph given

Answer 5

(a) Everyone who loves all animals is loved by someone

S1 $\forall x\,[\forall y\ Animal(y) \rightarrow loves(x,y)] \rightarrow [\exists z\ loves(z,x)]$

Anyone who kills an animal is loved by noone.

S2 $\forall x\,[\exists y\ Animal(y) \wedge kills(x,y) \rightarrow [\forall z\ \neg loves(z,x)]$

~~John~~ Jack loves all animals.

S3 $\forall x\ Animal(x) \rightarrow loves(Jack, x)$

Either Jack or John killed the cat, who is named Pussy

S4 $kills(Jack, Pussy) \vee kills(Ravi, Pussy) \wedge Cat(Pussy)$

Cats are animals.

S5 $\forall x\ Cat(x) \rightarrow Animal(x)$

Predicates Used :

$Animal(x)$: x is an animal

$Cat(x)$: x is a cat

$loves(x,y)$: x loves y

$Kills(x,y)$: x killed y

(b) Converting to Prenex Normal Form

S1: $\forall x\,[\forall y \neg Animal(y) \vee loves(x,y)] \rightarrow [\exists z\ loves(z,x)]$

$\forall x\,[\exists y\ Animal(y) \wedge \neg loves(x,y)] \vee [\exists z\ loves(z,x)]$

S2: $\forall x\,[\forall y\ \neg Animal(y) \vee \neg Kills(x,y)] \vee [\forall z\ \neg loves(z,x)]$

S3: $\forall x\ \neg Animal(x) \vee loves(Jack, x)$

S4: $kills(Jack, Pussy) \vee kills(Ravi, Pussy) \wedge Cat(Pussy)$

S5: $\forall x\ \neg Cat(x) \vee Animal(x)$

(c) <u>Skolmize the formulae</u>

<u>S1</u>: $\forall x\ [Animal(F(x)) \wedge \neg loves(x, F(x))] \vee loves(G(x), x)$

Removing universal quantifier,

$= [Animal(F(a)) \hat{\vee} \neg loves(a, F(a))] \vee loves(G(a), a)$

$= [Animal(F(a) \vee loves(G(a), a)] \wedge$

$[\cancel{Ani} \cdot \neg loves(a, F(a)) \vee loves(G(a), a)]$

<u>S2</u>: $\neg Animal(b) \vee \neg Kills(a, b) \vee \neg loves(c, a)$

<u>S3</u>: $\neg Animal(a) \vee loves(Jack, a)$

<u>S4</u>: $Kills(Jack, Pussy) \vee Kills(John, Pussy) \wedge Cat(Pussy)$

<u>S5</u>: $\neg Cat(a) \vee Animal(a)$

<u>Finally</u>, In CNF

S1 1. $Animal(F(a)) \vee loves(G(a), a)$

S1 2. $\neg loves(a, F(a)) \vee loves(G(a), a)$

S2 3. $\neg Animal(b) \vee \neg Kills(a, b) \vee \neg loves(c, a)$

S3 4. $\neg Animal(a) \vee loves(Jack, a)$

S4 5. $Kills(Jack, Pussy) \vee Kills(John, Pussy)$

S4 6. $Cat(Pussy)$

S5 7. $\neg Cat(a) \vee Animal(a)$

(d) Who Killed Pussy?

1. Animal(f(a)) v loves(G(a),a)
2. ¬loves(a,f(a)) v loves(G(a),a)
3. ¬Animal(b) v ¬kills(a,b) v ¬loves(c,a)
4. ¬Animal(a) v loves(Jack,a)
5. kills(Jack,Pussy) v kills(John,Pussy)
6. Cat(Pussy)
7. ¬Cat(a) v Animal(a)
8. Animal(Pussy) Using 6 & 7
9. loves(Jack,Pussy) Using 4 & 8
10. ¬kills(Jack,Pussy) Using 3,8 & 9
11. kills(John,Pussy) Using 5 & 11

Hence, using resolution algorithm, we infered following :-

¬kills(Jack,Pussy) → Jack did not kill Pussy

kills(John,Pussy) → John killed Pussy.

Who killed Pussy? John.

Answer6.

Differentiate the following

(a) Turing Test approach Vs Rational agent approach

Turing Test approach	Rational agent approach
It concerns on acting humanly.	It concerns on acting rationally.
How human acts in a given situation.	Acting so as to achieve one's goals, given one's beliefs.
Human behavior is well adapted for one specific environment and is defined by sum of total of things that human do.	It is more amenable to scientific development as the standard of rationality is clearly defined and completely general.
This approach gives optimal solution since solvable by humans.	This approach gives the best expected outcome if best outcome is not achievable.

(b) Knowledge based systems Vs Expert systems

Knowledge based systems	Expert systems
It uses reasons and a knowledge base to solve complex problems.	It emulates decision-making ability of a human expert.
It is divided into Logic, Frames, Rules or Semantic Networks for knowledge representation.	It is divided into two subsystems: the knowledge base and inference engine. (rule interpreter).
It represent knowledge in a declarative way	It ~~is~~ represents knowledge as if-then rules. Example: MYSIN system.

(c)

Best first search Vs	Breadth first search.
It is informed search strategy - it has additional information available(about the goal).	It is uninformed search strategy - uses only the information available in the problem definition.
It uses an evaluation ~~evaluat~~ function f(n).	It does not uses any evaluation function.
It selects the node which has lowest value of f(n).	It expands/selects the shallowest unexpanded node.
It uses advantages of both breadth first and depth first search.	It does not resemble to depth first search.

(d)

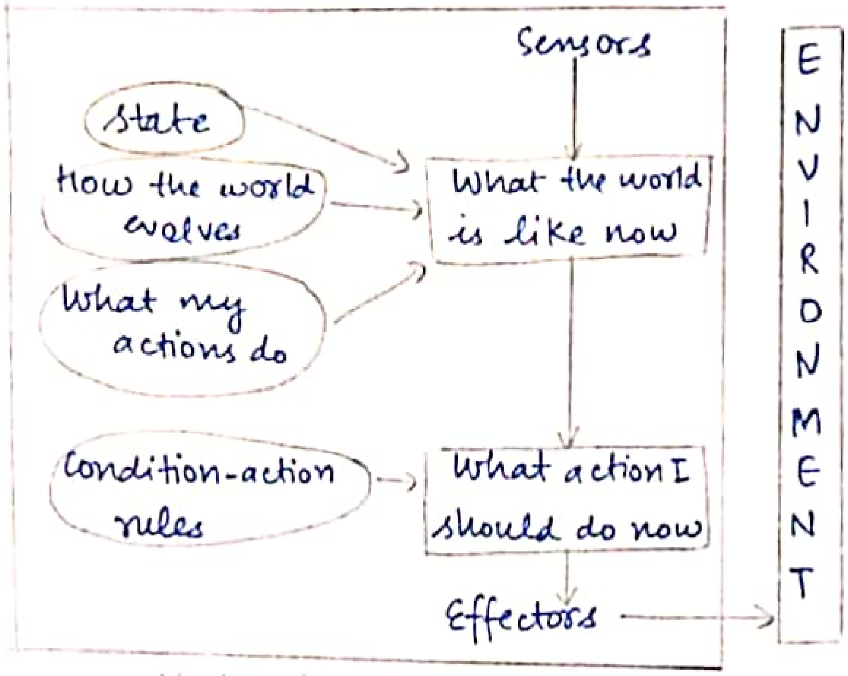
Hill climbing search Vs	Iterative deepening search
It is informed search - has additional information about the goal.	It is uninformed search - has no information other than problem definition.
It is a local search algorithm and checks only the immediate neighbours and selects the most promising node from them.	It performs depth-first search up to a certain "depth limit" and keeps increasing it after each iteration.
It may fail to find a solution due to local maximum, plateau or ridge.	It always find solution, if solution exists.
It stops if there are no successor states with better values than the current state.	It stops if goal node is found or when all the possibilities (depth) are executed/exhausted.

(e) Substitution Vs Unification

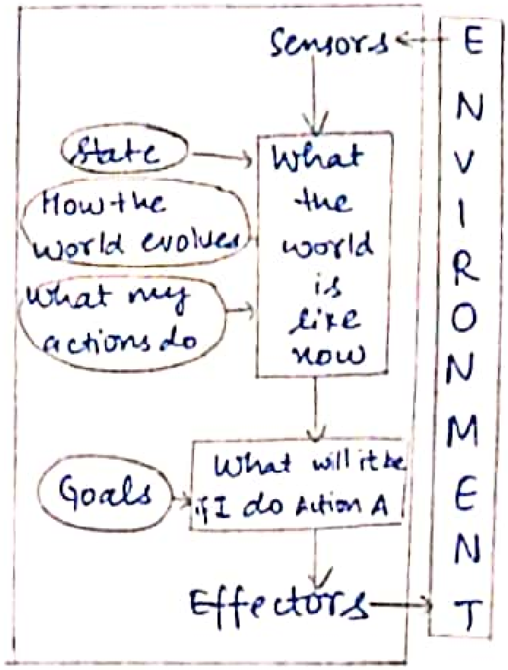
Substitution	Unification
A substitution is a finite set of the form $\{t_1\|v_1, t_2\|v_2, \ldots, t_n\|v_n\}$ where every variable v_i is different from term t_i and no two elements in the set have the same variable after the stroke symbol.	Unification is a process of determining and applying a certain substitution to a set of expressions in order to make them identical.
It does not uses disagreement set.	It requires a notion of disagreement set.
There is no predefined goal, substitution is applied to 'unify' terms and formulae.	The goal is to find a most general unifier (m.g.u.)
It allows use of $\{f(x)\|x\}$ i.e., it is a valid substitution.	It does not allow use of $\{f(x)\|x\}$ since it would not lead to obtaining m.g.u.

(f) Model based agent Vs Goal based agent

Model based agent	Goal based agent
It keeps some internal state that depends on the percept history.	It is an extension of model based agent, it also uses goals info in addition to internal state.
It takes the decision based on how its actions affect the world.	It takes the decision on the basis if I am happy?
The goal cannot be updated.	The goal can be updated.
No planning involved.	It performs planning and searching.



Model Based Agent



Goal Based Agent

(g) Problem solving using Search V/s Planning

Problem solving using Search	Planning
It searches the state space of possible actions, starting from initial state.	It involves task of coming up with a sequence of actions that will achieve a goal component of planning system.
In this the problem is solved in order. It follows any path that it believes will lead it to goal state - not subgoal state.	The planner does not have to solve the problem in order - it can suggest actions to solve any subgoals at anytime.
There is no subgoaling, so problem is not divided into chunks.	The planner assumes that most part of the world are independent so they can be stripped apart and solved individually