Page: Date:

Chapter - 1

* What is AI ?

AI is a branch of computer science concerned with making computers behave like humans.

In other words, AI is the simulation of human intelligence by machines. These process includes :-

* Learning

the calculation of information and rules for using the information.

* Reasoning

Using the rules to reach approximate or definite conclusions.

* Self correction

Thus, AI is the branch of computer science that concerned with the study and creation of computer system that exhibit human-like intelligence that

→ can learn new concepts
→ can understood the natural language &
→ can perform other types of tasks which requires human-type intelligence.

On summarizing above we get,
An AI system
→ Should have reasoning capabilities,

→ Should have capabilities of Natural Language Processing (NLP).

→ Should have capability of learning from past experience.

→ Should have capability of self correction.

NLP Task:
→ Machine Translation (MT) → summarization
→ Information Retrieval (IR)

Some AI related Fields:
→ Game playing
→ Expert System
eg - MYCIN
→ analyses system and Draw conclusion
→ NLP (Natural Language Processing)
→ Neural Networks
→ Robotics
→ Computer vision
- receives images, analyzes & draw conclusions
→ Image processing
→ Voice Recognition
→ Industrial Control

Applications of AI :

→ Game playing
→ Speech recognition
→ Understanding Natural language
→ Computer vision [eg - traffic System]
→ Expert System

v. Imp

facts, data, knowledge ; Belief & hypothesis

facts : Something that actually exists, reality, truth
→ Something known to be true
eg - Sun rises in the east.

Data : → is a collection of facts such as values or measurements.
→ Data is information that has been translated into a form that is more convenient to process
→ types of data
→ qualitative data
eg - a good student.
→ quantitative data
no. of student of this ~~dat~~ class = 28

knowledge :
→ It is a familarity with someone / something which can include information, facts, description or skill acquired

through experience or education.

Belief :
→ Something accepted as true,
→ are assumed truths,
→ A false belief isn't considered to be knowledge.

Hypothesis :
→ It is an explanation of a phenomenon that can be tested in a way which proves or disproves it.
→ In the time of testing, the hypothesis is taken to true.

knowledge and it's types :-

① Relational knowledge
② Procedural knowledge
③ Declarative knowledge
④ Heuristic knowledge

Procedural knowledge :
→ How to do something is procedural knowledge
→ It is ~~structure~~ instruction oriented (contain instruction stepwise).

→ Focus on how to obtain an a result.
→ eg :- the steps used to solve an algebric equation.

- Declarative knowledge :
→ It is a factual knowledge (eg - knowing that a cathode ray tube is use to project a picture in most televisions is a declarative knowledge.)

→ It is also known as descriptive knowledge,

→ It describes how things are,

→ It is assertion oriented (act of stating something)

→ It focuses on describing the properties.

- Relational knowledge
→ The simplest way of storing facts is to use a relational method where each fact about a set of objects is set out systematically in columns.

→ This representation gives little opportunity for inference, but it can be used as the knowledge basis for inference engines

eg.

Musician	Style	Instrument	Age
~~Miles~~ Miles Davis	Jazz	Trumpet	deceased
John zorn	Avant Garde	Saxophone	35
Frank Martin	Rock	Guitar	deceased
Jack Russel	Jazz	Guitar	47

fig :- Relational knowledge

We can ask things like
* who is dead ?
* who plays Jazz / Trumpet etc

Note: This sort of representation is popular in database systems.

- Heuristic search
  → The knowledge provided by experts of certain domains, subjects, disciplines and fields is known as the Heuristic knowledge, which they have been obtained after years of experience.

  → This type of knowledge helps in taking the best approach to particular problems & making decisions.

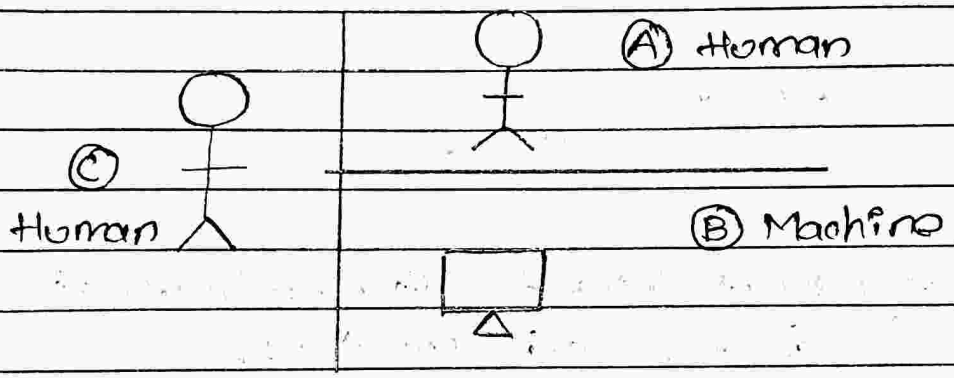
→ eg. chess game (moving to win doing some guess).

Imp

Turing Test :-

→ This test was purposed by Alan Turing in 1950

→ It is a test of machine's ability to exhibit intelligent behaviour.



→ The computer / Machine should need to Pass the following capabilities to pass the Turing test :

- Natural Language Processing (NLP)
- Knowledge Representation
- Automating reasoning
- Machine learning.

Intelligent Agent

→ An agent is anything that can be viewed as perceiving it's environment through sensors and acting upon that environment through actuators

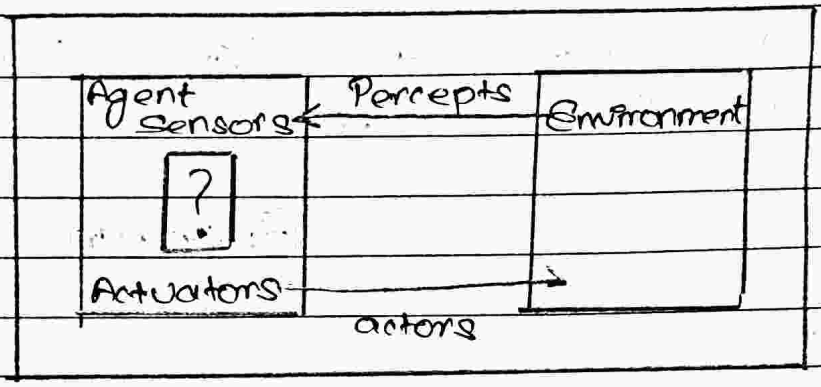


Fig: Agent interact with environments through sensors and actuators,

eg- human agent

sensors ⟶ eyes, ears

actuators ⟶ hands, legs, mouth

- Note: The term percept refers to the perceptual inputs of any given instant.

* Percept sequence is a complete history of everything the agent has ever perceived.

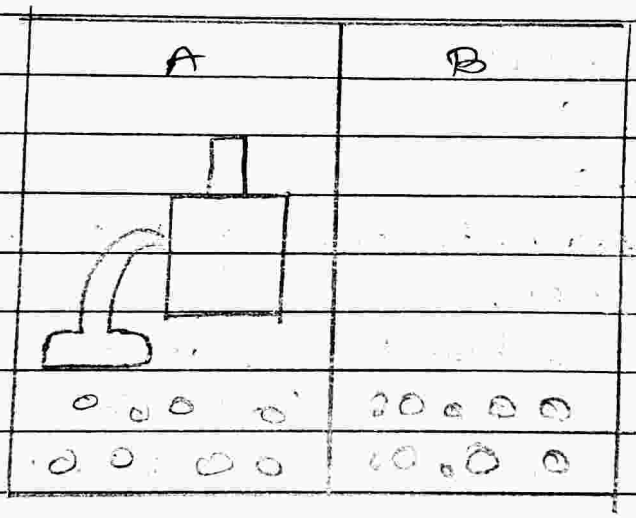
* Agent's behavior is described by the agent function that maps any given percept sequence

to an action.

* The agent function for an artificial agent will be implemented by an agent program.

* The agent function is an abstract mathematical description while the agent program is a concrete implementation running on the agent architecture.

eg- ~~The~~ The Vacuum Cleaner World:



→ We have two locations: square A & Square B

→ The vacuum gets perceives which square it is in whether there is dirt in the square.

→ It can choose to move left, move right, suck up the dirt or do nothing.

→ Agent Function: If the current square is dirty then suck otherwise move to other square.

→ A partial tabulations to this agent function is shown below:

Percept Sequence	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	Left
[B, dirty]	Suck

Structure of Agents

Agent Program:
→ that implements the agent function mapping percept to action.

agent = architecture + program
(Hardware Part that has sensor and actuators)

Types of agent
(i) Simple reflex Agent
(ii) Model-based reflex Agent
(iii) Goal-based Agent
(iv) Utility based Agent

Simple reflex Agent

→ These agents select action on the basis of the current percept, ignoring the rest of percept history.

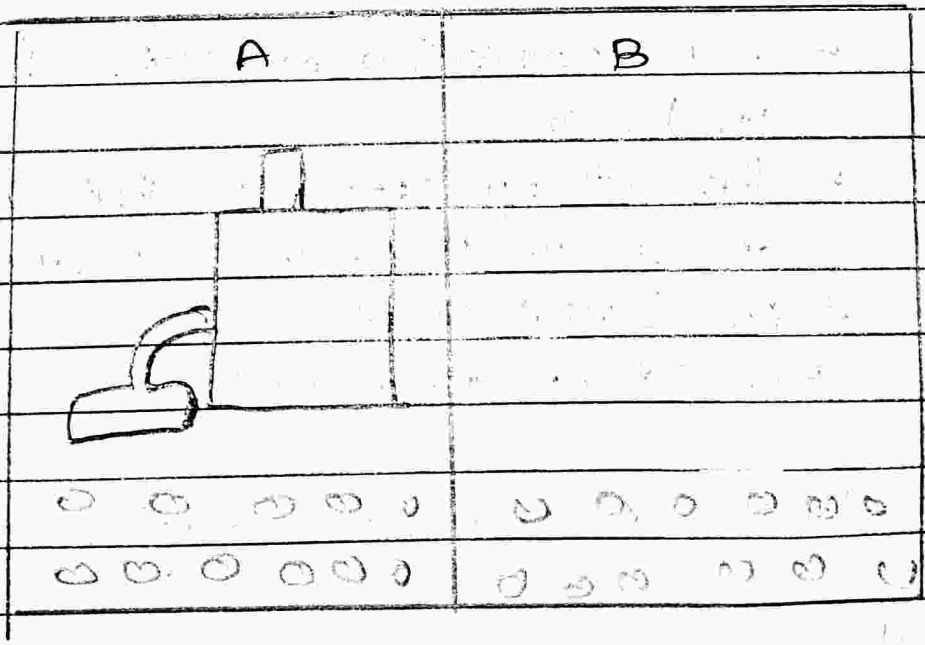
eg →

Let us take the vacuum agent program for a simple reflex agent.

function REFLEX_VACCUM_AGENT ([Location, status]) returns an action

If status = Dirty then returns suck.
elseif location = A then return Right.
elseif location = B then return Left.

fig 1: The agent program for a simple reflex agent in the two state vaccum environment.



Ch-2

Problem Solving

Steps to solve a Particular Problem :-

* Define the problem precisely.
* Analyze the problem.
* Represent the knowledge.
* Find out the solutions to choose the best one.

v. Important

Defining problem as a state space search :-

State space :-

→ represents a problem interms of states and operators that change the states.

→ State space consists of
* a representation of states that the system can be in.
* the set of operators that change a state of system from one to another.
* An initial state
* a set of final state.

Terminologies in problem Solving :

→ Goal

→ Goal formation :-

→ based on the current situation, it is process of finding out which sequence of actions will get to a goal.

→ Problem formulation :-

→ It is the process of deciding what actions & states to consider to reach a given goal.

→ Search :-

→ Find out a goal from doing initial state to final state.

v.imp

Well defined problem and Solution :-

→ A problem can be defined by 4 components:-

* The initial state
* Description of possible actions.
* The goal test (to check whether it is final state or not)
* A path cost

→ assign cost to every path so which way we reach to final state by estimate low cost.

eg → Travel Salesman Problem (TSP)

eg → Problem formulation of 8- Puzzle

eg → Problem formulation of 8-puzzle

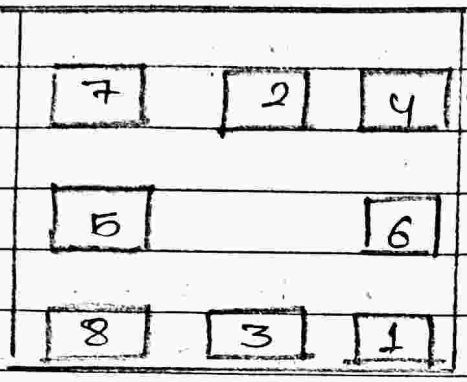


Fig 1 : Initial State

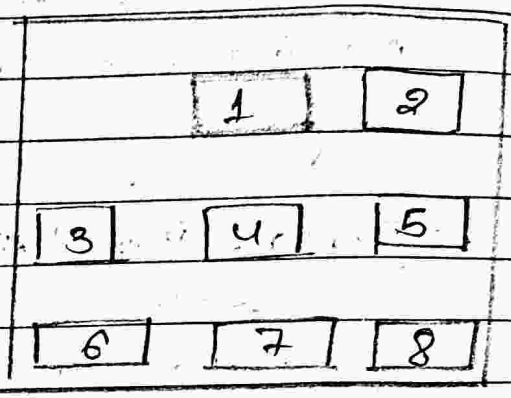


Fig 2 : Goal State

→ States :
- → location of tiles.

→ Initial State :
- → given in fig (1)

→ Action (operators)
- → move blank left
- → move blank right
- → Move up
- → Move down

→ Goal State
- → given in fig (2)

→ Path cost
- → 1 per move (1 unit)

Tuesday test

Page: Date:

Formulate the Problem:

eg - The water Jug Problem

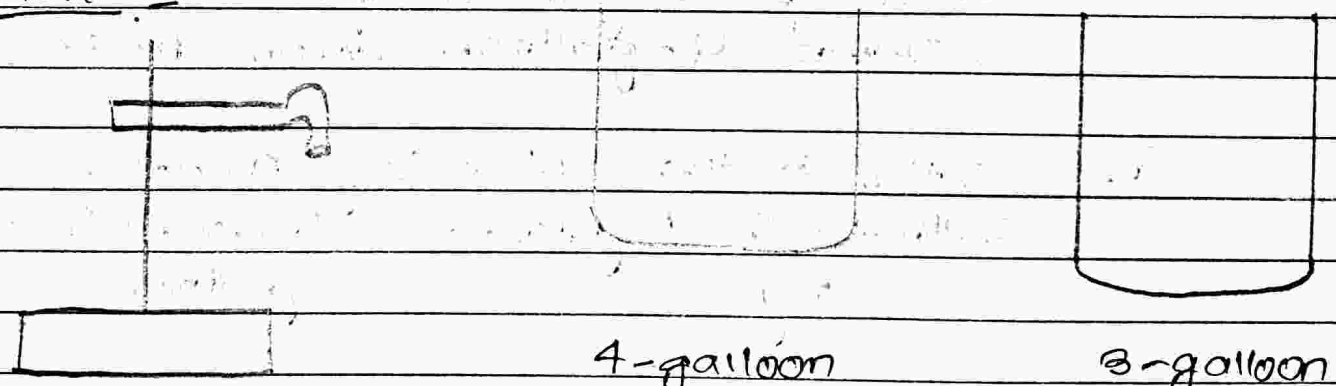
→ You have a 4 galloon and 3-galloon water jug without any measuring markers on them.

→ You have a faucet (tap) with an unlimited amount of water.

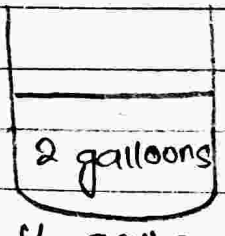
→ You need to get exactly 2 galloons in 4 galloon jug.

Soln:

Start:



Goal:



4 galloon jug

① State representation :-
- (x, y) where x : content of 4-galloon
y : content of 3-galloon

② Initial State → (0, 0)

③ Goal State → (2, n)
↳ 1 to 3 only

④ Actions (operators) :
→ Fill 3 galloon from faucet.
→ Fill 4 galloon from faucet.
→ Fill 3 galloon from 4-galloon
→ Fill 4 galloon from 3-galloon
→ empty 3-galloon into 4-galloon
→ empty 4-galloon into 3-galloon
→ Dump 3-galloon ~~into~~ down drain
→ Dump 4-galloon down drain.

One solution to the water jug problem

Galloons in 4 galloons jug	Galloons in 3 galloon jug
0	0
0	3
3	0
3	3
4	2
0	2
2	0

Century
Page : Date :

Path Cost :- 6 per Move (6 Unit)
(As long More Move More accuracy)

Assignment :- (2080/01/17)

(1) State the 8-queen Problem and formulate this Problem (State / operator / Goal Test)

(2) Human Versus Machine Performance.

(3) What are the aspects of a formal definition of a problem? Explain with example.

(4) What do you mean by well-defined problem? Explain the steps of problem solving?

(5) What do you mean by production system in AI? What are the major components of AI Production System?

(6) Discuss the role of control strategy in AI production system?

(7) What do you mean by constraint satisfaction problem? explain with some example.

Century

Searching for solutions

→ Search tree (Search graph):
→ It is generated by the initial state and successor function that together define the state space.

→ Search node:
→ the root of the search tree.

→ Search strategy:
determine which state to expand.

Measuring problem-solving Performance
There are 4 ways to evaluate an algorithm Performance :-

① Completeness:
→ It is the algorithm guaranteed to find out the solution when there is one?

② Optimality:
→ Does the strategy find the optimal (favorable) solution?

③ Time complexity
How long does it take to find a solution?

(3) Space Complexity
→ How much memory is needed to perform the search?

Production System
→ Since search forms the core of many intelligent process, it is useful to structure AI programs in a way that facilitates describing & performing the search process.

→ Production systems provide such structure.

→ Components of a production system.
* a set of rules.
* one or more knowledge / database.
* a control strategy.
* specifies the order of rules to be applied.
* resolve conflicts that arise when several rules match at once.

→ A rule Applier.

Imp

Constraint Satisfaction Problem

CSP are the problems in which the goal is to find some problem state that satisfies a given set of constraints.

→ A CSP is characterized by
a) A set of variables ($x_1, x_2, \ldots x_n$)
b) for each variables x_i, a domain D_i with the possible values for that variable.
c) a set of constraints i.e relations that are assumed to hold between the values of the variables.

Eg :-
1) The n-queen problem
2) A crossword problem
3) Crypto arithmetic problem
4) Soduku etc.

Eg 3) Crypto arithmetic problem
Solve the following problem

```
  SEND
+ MORE      S+M = M0 = 1
-------
 MONEY
```

soln :
Initial state :-
→ No letter have the same value.

→ The sum of digit is shown in the table,

step one

If we add 4 digit no, like SEND and MORE, the output come as MONEY with 5 digits. It is obvious that S+M gives a carry out result 'MO', so M=1.

step two

Since there are 5 choices as

S = 5, 6, 7, 8, 9 and M=1

	9	E	N	D
+	8	O	R	E
	1	6	N	E	Y

	9	E	N	D
+	1	O	R	E
	10	O	N	E	Y

step 3

Since there is

	9	2	N	D
+	1	O	R	E
	10	2	E	Y

S=9
M=1
O=0

SEND	TWO	SEND
+ MORE	+ TWO	MORE
MONEY	FOUR	MONEY

① SEND
+ 1ORE
10NEY

② 95ND
+ 1ORE
10NEY

③ 9 5 6 7 → E
1 0 8 5
10 6 5 2 → E

provid

step-1

S → 9 M → 1
E → 5 O → 0
N → 6 R → 8
D → 7

step-2

S = 9 D → 5 R → 3
E = 2 M → 1 Y → 7
N = 4 O → 0

9 2 4 5
1 0 8 2
10 3 2 7 → E - value

V. Important

Search Strategies

(i) Uninformed Search
(ii) Informed Search

(i) Uninformed Search

→ It is also known as binary search.
→ It uses no information other than
  → initial state
  → Search operator
  → a goal state

→ There is no information about
  * number of steps or path cost from current state to Goal.

→ All they can do is distinguish a goal state from a non-goal state.

→ The following are the uninformed search techniques :-
(a) Breadth-first search
(b) Depth-first search
(c) Death-limit search
(d) Bi-directional search
(e) Uniform cost Search
(f) Iterating deepening Search etc.

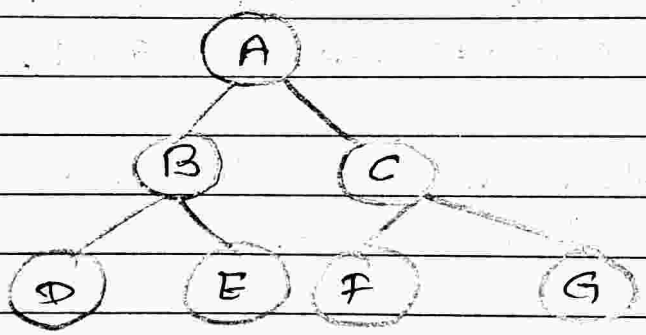
(a) Breadth - First Search

→ It is a simple search technique in which the root node is expanded first, then all the successors of the root node are expanded, then their successors and so on.

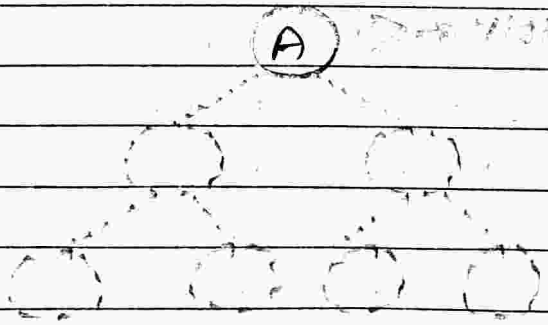
→ All the nodes are expanded at a given depth in a search tree before any nodes at the next level are expanded.

→ BFS is implemented using queue assuming that the node that are visited first are expanded first.

eg :→

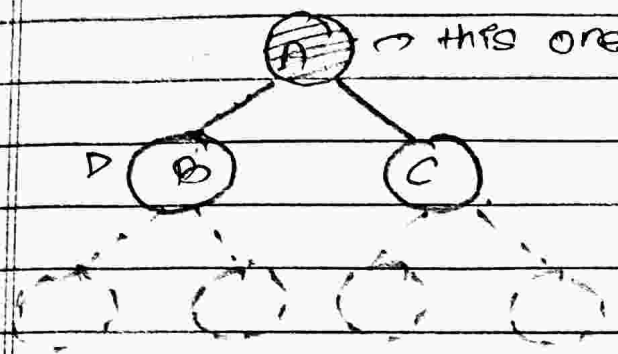


1st step :



Q = (A)

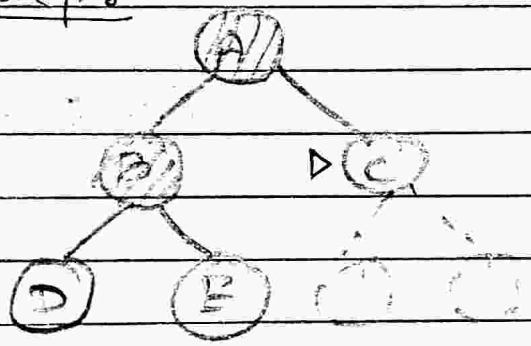
2nd step :-



this one is expanded

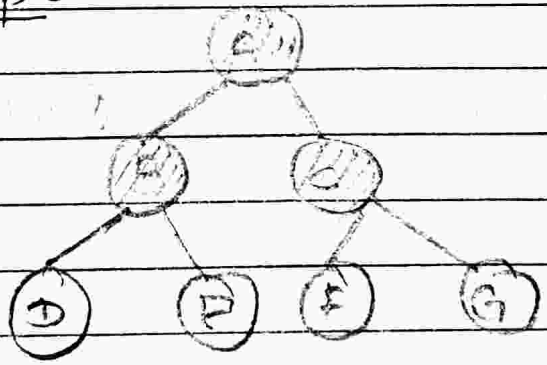
O = (B, C)

3rd step :-



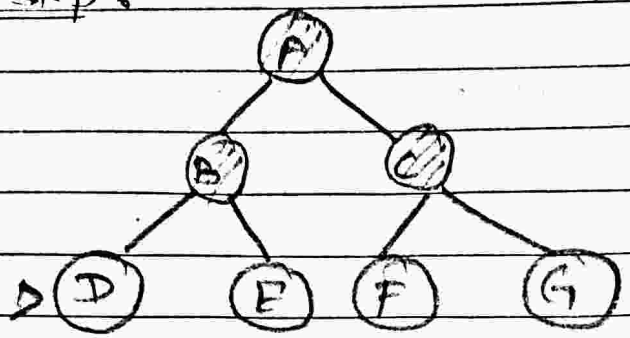
O = (C, D, E)

4th step :-



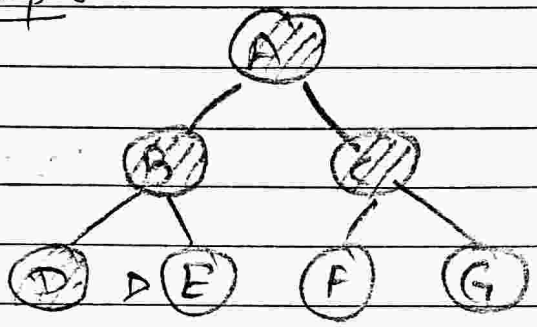
O = (D, E, F, G)

5th step :-



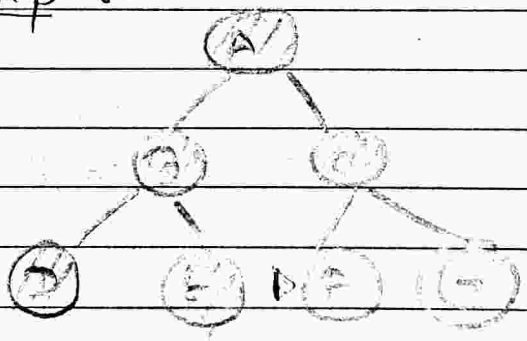
O = (E, F, G)

6th step :-



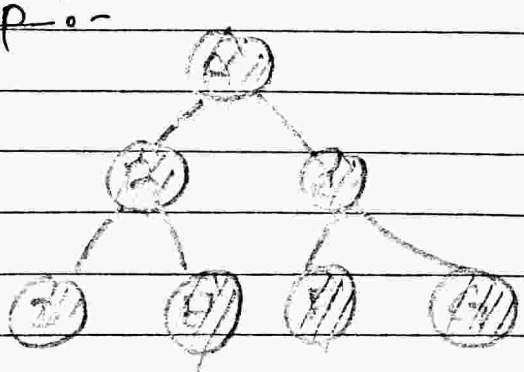
O = (F, G)

7th step :-



O = (G)

8th step :-



O = { } - empty

Page : Date :

BFS evaluation

→ Completeness : Yes

→ Time Complexity → order

$O(b^{d+1})$

where, b → branching factor (no of succ)
d → depth (solution is at depth d)

→ ~~Spa~~ Space complexity : $O(b^{d+1})$
↳ order

→ Optimality :- Yes

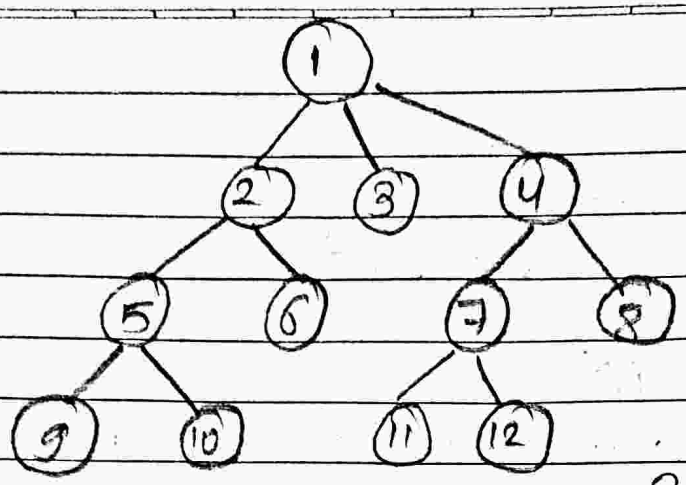
Complexity

$f(n) = 1 + b + b^2 + \ldots\ldots (b^{d+1} - b)$

$= O(b^{d+1})$ ↓

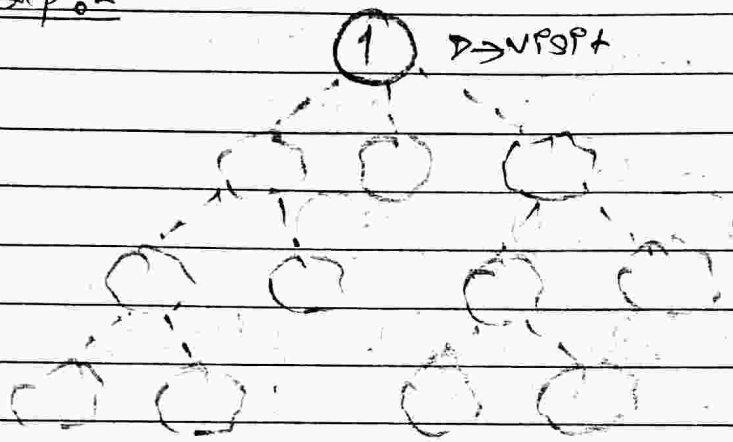
complexity can go never greater than b^{d+1})

eg :-



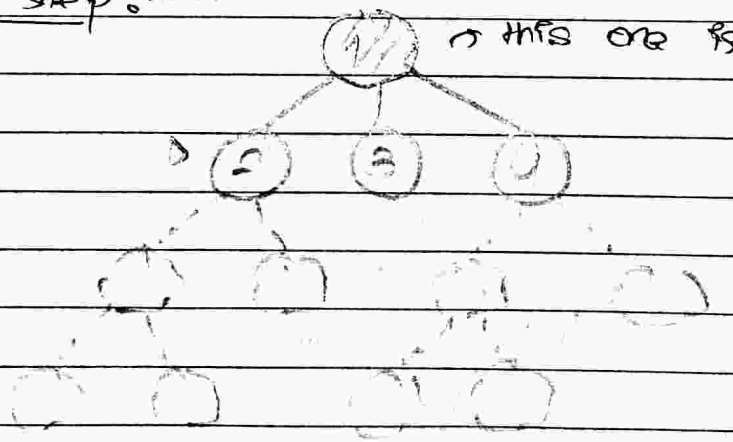
Perform BFS Search

1st step :-



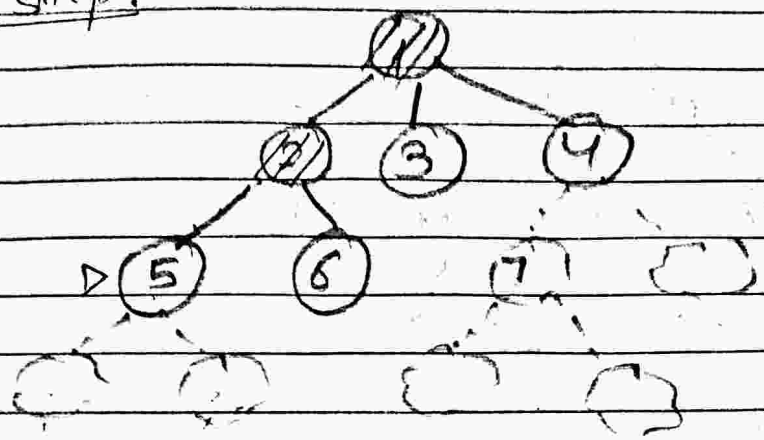
~~O(n+A)~~
~~O(1)~~
O=(1)

2nd step :-



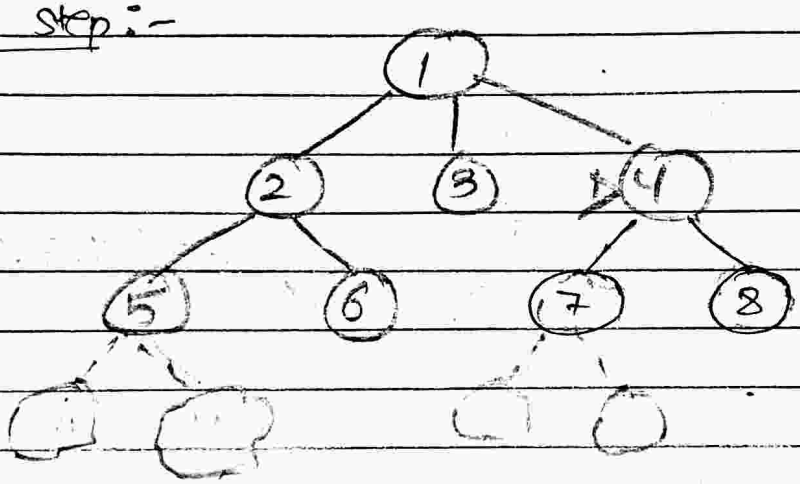
O=(2,3,4)

3rd step :-



O = {3, 4, 5, 6}

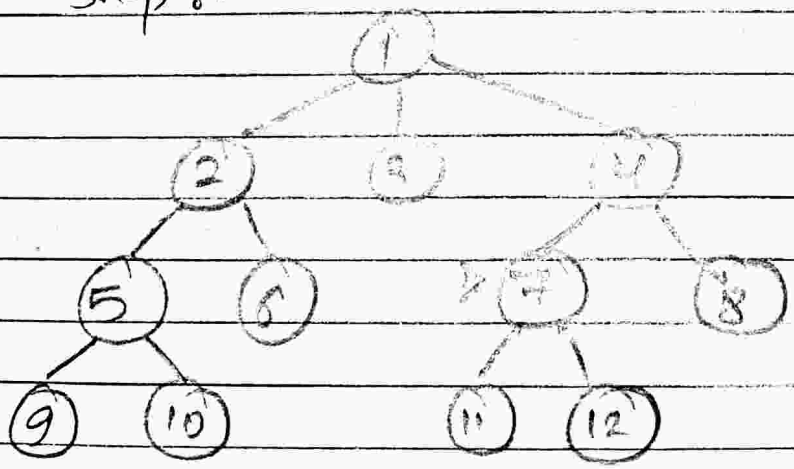
4th step :-



~~O = {6, 9, 10~~

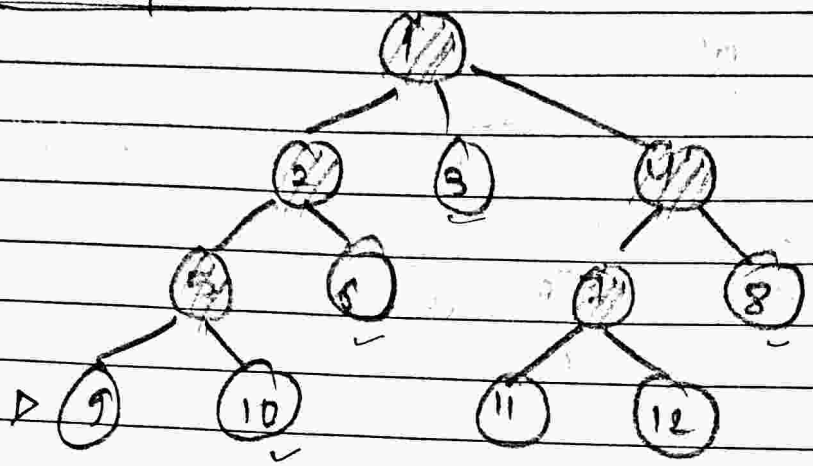
O = {3, 6, 7, 8, 9, 10}

5th step :-



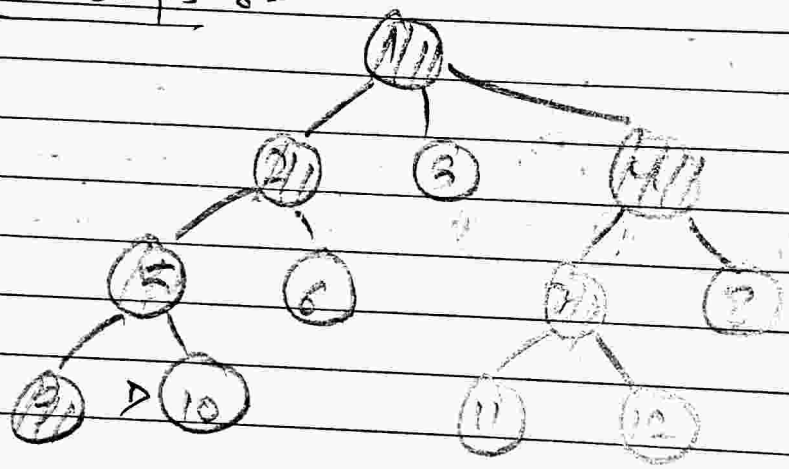
O = {3, 6, 8, 9, 10, 11, 12}

6th Steps :-



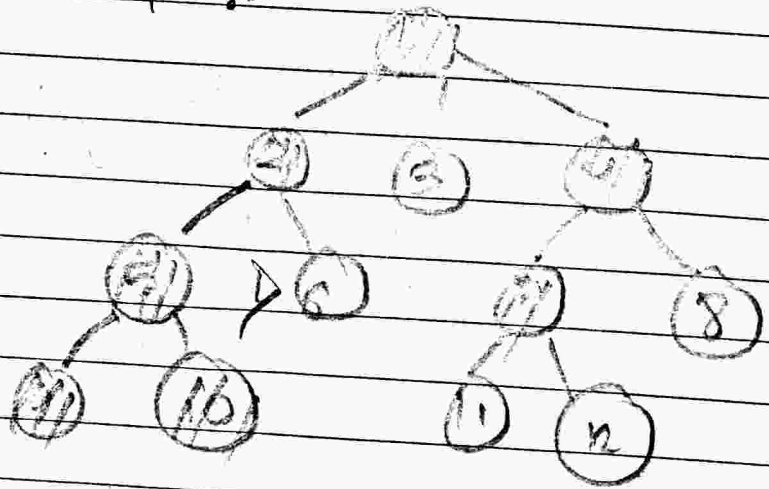
O = {3, 6, 10, 8, 11, 12}

7th Steps :-



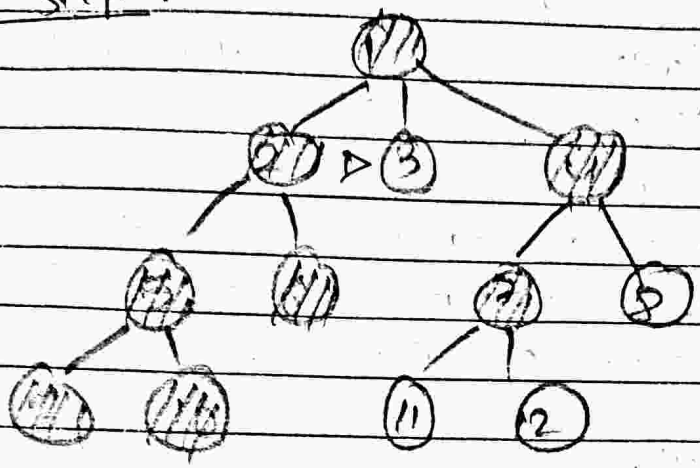
O = {6, 3, 11, 12, 8}

8th Steps :-



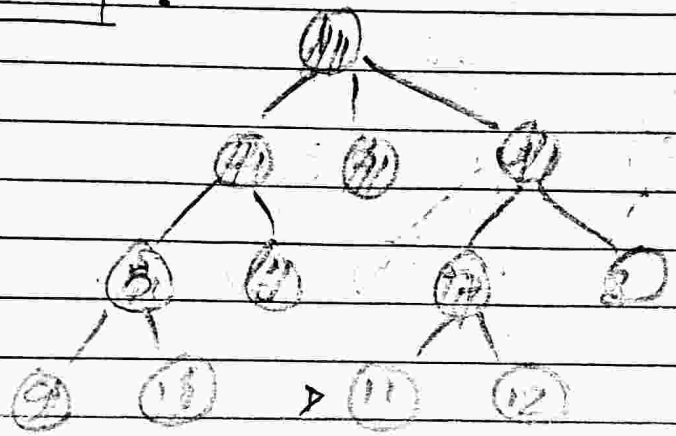
O = {3, 11, 12, 8}

9th step :-



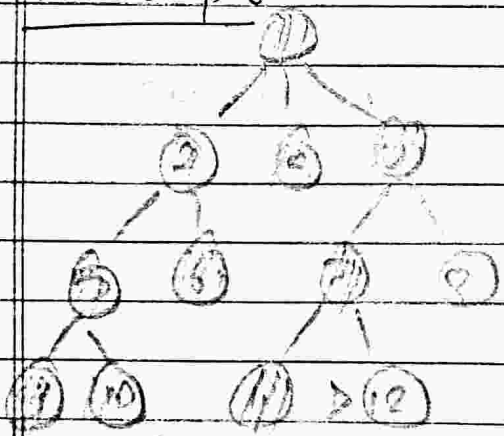
O = { 11, 12, 8 }

10th step :-



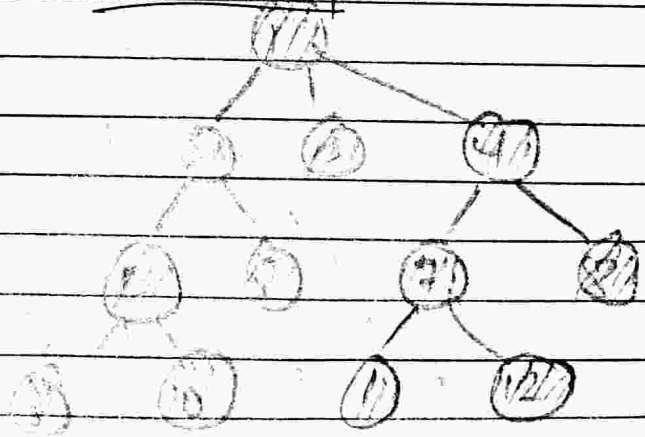
O = { 12, 8 }

11th step :-



O = { 8 }

12th step :-



O = { }

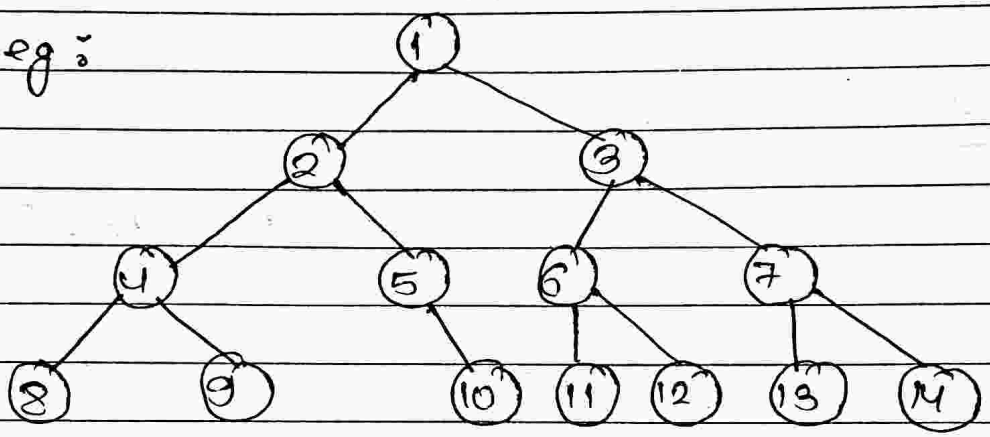
(b)

DFS (Depth First Search)

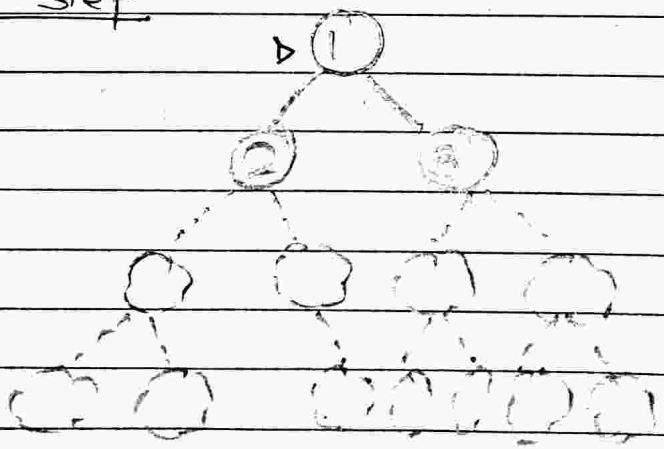
→ It is a recursive algorithm that uses back tracking principle.

Fringe :- The set of all nodes at the end of all visited paths is called fringe, frontier or border.

eg :

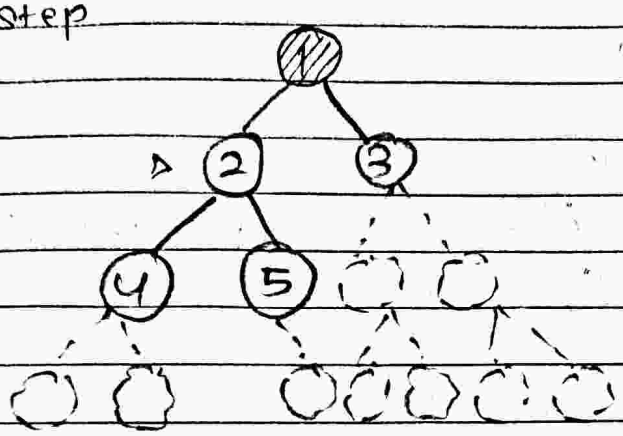


1st step



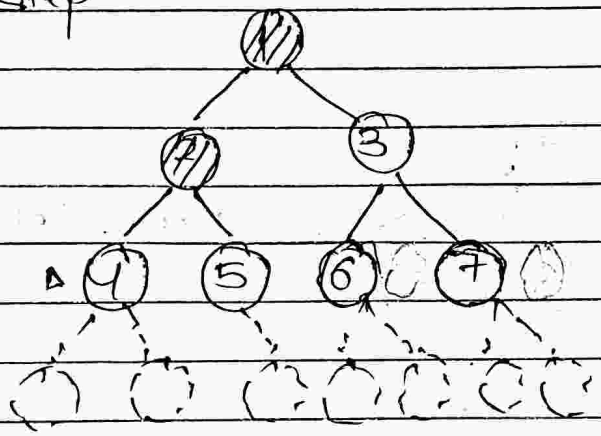
fringe (1)

2nd step



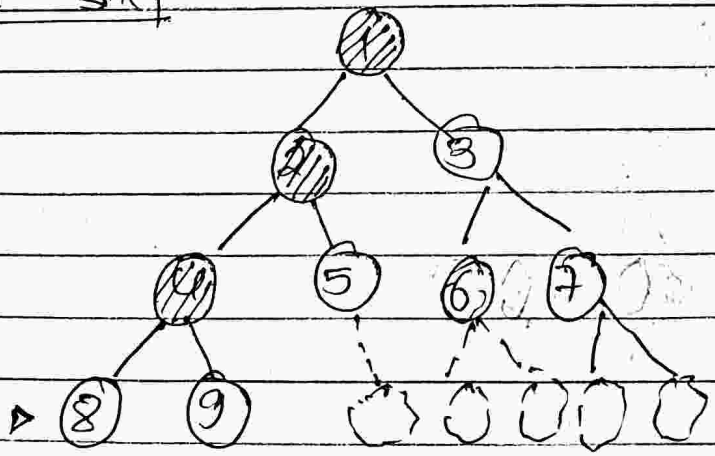
fringe (2,3)

3rd step



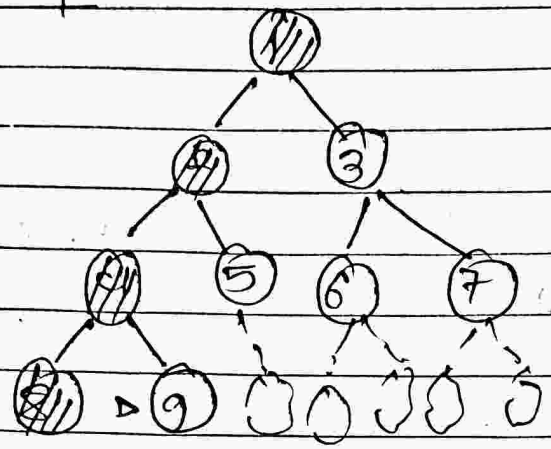
fringe (4,5,3)

4th step



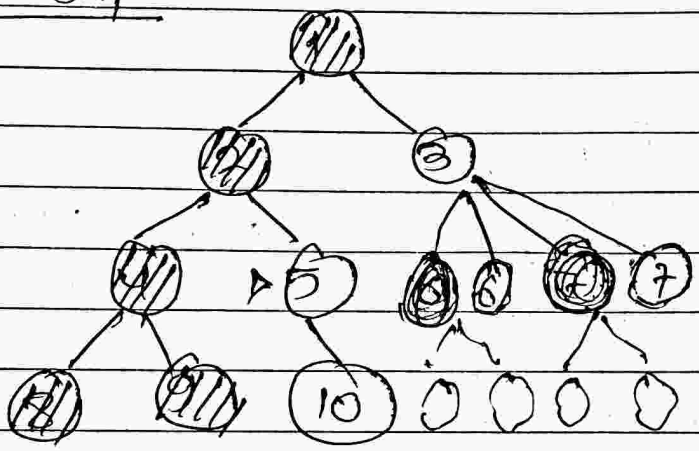
fringe (8,9,5,3)

5th step



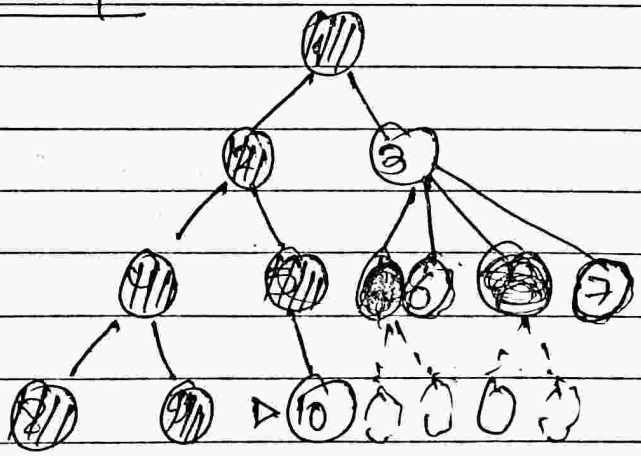
fringe (9, 5, 3)

6th step



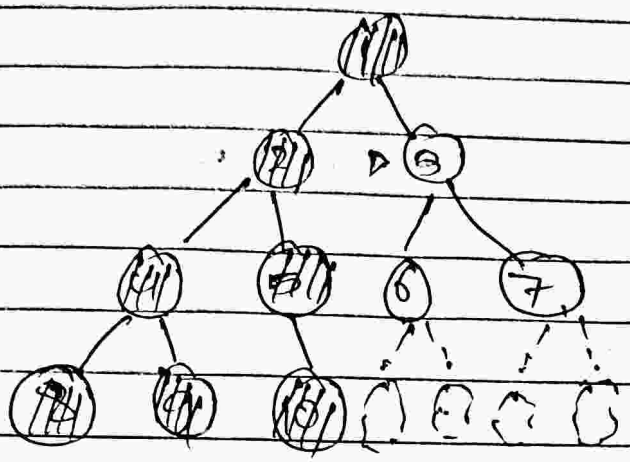
fringe (5, 3)

7th step



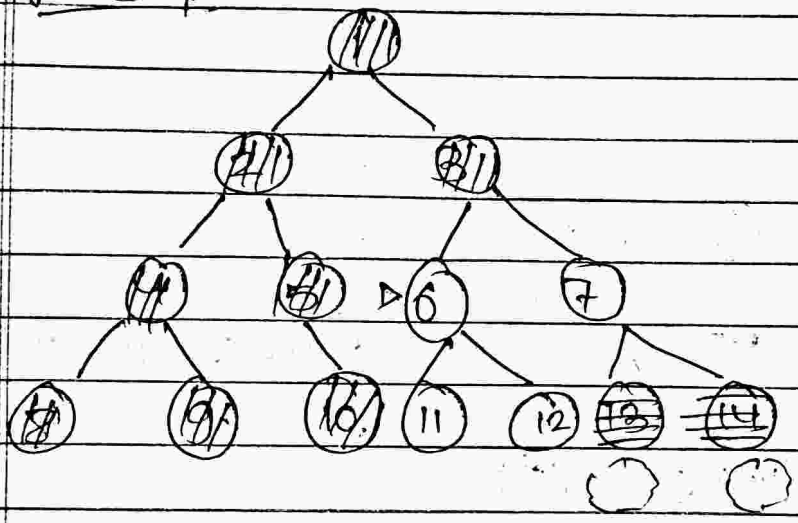
fringe (~~10~~ 10, 3)

8th step



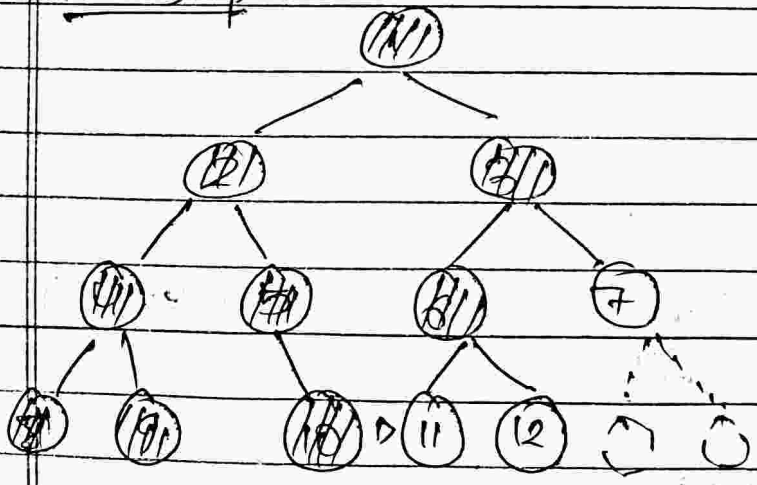
fringe (6, 7)

9th step



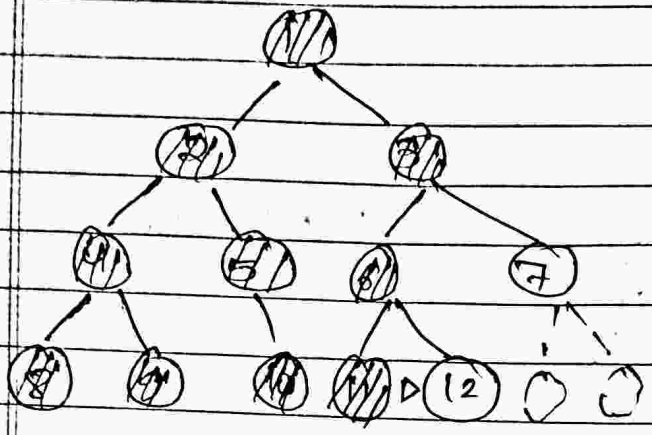
fringe (11, 12, 7)

10th step



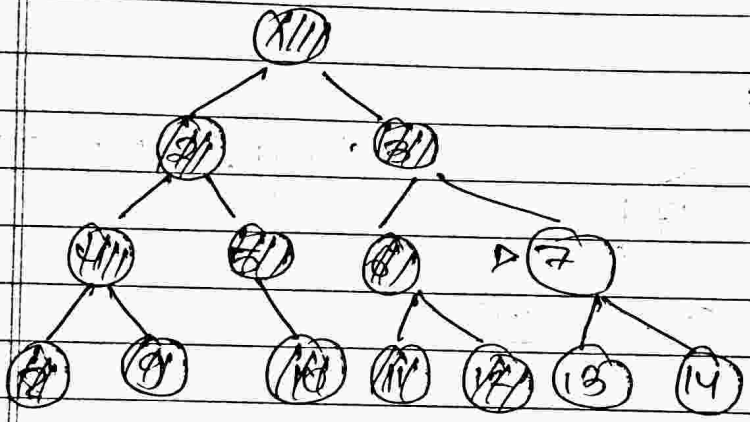
fringe (12, 7)

11th step



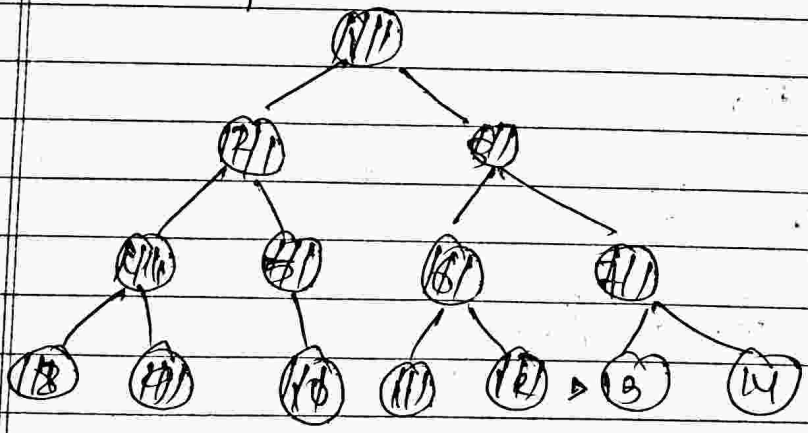
fringe (~~12~~, 7)

12th step



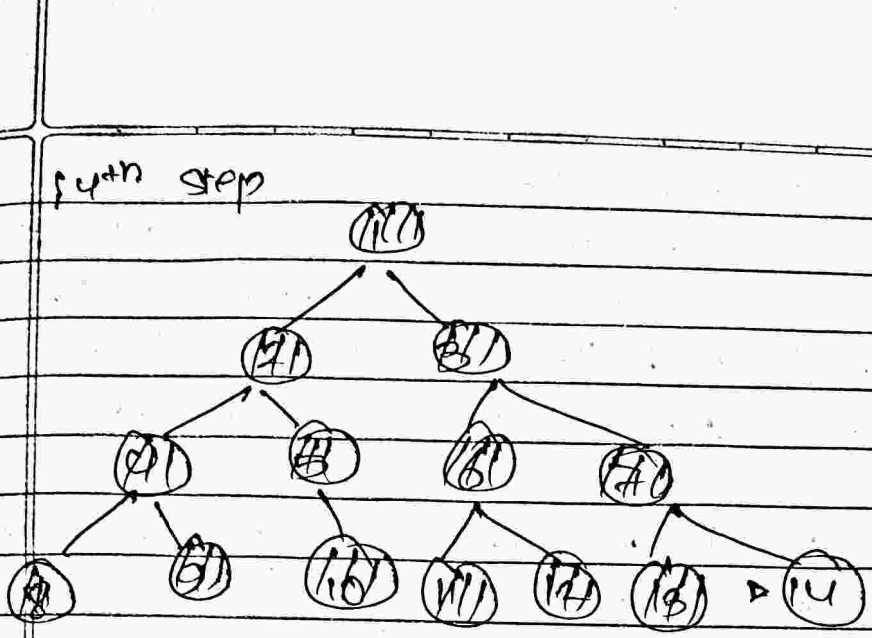
fringe (13, 14)

13th step



fringe (14)

4th step



fringe = ?

DFS Evaluation

→ for a state space with branching factor b and maximum depth m,

Time complexity : $O(b^m)$

Space complexity : $O(b^m + 1)$

→ Thus, DFS requires less memory. Since the nodes on the current path are stored.

→ Completeness : NO

unless search space is finite and no loops are possible.

→ Optimality : NO

(might never find any solutions).

(c) Bi-directional Search :-

In normal graph search using BFS / DFS we begin our search in one direction usually from source vertex toward the goal vertex but what if we start search from both direction simultaneously.

Bi-directional search is a graph search algorithm which find smallest path from source to goal vertex.

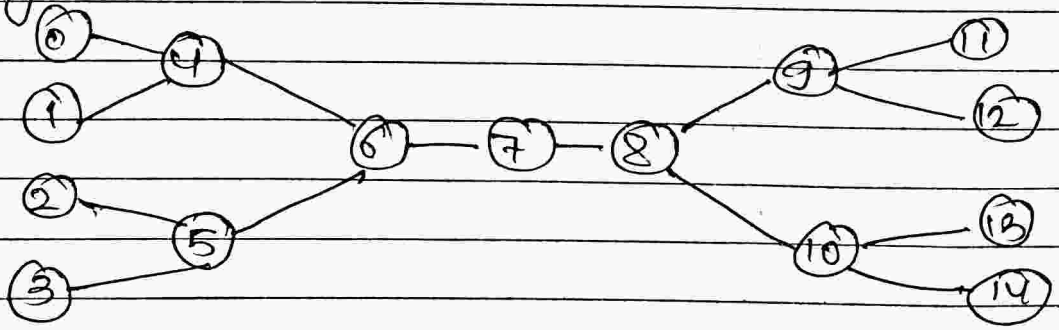
→ It runs two simultaneous each :

1) forward search from source / initial vertex toward goal vertex.

2) Backward search from goal / target vertex toward source vertex.

→ Bidirectional search replaces single search graph with two smaller sub-graphs - one starting from initial vertex and other starting from goal vertex. The search terminates when two graphs intersect.

e.g.-



→ Suppose we want to find if there exists a path from vertex 0 to vertex 14.

Here we can execute two searches, one from vertex 0 & other from vertex 14.

→ When both forward & backward search meet at vertex 7, we know that we have found a path from node 0 and 14 & search can be terminated now.

~~→ Bi-directional Avalia~~

Why Bi-directional Search

→ Because in many cases it is faster, it dramatically reduces the amount of required exploration.

→ Suppose if branching factor of tree is b and the distance of goal vertex from source is d, then the normal BFS / DFS searching complexity would be $O(b^d)$. On the other hand, if we execute two search operation then the complexity would be $O(b^{d/2})$ for each search and total complexity would be $O(b^{d/2} + b^{d/2})$ which is far less than $O(b^d)$.

When to use bidirectional approach?

We can consider bidirectional approach when:

→ Both initial & goal states are unique & completely defined.

→ The branching factor is exactly the same in both directions.

Performance Measures:

* Completeness :- Bidirectional search is complete if BFS is used in both searches.

* Optimality :- It is optimal if BFS is used for search and path have uniform cost.

* Time and space complexity :- Times & space complexity is $O(b^{d/2})$.

(11) Informed Search Techniques :

→ Use problem specific knowledge as much as possible to drive the search.

→ Are almost more efficient than uninformed search and often also optimal.

Main idea

(1) Use of the knowledge of the problem domain to build an evaluation function 'f'.

(2) for every node 'n' in the search space function f(n) quantifies the desirability of expanding 'n' inorder to reach the goal.

(3) Then use the desirability value of the nodes to decides which node to expand next.

Note :- The right choice of nodes isn't always the one suggested by 'f'.

Some Informed Search techniques :- (special Search)

(1) Best first Search
(2) Greedy Search
(3) Heuristic Search
(4) Iterative deepening A* search
(5) Hill climbing.

(1) Best first Search

→ It makes the use of heuristic function to select most ~~eff~~ desirable path to the goal state.

→ This algorithm retains all estimated computed for

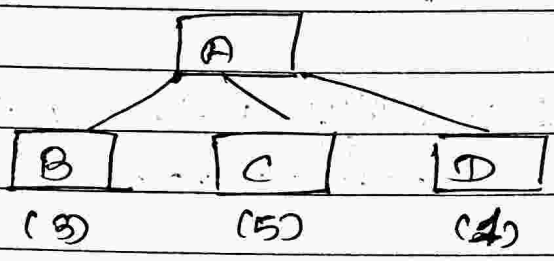
previously generated nodes and makes it selection on the best among them.

eg :-
If node upto 10

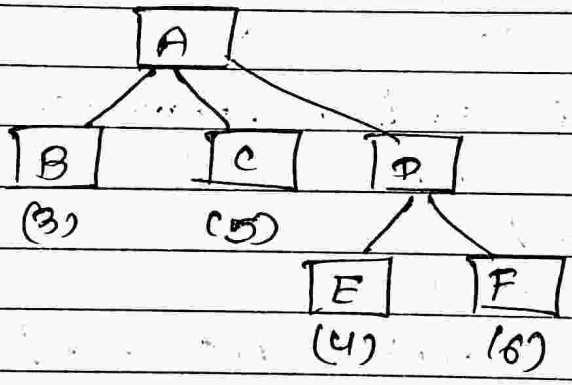
Note :- (Assuming any no. at first & number first have to branch first node upto given in Q.)

Step -1 :

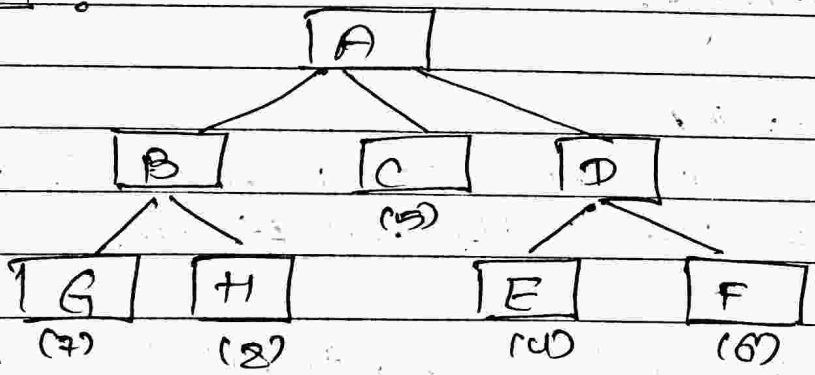
Step -2 :



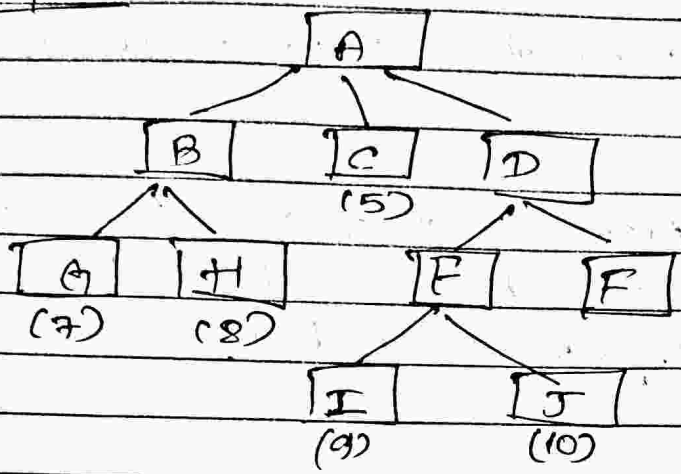
Step-3 :



Step-4 :



Step - 5 :

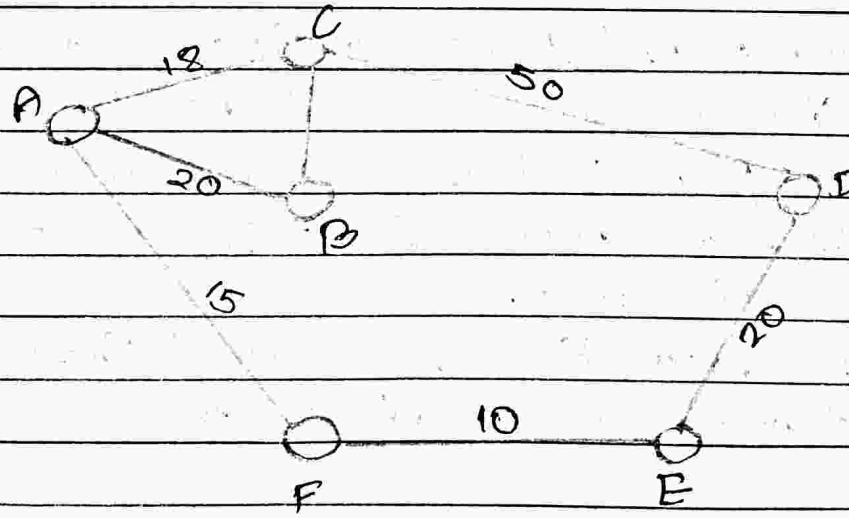


(2) Greedy Search

→ expands the node that is closest to the goal on the grounds that is likely to lead to the solution quickly.

$f(n) = h(n)$ = straight line distance from node n to goal node.

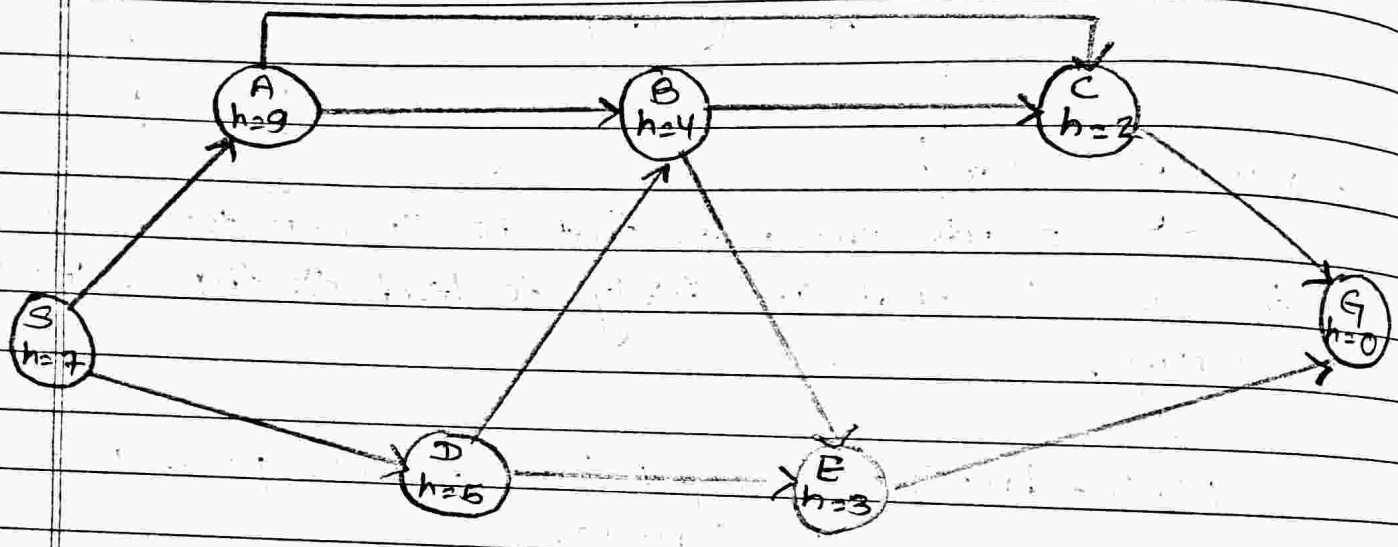
eg-



where, The "closeness" is estimated by a heuristics $h(n)$.

Strategy :- Expand the node closest to the goal state i.e expand the node with a lower h value.

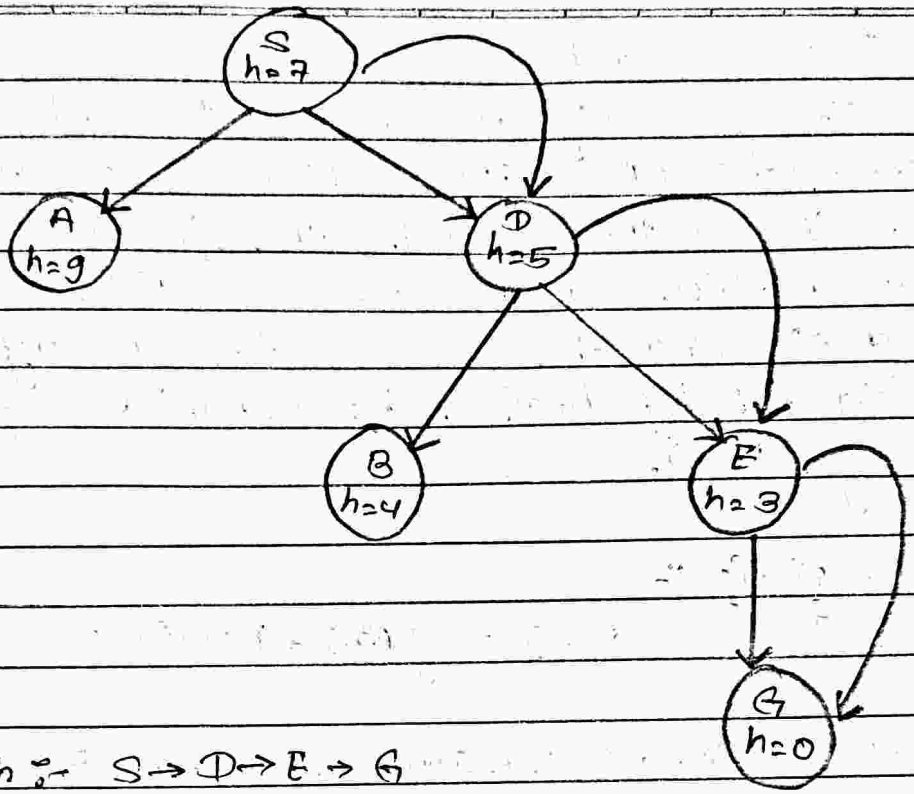
Q. Find the path from S to G using greedy search (The heuristic value h of each node is given below the name of the node).



Solution :-

Starting from 'S', we can traverse to A(h=9) or D(h=5) we choose D, as it has lower heuristic cost. Now from D(h=5), we can move to B(h=4) or E(h=3). We choose E(h=3) as it has a lower heuristic cost.

→ Finally, from E, we go to G(h=0). This entire traverse is shown in the search tree below.



path :- S → D → E → G

Advantage :-

→ Works well with informed search problems, with fewer steps to reach a goal.

Disadvantage :-

→ Can turn into unguided DFS in the worst case.

Q) A* Tree Search

→ It is simply known as A* search, combines the strengths of uniform-cost search and greedy search. In this search, the heuristic is the summation of the cost in UCS (Uniform-cost search) denoted by g(x) and the cost in the greedy search denoted by h(x). The summed cost is denoted by f(x).

Heuristic :-

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

Here, h(x) → forward cost and is an estimate of the distance of the current node from the goal node.

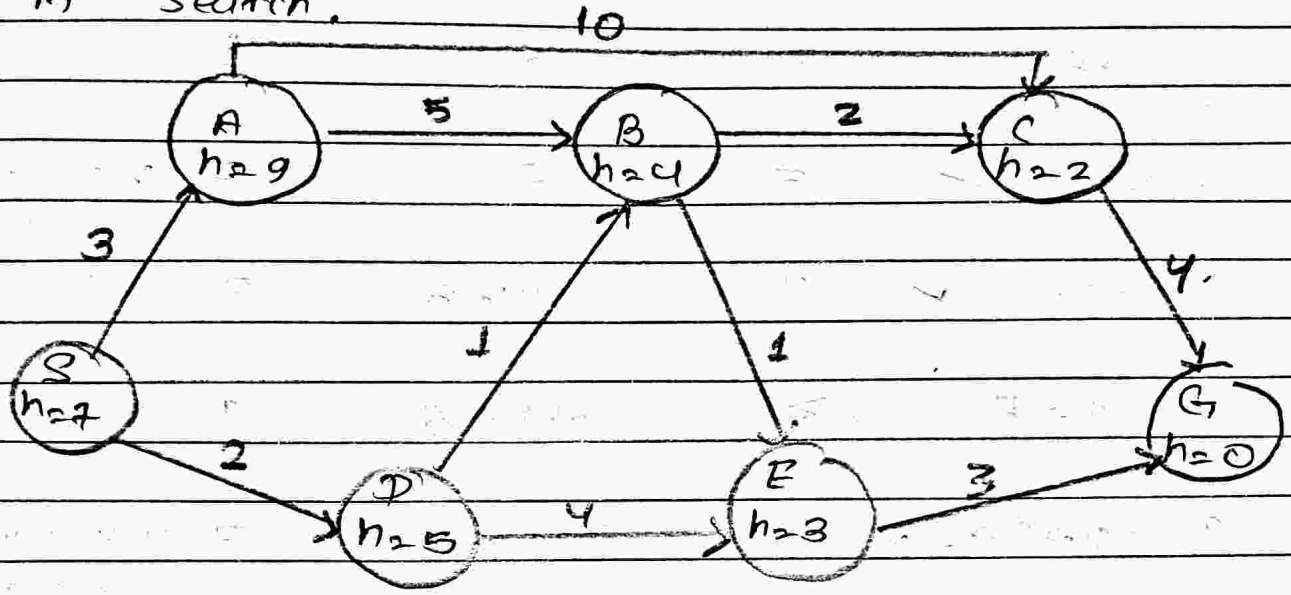
g(x) → backward cost and is the cumulative cost of the node from the root node.

A* search is optimal only when all nodes, the forward cost for a node h(x) underestimates the actual cost $h^*(x)$ to reach the goal. The property of A* heuristic is called admissibility.

$$0 \leq h(x) \leq h^*(x)$$

Strategy : choose the node with the lowest f(n) value

Qn :- Find the path to reach from S to G using A* search.



Note :- The path with a lower cost on further expansion is the chosen path.

Path | h(n) | g(n) | f(n) = h(n) + g(n)

Path	h(n)	g(n)	f(n) = h(n) + g(n)
S	7	0	7
S → A	9	3	12

Solution :-

Starting from 'S', the algorithm computes g(n) + h(n) for all nodes in the fringe at each step, choosing with the lowest sum.

Path	h(x)	g(x)	f(x) = h(x) + g(x)	
S	7	0	7	
S → A	9	3	12	
S → D ✓	5	2	7	∵ (smallest need to expand)
S → D → B ✓	4	2+1 = 3	7	∵ (smallest need to expand)
S → D → E	3	2+4 = 6	9	
S → D → B → C ✓	2	2+1+2 = 5	7	∵ (same so both need to expand)
S → D → B → E ✓	3	2+1+1 = 4	7	
S → D → B → C → G	0	~~2+4~~ 5+4 = 9	9	
S → D → B → E → G ✓	0	4+3 = 7	7 (Smallest) ∴	

~~Path : h(x) g(x) f(x) = h(x) + g(x)~~

Path :-

S → D → B → E → G

Cost :- 7

Note :- In the fourth iterations, we get two paths with equal summed cost f(x), so we expand them both in the next set. The path with a lower cost on

further expansion is the chosen path.

(3) Heuristic Function

→ It is a function used in Informed search, and it finds the most promising path.

→ It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.

→ The heuristic method, however might not always give the best solution, but it guaranteed to find a good solution is reasonable time.

→ Heuristic function estimates how close a state is to the goal.

→ It is represented by h(n), and it calculates the cost between the pair of states.

→ The value of heuristic function is always positive.

→ Admissibility of the heuristic function is given as

$$h(n) \; c = h^*(n)$$

Here, h(n) is heuristic cost and $h^*(n)$ is the estimated cost. Hence, heuristic cost should be

less than or equal to the estimated cost.

Admissibility :
An algorithm is said to be admissable, if it returns an optimal solution.

~~Hill Clim~~

(5) Hill Climbing :-

→ It is a local search algorithm which continuously moves in the direction of increasing elevation / value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value when no neighbour has a higher value.

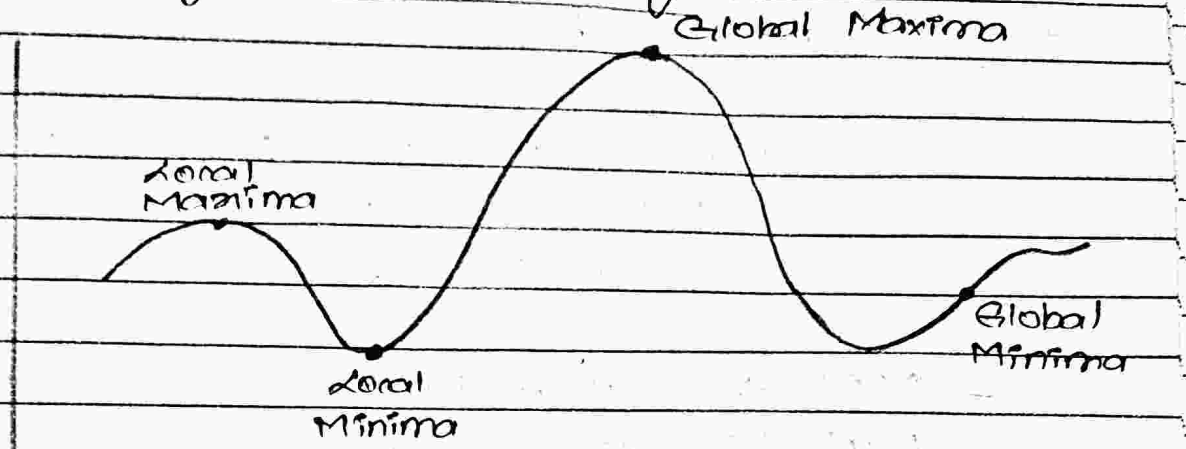
→ It is used for optimizing mathematical problems.

→ eg → TSP (Travelling Salesman Problem) which is used to minimize the distance travelled by the salesman.

→ It is also called greedy local search as it only looks to its good immediate neighbour state & not beyond that.

→ A node of hill climbing algorithm has two components which are state and value.

State Space Diagram - Hill climbing



Game Playing (Adeversarial Search)

→ It is an important domain of artificial intelligence.

→ Game don't require much knowledge; the only knowledge we need to provide the rules, legal moves and the conditions of winning or losing the game.

→ It is used in games where one player's attempt to maximize their fitness (win) is opposed by another player.

→ The two adversarial search techniques:

* minimax - procedure
* Alpha - beta Procedure

~~Draft~~

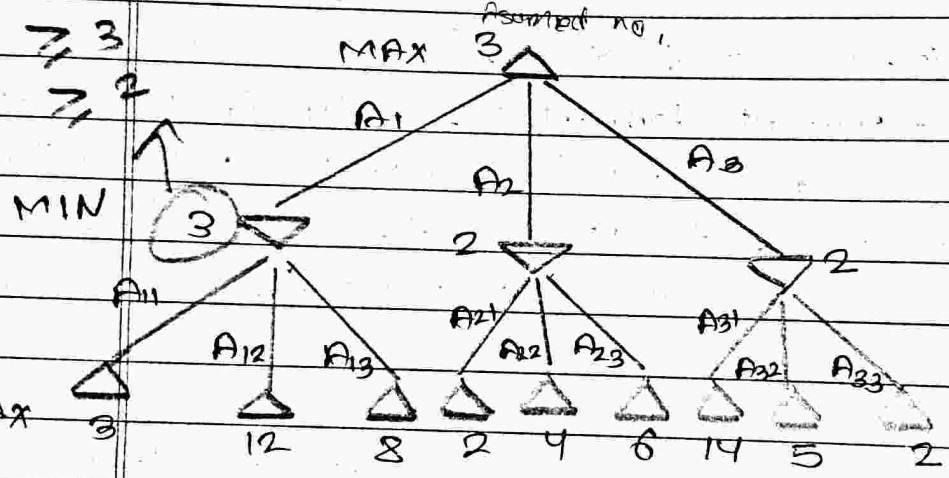
(1)

MiniMAX Procedure

→ In Minimax procedure, players are referred to as
- MAX (the player) and
- MIN (the opponent)

→ MAX tries to maximize its score while MIN tries to minimize the score of MAX.

eg. 2 Play game



→ If M = maximum depth of the tree
b = no. of legal moves at each point

→ Then,

Time complexity = $O(b^m)$

Space complexity = $O(bm)$

(11)

Alpha-beta pruning / procedure : (α-β procedure)

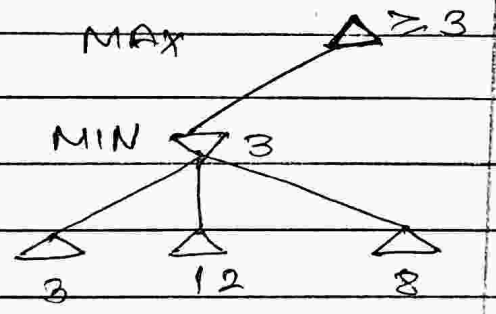
→ We don't always need to evaluate the full tree. If we notice that a particular path is going to be worse, than a previous one, then we don't need to explore & evaluated nodes in that path wasting time.

→ The α-β pruning is a procedure to reduce the amount of computation and searching during minimax by eliminating the nodes of the game tree (which are unnecessary from the consideration).

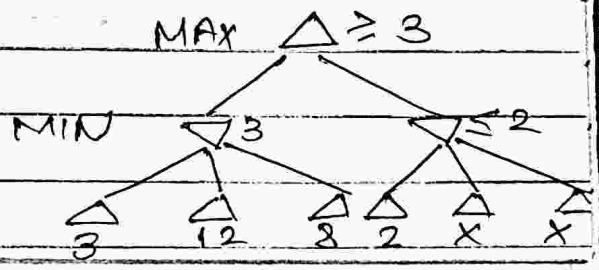
α = value of the best (high value) choice for MAX,
β = value of the best (lowest value) choice for MIN.

eg

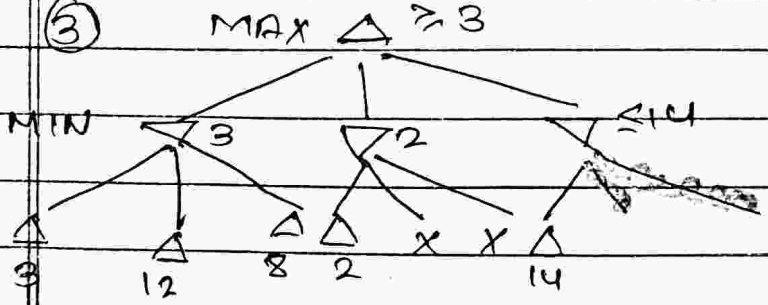
① ~~MAX~~



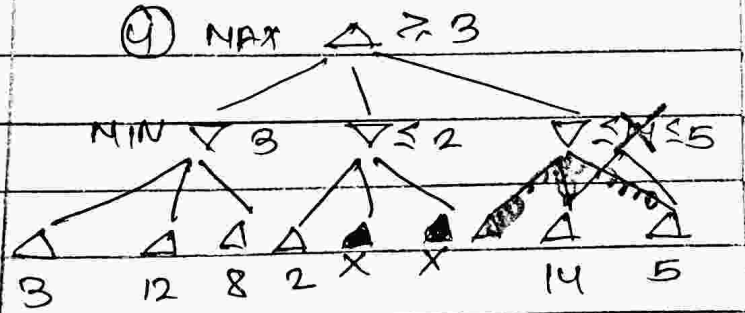
②



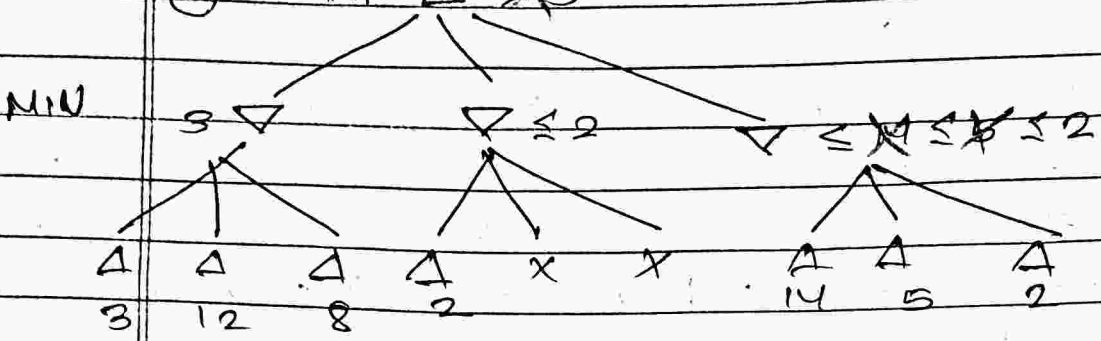
③



④



(5) MAX



Time complexity : $O(b^{m/2})$

Page: Date:

Unit 3
Knowledge and Reasoning

Objective of knowledge Representation :

→ To express knowledge in a computer tractable form such that it can be used by an algorithm to perform any basic.

→ KR is defined by :

① Syntax –

It is the grammar / structure or order of element in a language.

② Semantics –

It is the study of meaning of sentence / semantics.

Formal logic Connectives :

→ A logical connective (operator) is a symbol or word used to connect two / more sentence or statements in a grammatically valid way such that the compound sentence produced has a truth table dependend on the respective truth values of the original sentences

→ Commonly used logical connectives include :-

(a) Negation
- not
- $\neg$ or $\sim$ (symbol)

(b) Conjunction
- and
- $\wedge$

(c) Disjunction
- or
- $\vee$

(d) Conditional
- If ... then
- $\rightarrow$

(e) Biconditional
- iff (if and only if)
- $\leftrightarrow$

p	q	$q \rightarrow p$
T	T	T
T	F	T
F	T	F
F	F	T

eg.
P: It is raining.
q: I am indoors
$q \rightarrow p$ (If q then p) q implies p
If I am Indoors then it is raining.

Q. Draw truth table of prove $(p \vee q) \wedge \neg p$.

p	q	$p \vee q$	$\neg p$	$(p \vee q) \wedge \neg p$
T	T	T	F	F
T	F	T	F	F
F	T	T	T	T
F	F	F	T	F

Q. There are two restaurant suppose that first has a signboard saying that "Good food isn't cheap" and other has a signboard saying that "cheap food is not good". Prove that both case saying same thing.

Soln: Let,

G : Food is good

C : Food is cheap

Then, Good food isn't cheap : $G \rightarrow \neg C$

cheap food isn't good : $C \rightarrow \neg G$

Now we have to prove,

$G \rightarrow \neg C = C \rightarrow \neg G$

G	C	$\neg G$	$\neg C$	$G \rightarrow \neg C$	$C \rightarrow \neg G$
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	T	T

Note: $p \rightarrow q$: $T \rightarrow F = F$

Hence: $G \rightarrow \neg C$ & $C \rightarrow \neg G$ so proved ✓

imp Q Define Propositional logic & Predicate Logic

→ Tautology - all value to be true (P ∨ ¬P)

→ Contradictions → all value to be false. (P ∧ ¬P)

→ Logically Contingent → either not Tautology / contradiction.

Logic :-
Logic is used to supply formal semantics of how reforming functions should be applied to the symbols in knowledge Representation (KR).

imp → Symbols in logic has two methods :-
① Propositional Logic
② Predicate Logic

① Propositional Logic (PL)
→ also known as sential logic / statement logic.
→ deals with statements or propositions and the connection between them.
→ eg : The symbol 'p' could represent the proposition "Sita is mother of Gita".

Syntax of PL
(i) Simple (atomic sentence) eg - P
(ii) Compound Sentence eg → p → q

→ PL uses uppercase letter to represent the statements (variables).

→ PL uses logical constants : TRUE, FALSE

→ logical connective (operators)

$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (Precedence)

→ It uses Parentheses ( )

→ A WFF (Well-Formed Formula) of PL is defined as :

(1) An atomic statement is WFF.

~~(2) If p and q~~

(2) If 'p' and 'q' are WFF, then $p \wedge q$, $p \vee q$, $p \rightarrow q$ & $(\neg p, \neg q, \wedge q)$, $p \leftrightarrow q$ also WFF. (In Upper case)

(3) Any formula formed by applying rules 1 to 2 are WFF.

Semantics

→ In PL, the semantics of formula is just the value TRUE or FALSE.

→ Semantics of a WFF is ~~se~~ specified by the interpretation of the propositions constants and logical connectives.

→ Interpretation
- assigning truth value to each of the atomic sentence.

→ eg - $((P \wedge \neg Q) \rightarrow R) \vee Q)$ —— (1)

→ Suppose p is true and Q and R are false, then (1) gives false.

$$((T \wedge T) \rightarrow F) \vee F$$

$$\text{Or}_1 = (T \rightarrow F) \vee F$$

$$\text{Or}_1 = F \vee F$$

$$\text{Or}_1 = F$$

Properties of Wff

① Valid / tautology :-

② Satisfiable :

$\forall a \; N(a)$ where, $N(a)$ means a is non-negative is satisfiable, for if the universe is the set of natural numbers, the assertion to wff to be true.

③ Falsifiable :

Hypothesis to be proven wrong.

④ Contingent :

If both satisfiable and ~~for~~ falsifiable is known as contingent.

Chain rule :

$$(A \rightarrow B) \wedge (B \rightarrow C) = A \rightarrow C$$

* Condition

$$A \rightarrow B = \neg A \vee B$$

$$A \rightarrow B = \neg B \vee \neg A$$

$$A \leftrightarrow B = (A \leftrightarrow B) \wedge (B \rightarrow A)$$

Duality law

(1) $A \to B = \neg A \vee B$

→ Two formula A & A^*

→ A and A^* are said to be dual of each other if either one can be obtained from other by replac

$A \to B = \neg A \vee B$

(1)

A	B	$A \to B$	$\neg A$	$\neg A \vee B$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

proved

(2) $A \to B = \neg B \to \neg A$

A	B	$A \to B$	$\neg B$	$\neg A$	$\neg B \to \neg A$
T	T	T	F	F	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Not proved

(3) $\neg(A \Leftrightarrow B) \wedge (B \to A)$

A	B	$A \Leftrightarrow B$	$B \to A$	$(A \Leftrightarrow B) \wedge (B \to A)$
T	T	T	T	T
T	F	F	T	F
F	T	F	F	F
F	F	T	T	T

Not proved

$(A \to B) \wedge (B \to C) \equiv A \to C$

A	B	C	$A \to B$	$B \to C$	$(A \to B) \wedge (B \to C)$
T	T	T	T	T	T
T	T	F	T	F	F
T	F	T	F	T	F
T	F	F	F	T	F
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	T	T	T
F	F	F	T	T	T

$A \to C$

T
F
T
F
T
T
T
T

Some Question

Page : Date :

(2) Predicate logic

→ as first order predicate logic (FOPL) / first order predicate calculus

→ assumes that the world consist of objects with individual properties.

→ FOPL includes a richer ~~ontology~~ ontology to analyse each sentence.

→ Objects :-
  → terms (name of the objects)
  → eg - RAM, car

→ Properties :-
  → unary predicate on terms
  eg :- nary prediction on terms
  - Brother of, greater than

→ function :-
  - mapping from terms to other terms
  eg - Plus (1, 2)

Syntax :-

(a) Sentence :- Atomic Sentence
  - Sentence Connective Sentence
  - Quantifier Variable Sentence
  - ¬ (NOT) Sentence

- (Sentence)

b) Atomic Sentence

Predicate (term, term, ...)

c) Term → function (term, term, ...)
- Constant - Variable

d) Connectives :

$\neg, \wedge, \vee, \rightarrow, \leftarrow$

e) Quantifiers :

$\exists$ → Existential quantification,
$\forall$ → Universal quantification

f) Variables

g) Predicate

→ is a verb phrase template that describes property or relationship among objects

→ eg - The sky is blue

Predicate →

Is blue (sky) or blue (sky)

Page: Date:

* function
  eg - father-of , plus ( )

* Universal quantifier ∀

→ let a statement ∀x P(x)
eg - All car has wheels
∀x p(x) → ∀cars has_wheels (cars)
where, p → has_wheels
x → cars

* Existential quantification : ∃

- let a statement ∃x P(x)
eg - Someone loves you.
∃someone Loves_you (someone)

~~eg -~~

* Nesting of quantifiers

1) ∀x ∃y (loves (x,y))

⇒ Everyone loves Someone

2) ∃y ∀x (loves (x,y))

⇒ ~~Someone loves~~ Everyone
→ Everyone has someone who loves them.

3) $\exists x \forall y$ (loves (x, y))

⇒ There is Someone who loves everyone.

* Logical Equivalent :-

1) $\forall x \neg P, \Leftrightarrow \neg \exists x P$

2) $\neg \forall x P \Leftrightarrow \exists x \neg P$

3) $\forall x P \Leftrightarrow \neg \exists x \neg P$

4) $\exists x P \Leftrightarrow \neg \forall x \neg P$

5) $\forall x (P(x) \wedge Q(x)) \Leftrightarrow \forall x P(x) \wedge \forall x Q(x)$

6) $\exists x (P(x) \vee Q(x)) \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$

Solutions :-

* Representing facts in predicate logic

eg → ① Marcus was a Man
→ Man(Marcus)

② Marcus was a Pompeian
→ Pompeian(Marcus)

③ All Pompeians were Romans
→ $\forall x$: Pompeians(x) ⟶ Romans(x)

④ Caesar was a ruler
→ Ruler(Caesar)

⑤ All romans were either ~~beg~~ loyal to caesar or hated him.
→ $\forall x$: Romans(x) ⟶ loyal_to(Caesar) ∨ hate(x, Caesar)

⑥ Everyone loyal to someone.
→ $\forall x \exists y$ (loyal(x, y))
OR → $\forall x \exists y$: loyal_to(x, y)
OR → $\forall x$! $\exists y$: (loyal_to(x, y))

⑦ People only try to assosinate ruler they arent loyal to

→ $\forall x \exists y$: People(x) ∧ ruler(y) ∧ trytoassosinate(x, y) ⟶ ¬ loyal_to(x, y)

(8) Marcus tried to assasinate Caesar.
→ tryassasinate (Marcus, Caesar)

(9) All mens are people.
→ ∀x : Man(x) → People(x)

V.imp End 1st Assesment

Prove ¬loyalto(Marcus, Caesar) using above 9 rules :

sol^n a → Marcus
g → Caesar

* ~~People any~~

Q1. People(Marcus) ∧ ruler(Caesar) ∧ tryassasinate(Marcus, Caesar)
→ ¬loyal_to(Marcus, Caesor)

sol^n Here, People(Marcus)
↓ (8)
People(Marcus) ∧ tryassasinate(Marcus, Caesar)
↓ (4)
People(Marcus) ∧ ruler(Caesar) ∧ tryassisnate(Marcus, Caesar)

↓ (Substitution, 7)

¬ loyalto(Marcus, Caesar)

Hence, proved

Rules of Inference :

→ States that

given a particular formula which a certain property as a hypothesis, another specific formula can be derived as a conclusion.

→ One common rule of inference is

→ rule of Substitution

Let us consider a set of facts about Marcus

① Man(Marcus)

② Pompeian(Marcus)

③ born(Marcus, 40)

④ $\forall x$: Man(x) → Mortal(x)

⑤ $\forall x$: Pompeian(x) → died(x, 79)

⑥ erupted(volcano, 79)

⑦ $\forall x : \forall t_1 : \forall t_2$ Mortal(x) ∧ born(x, t_1) ∧ gt(t_2, t_1) ⟶ dead(x, t_2)

gt : greater than

⑧ now = 1991

⑨ ~~$\forall x : \forall t$: [alive(x, t)] → ¬ dead(x, t) ∧ (¬ dead(x, t) ⟶ alive(x, t))~~

(9) $\forall x : \forall t : [alive(x,t)] \rightarrow \neg dead(x,t)$

(10) $\forall x : \forall t_1 : \forall t_2 : died(x,t_1) \wedge gt(t_2,t_1) \rightarrow dead(x,t_2)$

Q. ⇒ Now answer the question - Is Marcus alive now? by proving $\neg alive(Marcus, now)$

Soln:

(1) Marcus was a man.

(2) Marcus was a Pompeian.

(3) ~~$\forall x : \forall t : [alive(x,t)] \rightarrow \neg dead(x,t) \wedge (\neg dead(x,t) \rightarrow alive(x,t))$~~

Soln:

$n = 1$

$\downarrow$

$gt(1991, 79)$

$\downarrow$ (substitution, 8)

$gt(now, 79)$

$\downarrow$ (2)

$Pompeian(Marcus) \wedge gt(now, 79)$

$\downarrow$ (5, substitution)

$x = Marcus$

$died(Marcus, t_1) \wedge gt(now, t_1)$ where, $t_1 = 79$, $t_2 = now$ time (1991)

$\downarrow$ (10, substitution)

$dead(Marcus, now)$

$\downarrow$ (9, substitution)

$\neg alive(Marcus, now)$ $x \rightarrow Marcus$.

Resolution

→ produces proof by refutation
(to prove a statement, resolution attempt to show that the negation of the statement produces a contradiction with the known statement)

→ resolution requires sentences to be in Conjuctive Normal form (CNF)

Conversion of clause form (CNF)

① Eliminate '→' (implications)
using the fact that
$a \rightarrow b \iff \neg a \vee b$.

② Reduce the scope of each ¬ to a single term using the fact.
$\neg(\neg P) = P$
$\neg(a \wedge b) = \neg a \vee \neg b$
$\neg \forall x : P(x) = \exists x : \neg P(x)$

③ Standardize variable so that quantifier binds a unique variable.
eg - The formula
$\forall x : P(x) \vee \forall x : Q(x)$
could be converted to
$\forall x : P(x) \vee \forall y : Q(y)$

(4) Move all quantifiers to the left of the formula without changing their relative order.

(5) Eliminate existential quantifiers by substituting a variable by a function that produces the desired value.

eg - (i) The formula: $\exists y : P(y)$ can be converted to $P(S_1)$ where, S_1 is a fn with no arguments.

(ii) The formula:

$\forall x \exists y :$ father_of (y, x)

can be converted to

$\forall x :$ father_of $(S_2(x), x)$

where, S_2 is a function with arguments

(6) Since, all this point, all the variables are universally quantified, now just drop the universal quantifiers.

(7) Convert the formula into conjunction of disjuncts

(8) Create a separate clause corresponding to each conjunct.

(9) Standardize apart the variable in the set of clauses generated at step 8.

eg - Convert :

$\forall x [\forall y \text{ animal}(y) \rightarrow \text{loves}(x,y)] \rightarrow (\exists y \text{ loves}(y,x)]$ into CNF (Conjunctive Normal form)