(A.I.)

Intelligent :→

(1) Who has sufficient knowledge. &
(2) Has Ability of reasoning using his knowledge

Diff. b/w Data & knowledge :-

Data	Knowledge
(1) Data is stored in the form of tables	(1) Knowledge is stored in many forms
(2) In database we just search the data exactly	(2) In knowledge representation some methods are provided for reasoning

Knowledge Representation :→

1. Proposition logic :→
(zero order logic)

1. If it rains then road will be wet.
2. It is raining.

Here It raining : R } These are
road is wet : W } propositions

Propositions represent sentences.

Knowledge { 1. $R \rightarrow W$ → means If R then W
2. R

Make propositional logic :→

1. If it is hot and plant is open then plant will be dead.
2. It is hot
3. Plant is open.

Ans (i) It is hot : H

(ii) Plant is open : P

(iii) Plant will be dead : D

1. $H \wedge P \rightarrow D$

Means If H & P then D

2. H
3. P

Definition :→ Propositional logic formula :→

1. T & F are Propositional logic Formula (PLF)
2. A, B, C etc are PLF where A, B, C are propositions and each proposition represents sentence / statements.
3. Item discussed in point 1 & 2 combine with following operators will

also be a propositional logic Formulae

Operators →

1. ¬ – Negation (Unary operator, only 1 op^r required) — ¬A
2. ∧ – Conjunction / AND
3. ∨ – Disjunction / OR
4. → – Implies
5. ↔ – Equivalence / Equal to

(2, 3, 4, 5: Binary operators)

precedence

Precedence decreases downward.

Mathematical Laws :→

Laws of propositional & predicate logic :→

1. Commutative laws

$A \vee B = B \vee A$

$A \wedge B = B \wedge A$

$A \leftrightarrow B = B \leftrightarrow A$

$A \rightarrow B \neq B \rightarrow A$ (Implies is not commutative)

2. Associative laws :→

$A \vee (B \vee C) = (A \vee B) \vee C$

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

3. Distributive Laws :→

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

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

4. Demorgan's Laws :→

$\neg (A \wedge B) = \neg A \vee \neg B$

$\neg (A \vee B) = \neg A \wedge \neg B$

5. Law of Negation :→

$\neg (\neg A) = A$

6. Law of Excluded middle :→

$A \vee \neg A = T$ (True)

{Because $T \vee F = F \vee T = T$ This is tautology}

7. Law of contradiction →

$A \wedge \neg A = F$

This is Contradiction.

8. Law of implication :→

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

7. Law of equality :-

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

10. Laws of OR - simplification

$A \vee A = A$
$A \vee T = T$
$A \vee F = A$
$A \vee (A \wedge B) = A$

11. Laws of AND simplification

$A \wedge A = A$
$A \wedge T = A$
$A \wedge F = F$
$A \wedge (A \vee B) = A$

Q. Simplify the following logical exp^n,

1. $\neg(b \wedge \neg b)$
2. $(b \wedge (b \rightarrow c)) \rightarrow c$

A. 1. $\neg(b \wedge \neg b)$ By Demorgan's law

or $\neg b \vee b = T$

$\Rightarrow \neg(F)$

$\Rightarrow T$

This statement is tautology because every interpretation of 1 will give true

2. $(b \wedge (b \rightarrow c)) \rightarrow c$

$(b \wedge (\neg b \vee c)) \rightarrow c$

$((b \wedge \neg b) \vee (b \wedge c)) \rightarrow c$

$(F \vee (b \wedge c)) \rightarrow c$

$(b \wedge c) \rightarrow c$

$\neg(b \wedge c) \vee c$

$(\neg b \vee \neg c) \vee c$

$\Rightarrow \neg b \vee (\neg c \vee c) \rightarrow c$

$\Rightarrow \neg b \vee T$

$\Rightarrow T$ (Tautology)

Q. Prove that

$(\neg(b \rightarrow c) \wedge \neg(\neg b \rightarrow (c \vee d))) \rightarrow (\neg c \rightarrow d)$

A. $(\neg(\neg b \vee c) \wedge \neg(\neg b \rightarrow (c \vee d))) \rightarrow (\neg c \rightarrow d)$

$(\neg(\neg b \vee c) \wedge \neg(b \vee (c \vee d))) \rightarrow (c \vee d)$

$(b \wedge \neg c) \wedge (\neg b \wedge \neg(c \vee d))) \rightarrow (c \vee d)$

$(b \wedge \neg c) \wedge (\neg b \wedge (\neg c \wedge \neg d))) \rightarrow (c \vee d)$

$(b \wedge \neg(\wedge \neg b \wedge \neg(\wedge \neg d)) \rightarrow (c \vee d)$

$F \rightarrow (c \vee d)$

$\neg(F) \vee (c \vee d)$

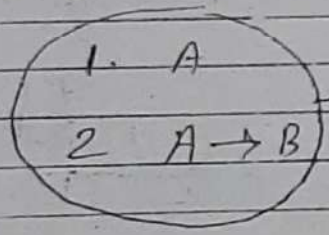
$T \vee (c \vee d)$

T (Tautology)

Reasoning Techniques (Inferencing Rules):-

1. Modes Ponen

Ex



2. Principle of resolution :→ This Inferencing technique is applicable if the given formulas are in clause form.

Clause form → A formula is said to be in clause form if only 2 operators are used ie. 'v' and '¬'. ¬ must be associated with individual proposition (not in group)

Ex 1. $A \vee \neg B$ is clause

2. ~~not~~ $\neg(A \vee B)$ is not clause then we break it. But $\neg A \wedge \neg B$

$\Rightarrow \neg A$ } & Then
$\Rightarrow \neg B$ } These are clause.

3. $\neg(A \wedge B)$

Pf:

Ex 1. $\neg A \vee B$
2. A
} 3. B

Ex 2

1. $A \vee B$
2. $\neg A$
} → 3. B

Ex 3

1. $A \vee B$
2. $\neg A \vee C$
} 3. $B \vee C$

3. Chain Rule :→

Ex 1. $A \rightarrow B$
2. $B \rightarrow C$
} 3. $A \rightarrow C$

1. All SATI Students are intelligent

2. Vijay is SATI Student

Query → Is Vijay intelligent?

1. S → Q → Not possible
2. V

Query → Q

We can't do it by propositional logic.
Because propositional logic is very limited

Predicate Logic (1st order logic):→

Predicate logic includes all the things which are the part of proposinal logic formula.
In predicate logic there are extra facilities over propositional logic for knowledge representation. there are :

i) Predicate can also be ~~any~~ operands

where predicate represents English sentences very similar to propositions but the structure of predicate is similar to the functions used in high level languages.

ex.

Greater (x, y) : x is greater than y

FATHER (x, y) : x is father of y

FATHER (x) : x is father

Predicate will always be true or false.

(ii) Functions can also be used in predicate logic formula.
The structure of func^n is very similar to structure of predicate but function returns non-boolean values.

Each func^n represents english sentence.

ex. Age (x) : age of 'x'

↑ function

Salary(x) : Salary of 'x'

father(x) : Father of 'x'

x+y is greater than 5 by predicat L.F.

GREATER (add(x,y), 5)

add(x,y) :→ x+y

(iii) In predicate logic formula quantifiers can also be used.

There are 2 types of quantifiers

a. Universal Quantifier :→ ∀: (For all)

b. Existential Quantifier

→ ∃ : there exist

→ ∃ UITSTUD(x)

: Some are UITstudents

Convert the following sentences into predicate L.F.

(i) If x is on top of y, y supports x.

(ii) If x is above y and they are touching each other, x is on top of y

(iii) A cup is above a book.

(iv) A cup is touching a book.

By resolution prove that book supports cup.

Following predicates are used

1. TOP(x,y) : x is on top of

2. SUPPORT(x,y) : x supports y

3. ABOVE(x,y) : x is above

4. TOUCH(x,y) : x & y are touch each other

(i) TOP(x,y) ⟶ SUPPORT(y,x)

(ii) ABOVE(x,y) ∧ TOUCH(x,y) ⟶ TOP(x

(iii) ABOVE(cup, book)

(iv) TOUCH(cup, book)

By Resolution :→ We have to prove book supports cup

or SUPPORT(book, cup)

For resolution we will have to make the predicates in clause form

Clauses:→

(i) ¬ TOP(x,y) ∨ SUPPORT(y,z)

(ii) ¬ (ABOVE(x,y) ∧ TOUCH(x,y)) ∨ TOP(x,y)

~~¬ (a)~~ (¬ ABOVE(x,y) ∨ ¬TOUCH(x,y)) ∨ TOP(x,y)

(iii) ABOVE (Cup, book)

(iv) TOUCH (Cup, book)

Apply resolution b/w clause (ii) & (iii) with x = cup & y = book

¬TOUCH(cup,book) ∨ TOP(cup,book) → (v)

Apply resolution b/w clause (iv) & (v)

(vi) TOP (cup, book)

Apply resolution b/w (i) & (vi) with x = cup & y = book

(vii) SUPPORT (book, cup)

Q1. Represent following sentences in predicate logic formula.

(i) All employees having salary more than 1 lac Rs. per annum supposed to pay income tax.

(ii) Some employees are sick today.

(iii) No employee earns more than the president.

∀x ∀y Employee(x) ∧ President(y) → ¬ greater(sal(x), sal(y))

Predicates :→

~~SALARY(x) Employee have salary~~

1. SALARY

(i) ∀x EMPLOYEE(x) ∧ GREATER(Sal(x), 1L) → IT(x)

Following predicates are used:→

EMPLOYEE(x): x is employee

GREATER(x,y): x is greater than y

IT(x) : x pays Income Tax

Following functions are used:→

~~Salary(x)~~ Sal(x): Salary of x

2.(ii) $\exists x$ EMPLOYEE(x) $\wedge$ SICK(x)

Predicates →
SICK(x): x is sick

Q2 Make predicate 2. F.
1. All SATI Students are intelligent.
2. Amit is a SATI Student.

A.
1. S (All students of SATI are intelligent)
2. A (Amit is SATI student)

Following predicates are used: →

INTEL(x): x is intelligent
SATISTU(x): x is SATI student

1. $\forall$ x IF SATISTU(x) → INTEL(x)
Here this is a rule so we use → sign.
2. IF SATISTU(Amit).

Q1(iii) NO employee earn more than the president.

A Predicates: →
EMP
Employee(x): x is employee
Function: → Earning(x): Earning of x
PRES(y): y is the president
MORE(x,y): x is more than y

$\forall x \forall y$ EMP(x) $\wedge$ PRESIDENT(y) →
¬(More(Earning(x), Earning(y))

Q3
1. All the persons of age more than 60 years are senior citizens

A Predicates: →

PERSON(x): x is person
SCIT(x): x is senior citizen
MORE(x,y): x is more than y

Functions: →
age(x): age of x

$\forall x$ PERSON(x) $\wedge$ MORE(age(x), 60) → SCIT(x)

Q4 Students get 60% or more in 12th and 35% marks in PET are illegible for engg. admision

A
Predicates:
STU(x): x is student
MARK(x): M
ELIGIBLE: x is eligible for engg. admision
MORE(x,y): x is more than y

Functions :→

Marks^{-12}(x) : marks of x in 12th

Marks-PET(x) : marks of x in PET

$\forall x$ STU(x) $\wedge$ MORE(Marks-12(x), 60%)
$\wedge$ MORE(Marks-PET(x), 35%)
$\longrightarrow$ ELIGIBLE(x)

Conversion of predicate logic formula into Prenex Normal Form (PNF) :→

$(Q_1 x)(Q_2 x) \cdots (Q_n x)\, M$

$Q_1, Q_2, \cdots Q_n$ are the quantifiers.
M - It is the formula (P.L.F) that doesn't contain any quantifier.

Rules required to convert predicate logic formula into PNF :→

1. $(Qx)\, F(x) \vee G = (Qx)[F(x) \vee G]$

F(x) & G are the predicates and propositions respectively
Q - Quantifier

Ex $F(y) \vee (Qx)\, G(x)$
then $\Rightarrow (Qx)[F(y) \vee G(x)]$

2. $(Qx)\, F(x) \wedge G = (Qx)[F(x) \wedge G]$
3. $\neg(\forall x)\, P(x) = (\exists x) \neg P(x)$
4. $\neg(\exists x)\, P(x) = (\forall x) \neg P(x)$
5. $\forall x\, F(x) \wedge \forall x\, H(x) = \forall x\,(F(x) \wedge H(x))$
6. $\forall x\, F(x) \vee \forall x\, H(x) \neq \forall x\,(F(x) \vee H(x))$
7. $\exists x\, F(x) \vee \exists x\, H(x) = (\exists x)[F(x) \vee H(x)]$
8. $\exists x\, F(x) \wedge \exists x\, H(x) \neq \exists x[F(x) \wedge H(x)]$
9. $\forall x\, F(x) \vee \forall \underset{x}{z}\, H(\underset{x}{z}) = \forall x \forall z [F(x) \vee H(z)]$
10. $\exists x\, F(x) \wedge \exists x\, H(x) = \exists x \exists y [F(x) \wedge H(y)]$
11. $(Q_1 x) F(x) \vee (Q_2 x) H(x) = (Q_1 x)(Q_2 y)[F(x) \vee H(y)]$
12. $(Q_1 x) F(x) \wedge (Q_2 x) H(x) = (Q_1 x)(Q_2 y)[F(x) \wedge H(y)]$

Convert it into PNF

Q. $(\forall x)(\forall y)[(\exists z)(P(x,z) \wedge P(y,z)) \rightarrow (\exists u) Q(x,y,u)]$

$\Rightarrow (\forall x)(\forall y)[\neg(\exists z)(P(x,z) \wedge P(y,z)) \vee (\exists u) Q(x,y,u)]$

$\Rightarrow (\forall x)(\forall y)[(\forall z)(\neg P(x,z) \vee \neg P(y,z)) \vee (\exists u) Q(x,y,u)]$

$\Rightarrow (\forall x)(\forall y)(\forall z)(\exists u)[\neg P(x,z) \vee \neg P(y,z) \vee Q(x,y,u)]$

(By 1st rule)

Q. $\forall x P(x) \rightarrow (\exists x) Q(x)$

$\neg(\forall x P(x)) \vee (\exists x) Q(x)$

$\Rightarrow \exists x \neg P(x) \vee (\exists x) Q(x)$

$\Rightarrow (\exists x)[\neg P(x) \vee Q(x)]$

Skolem function :→

(Skomalization)

All ∃ quantifiers should
b. left most than followed by
∀

ex: $\forall x \exists y [\neg P(x) \vee Q(y)] = \exists y \forall x [\neg P(x) \vee Q(f(x))]$

Q. $(\exists x)(\forall y)(\forall z)(\exists u) \forall (v)(\exists w) P(x,y,z,u,v,w)$
skomalize it ?

Q. $(\exists x)(\exists u)(\exists w)(\forall y)(\forall z)(\forall v)$.

$P(x, y, z, f(y,z), v, g(y,z,v))$

Conjunctive Normal Form (Imp step to convert sentence into PNF) is

AND of OR

For ex.
1. A 2. A∨B 3. A∧B
4. (A∨B) ∧ (¬C∨D)

Disjunctive Normal form (OR of AND)

1. A 2. A∨B 3. A∧B 4. (A∧B)∨C

Convert into CNF :→

1. $(P \rightarrow Q) \rightarrow R$

$(\neg P \vee Q) \rightarrow R$

$\neg(\neg P \vee Q) \vee R$

$(P \wedge \neg Q) \vee R$

$\Rightarrow (P \wedge R) \cdot (P \vee R) \wedge (\neg Q \vee R)$

2. $P \vee (\neg P \wedge Q \wedge R)$

$= (P \vee \neg P) \wedge (P \vee Q) \wedge (P \vee R)$

$= T \wedge (P \vee Q) \wedge (P \vee R)$

$= (P \vee Q) \wedge (P \vee R)$

Conversion of Predicate logic formula into clause form:-

1. Eliminate all implication & equivalence by law of implication & law of equivalence
   $A \rightarrow B \Rightarrow \neg A \vee B$
   & $A \leftrightarrow B \Rightarrow (A \rightarrow B) \wedge (B \rightarrow A)$
2. Move all negations to immediately proceeds operands by Demargan's law.
3. Convert the formula into Prenex Normal form
4. Apply Skolem function to make all existential quantifiers in prefix of the formula. Delete all existential quantifiers
5. Convert the formula into conjunctive N.F.
6. Eliminate all universal quantifiers & partition the formula into clauses from AND operator.

Convert into clause form:-

$\exists x \forall y [\neg \forall z P(f(x), y, z) \vee (\exists u Q(x,u) \wedge \forall v R(y,v)]$

$\Rightarrow \exists x \forall y [\exists z \neg P(f(x), y, z) \vee \forall u \exists v (Q(x,u) \wedge R(y,v))]$

$\Rightarrow \exists x \forall y [\forall z \neg P(f(x), y, z) \vee \forall u \exists v (Q(x,u) \wedge R(y,v))]$

$\Rightarrow \exists x \forall y [\forall z \neg P(f(x), y, z) \vee \forall u (Q(x,u) \wedge R(y, f(x,u)))]$

$\Rightarrow \exists x \forall y \forall z [\neg P(f(x), y, z) \vee \forall u (Q(x,u) \wedge R(y, f(x,u)))]$

$\forall y [\neg P(f(x), y, f(y)) \vee Q(x, f(y)) \wedge R(y, f(y))]$

$\forall y [(\neg P(f(x), y, f(y)) \vee Q(x, f(y))) \wedge (\neg P(f(x), y, f(y)) \vee R(y, f(y)))]$

1. $\neg P(f(x), y, f(y)) \vee Q(x, f(y))$
2. $\neg P(f(x), y, f(y)) \vee R(y, f(y))$

Q.1 Consider the following given sentences

1. John likes all kind of food.
2. Apples are food.

5. Bill eats peanuts & is still alive.
6. Sue eats everything Bill eats.

1. Convert into F.O. predicate logic.
2. Prove by backward chaining John likes peanuts.
3. Convert the formula into clause form
4. Prove by resolution - John likes peanuts

Following predicates are used :→

1. FOOD(x) : x is food.
2. LIKE(x,y) : x likes y
3. EAT(x,y) : x eats y.
4. KILLED(x) : x is killed.

I → $\forall x$ FOOD(x) ⟶ LIKE(John, x)

II → FOOD(Apple)

III → FOOD(chicken)

IV → ~~$\forall x$ FOOD(x) ∧ EAT(y,x) ⟶ ¬KILLED(x)~~
$\forall x \forall y$ EAT(x,y) ∧ ¬KILLED(y) ⟶ FOOD(y)

here we assume that not killed means still alive
i.e. ¬KILLED(x) ⟷ ALIVE(x)

V EAT(Bill, Peanuts) ∧ ¬KILLED(Bill) or ALIVE(Bill)

VI $\forall x$ EAT(Bill, x) ⟶ EAT(Sue, x)

Remaing at last page

Q.2 The law says it is a crime for a American to sell weapons to ___ a hostile nations. The country None, an enemy of America has some missiles and all of its missiles were sold to it by Kornel west, who is an American

(i) Convert the sentences in above paragraph into predicate logic statements.

(ii) Use resolution to do that West is criminal.

HOSTILE(x) ⟷ ENEMY(x)

Following predicates are used :→

ENEMY(x) : x is enemy country of America

SELLS(x,y) : x sells wepons to y

AMERICAN(x) : x is an American

CRIMINAL(x) : x is Criminal

HAS(x,y) : x has y

(i) I- $\forall x \forall y$ American(x) ∧ ~~SELLS(x,y)~~ ENEMY(y) ∧ SELLS(x,y) ⟶ CRIMINAL(x)

II ENEMY(None) ∧ AMERICAN(West) ∧ SELLS(West, None
~~AMERICAN(West)~~ ∧ has(None, weapon)

(ii) We have to prove CRIMINAL(WEST)

Convert it into Clause form

(a) ¬ American(x) ∨ ¬ Enemy(y) ∨ ¬ sell(x,y) ∨ Criminal(x)

(b) ENEMY(Nono)

(c) American(west)

(d) sell(west, nono)

(e) Has(Nono, weapon)

(f) ¬ Enemy(y) ∨ sell(west, weapon)

Apply resolution b/w clause (1) ← (3) with x = west

(g) ¬ Enemy(y) ∨ ¬ sell(west, y) ∨ Criminal(west)

apply resolution b/w (2) & (f) with y = Nono.

(h) ¬ sell(west, Nono) ∨ Criminal(west)

(i) Criminal(west)

page-167(6)

Q2 Assume the following facts and rules

(i) Steve only likes easy courses.

(ii) Science courses are hard.

(iii) All the courses in electronics dept are easy

Query Steve likes(steve, x)

Use resolution to answer the query what course would steve like.

Ans

Following predicates are used:→

(i) EASY(x) : x is easy course

(ii) HARD(x) : x is hard course ⟺ ¬ EASY

(iii) ELEX(x) : x is electronics course

(iv) LIKE(x,y) : x likes y

~~∀x ELEX(x) :~~

(1) ∀x EASY(x) → LIKE(steve, x)

(2) ¬ EASY(science)

(3) ∀x ELEX(x) ⟶ EASY(x)

(4) ELEX(EL-301)

Clause form :→

(1) ¬ EASY(x) ∨ LIKE(steve, x)

(2) ¬ EASY(science)

(3) ¬ ELEX(x) ∨ EASY(x)

(4) ELEX(EL-301)

quantifiers are coming at left

(5) EASY (EL-301)

Apply unif^n b/w (1) & (5)

(6) LIKE (Steve, EL301) (which was the query)

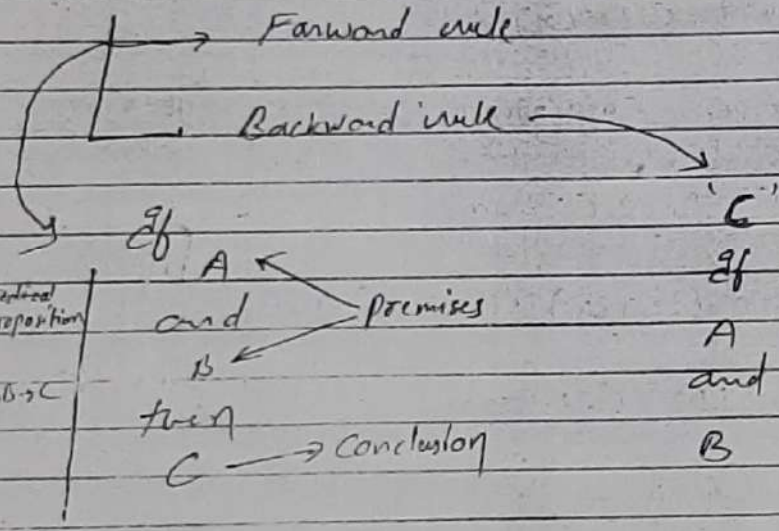
Backward Chaining :→

Knowledge Represented by Rules & facts

Rules

→ Forward rule

→ Backward rule → 'C'



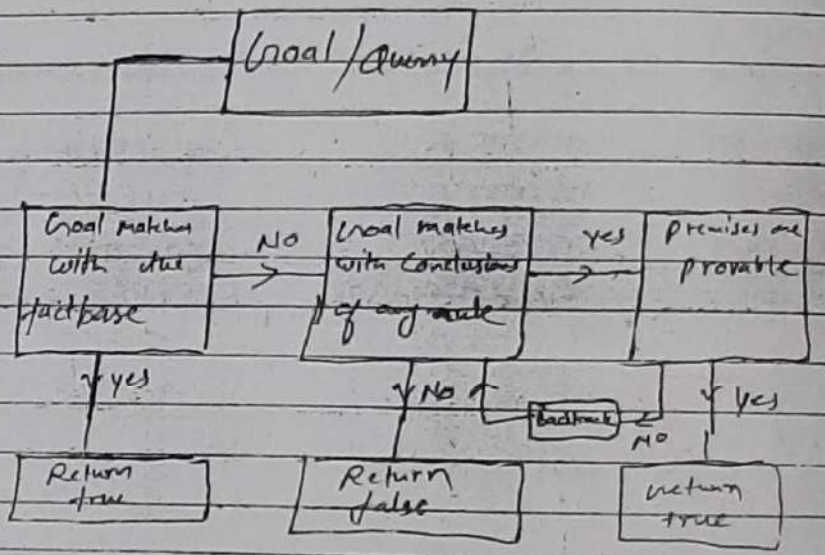
Ex 1.

If
Easy(x)
then
Like (Steve, a)

Backward Reasoning / Chaining :→

When the rules are written in the backward form and various rules and facts are changed (chained) for reasoning then this method is known as backward chaining

Block Diagram of backward Chaining :→



Ex Rules :→ (previous ex. rules)

(1) ~~Easy(x)~~ like (steve, x)
If
Easy(x)

(2) Easy(x)
If
EIEx(x)

facts:→

② ¬easy(science)

③ Flex(FL-301)

Query if Like(Steve, FL-301)

Query is matched with conclusion of rule-1

like(Steve, x)

|

easy(x) ← premises is no matched with any fact

Try to match it with conclusion of other rules.

→ matched with conclusion of rule no-2

|

FLEX(x)

matched with fact no.-2 with x = FL-301

B.1 Backward chaining:→

1. ~~∀x~~ ~~FOOD~~ FOOD(x) → LIKE(John, x)
2. FOOD(Apple)
3. FOOD(chicken)
4. ∀x∀y EAT(x,y) ∧ ¬KILLED(x) → FOOD(y)
5. EAT(Bill, peanuts) ∧ ¬KILLED(Bill)
6. ∀x EAT(Bill, x) → EAT(Sue, x)

Facts:→

1. FOOD(Apple)
2. FOOD(chicken)
3. EAT(Bill, peanuts) ∧ ¬KILLED(Bill)

Rules:→

1. FOOD(x) → LIKE(John, x)
2. EAT(x,y) ∧ ¬KILLED(x) → FOOD(y)
3. EAT(Bill, x) → EAT(Sue, x)

Query is LIKE(John, peanuts)

Rule 1. LIKE(John, x)

If

2. FOOD(y)
   ∀
   EAT(x,y)
   and
   ¬ KILLED(x)

3. EAT(Sue, x)
   ∀
   EAT(Bill, x)

Query → LIKE(John, peanuts).

↓

FOOD(peanuts)

↓

EAT(x, peanuts)

∧ ¬ KILLED(x)

↓ which is a fact so the given query is true

Searching

- Search
  - Random search
  - Systematic search
    - BFS
    - DFS
  - Efficient search
    - Optimal search
    - Hemistic search
      - Hill climb

Complexity of any greedy technique can not be more than complexity of sorting.

Hemistic Search :→ Searching for best but if not found then compromise with good solution. We will concentrate on the complexity more.

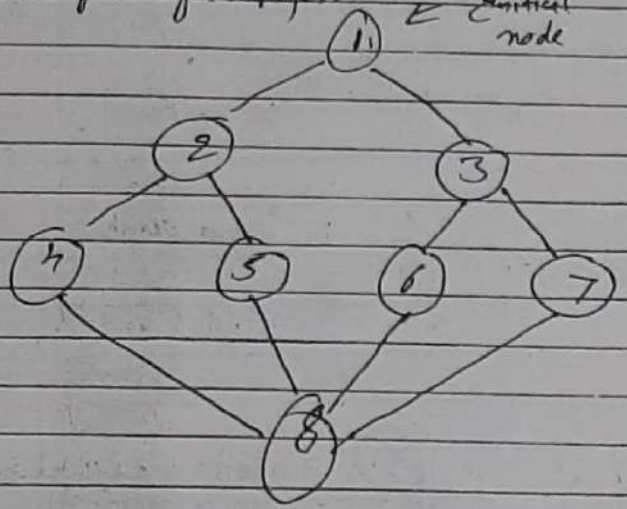
Traversal :→ Searching of whole dist ma

We will reach to last node.

Search :→ We search till the require element is found.

Breadth First Search (BFS) :→

State space diag (Graph) :→



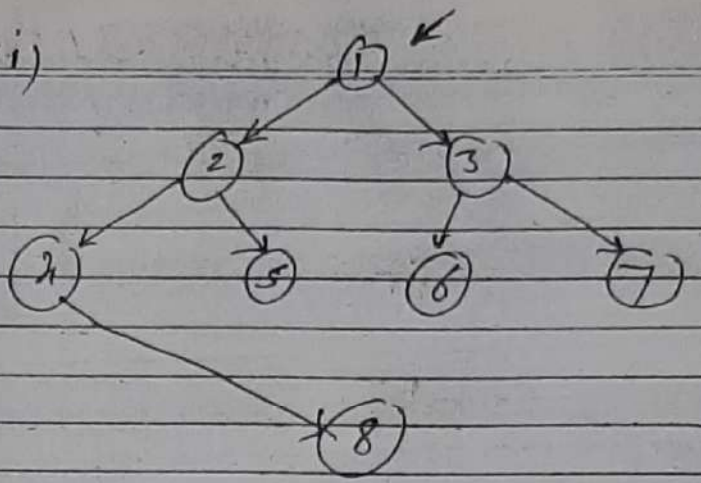
← Initial node

Here queue is used.
Here there is no left right nodes.

1

Q. For the given graph if searching starts at node as (1) find
(i) BFS spanning tree
(ii) Order of traversal.

B (i)



Order : 1, 2, 3, 4, 5, 6, 7, 8

1

~~1~~	2	3

~~2~~	3	4	5

~~3~~	4	5	6	7

1 has been traversed

~~4~~	5	6	7	8

~~5~~	6	7	8

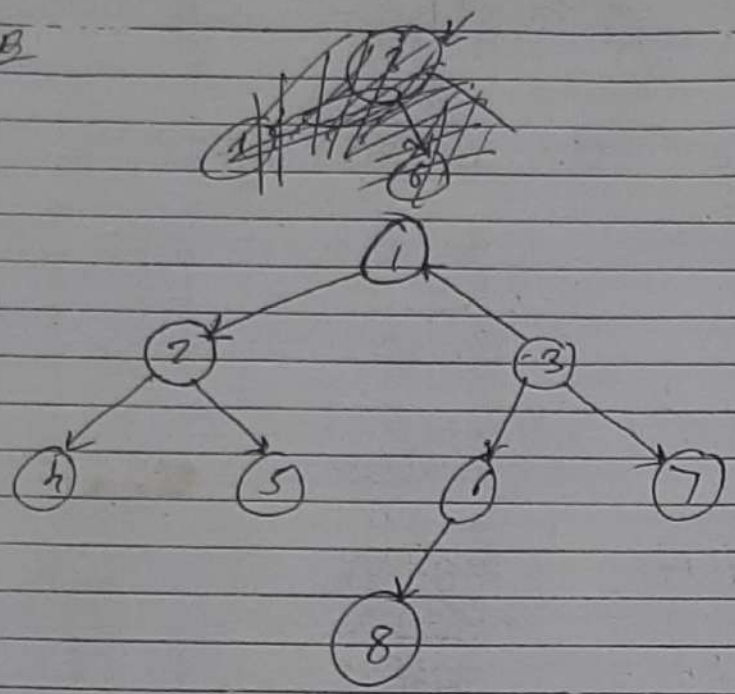
~~6~~	7	8

~~7~~	8

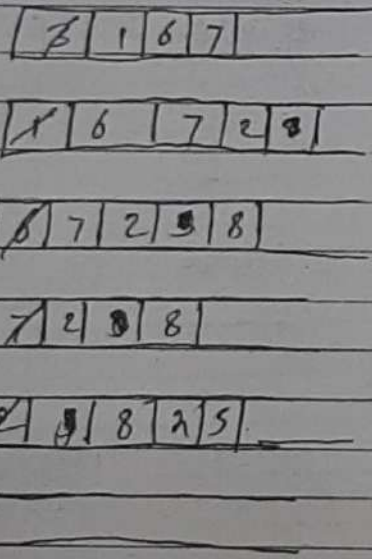
~~8~~

Q. Start from node-3 and do same

B

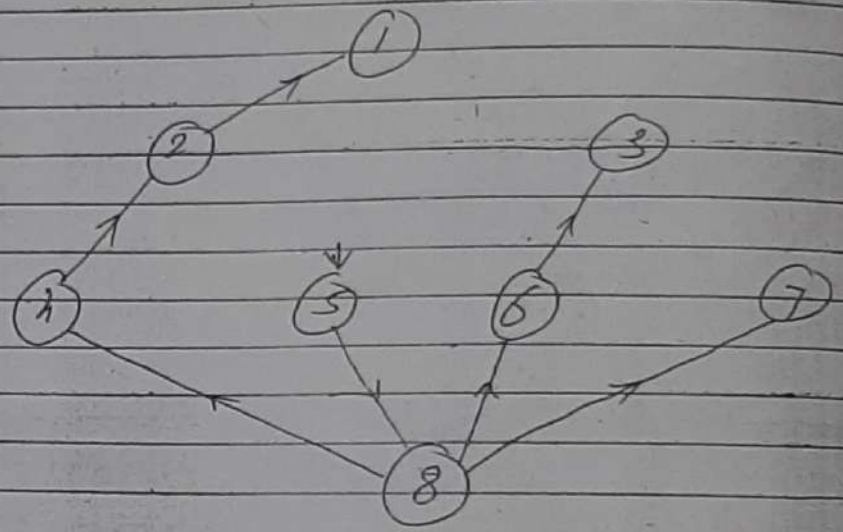


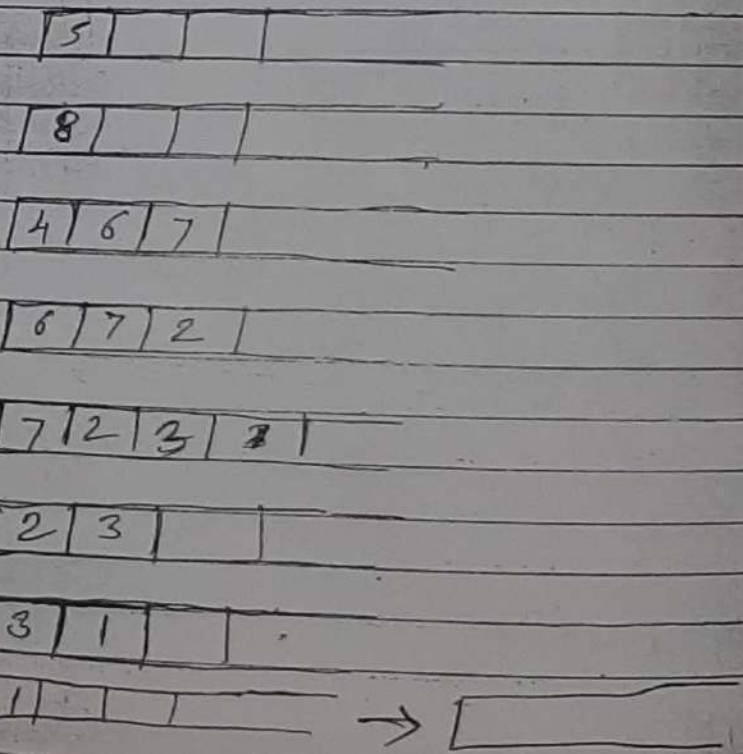
Order :→ 3, 1, 6, 7, 2, 8, 4, 5



Q. DO same from (5) & (8)

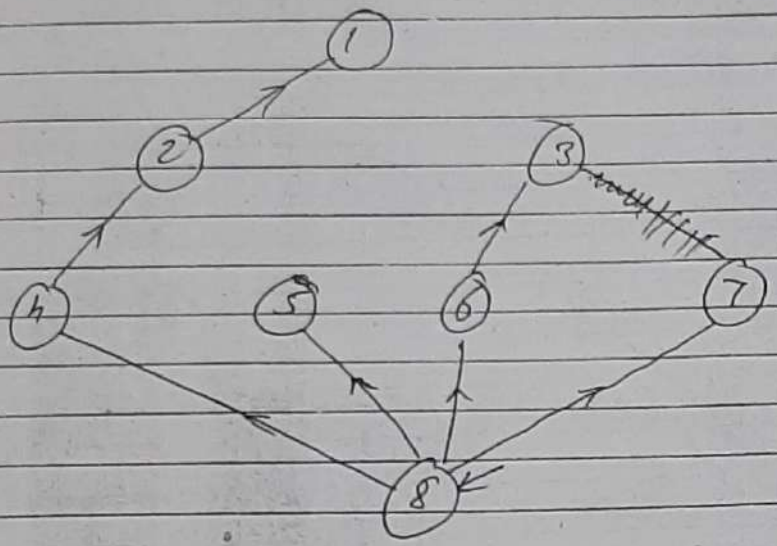
Order :→ 5, 8, 4, 6, 7, 2, 3, 1

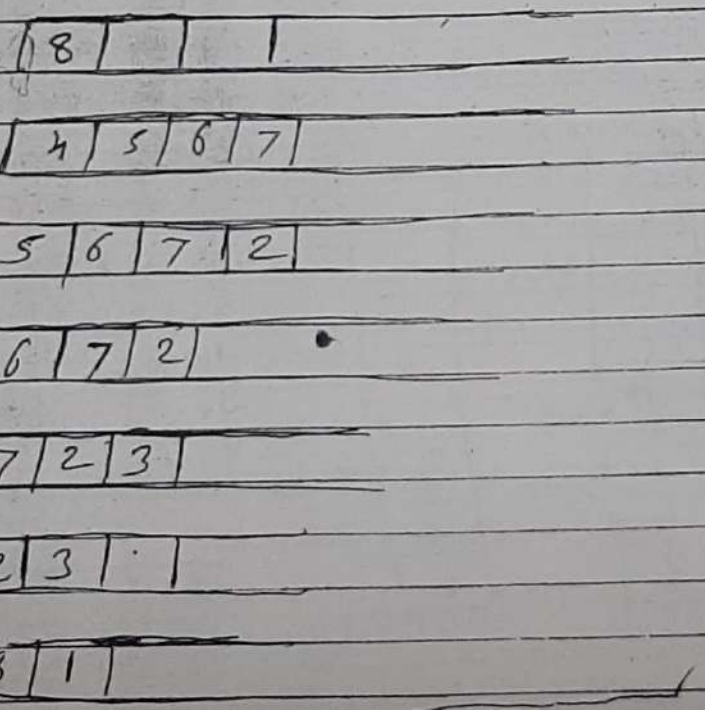




From node (8) :-

Order :-

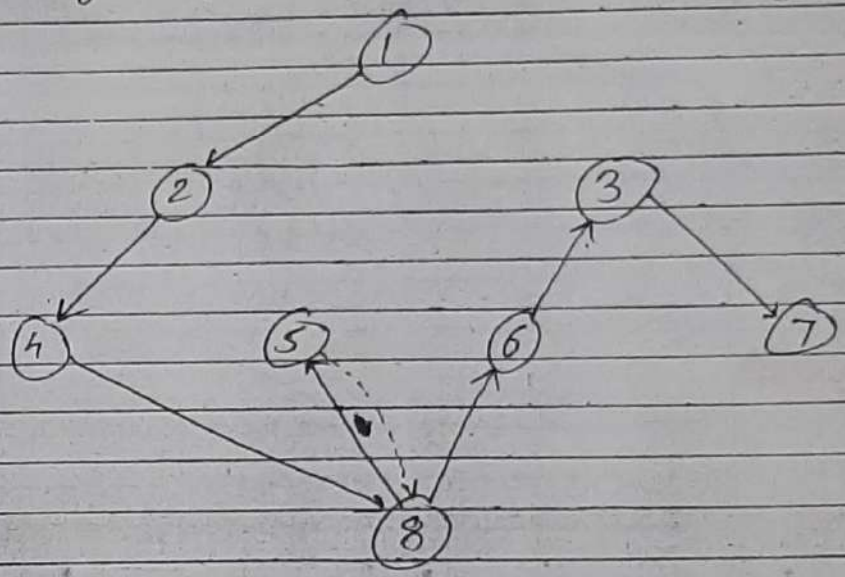


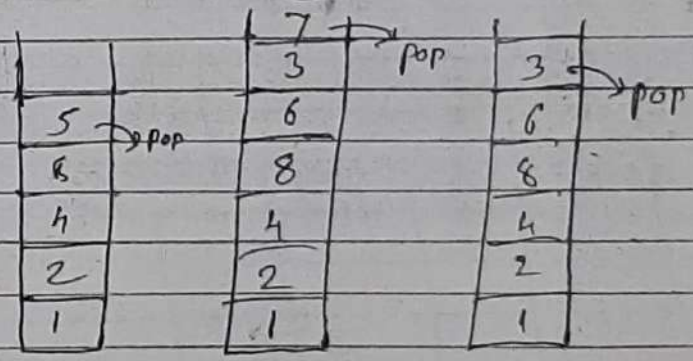


Depth first search (DFS) :-

Do the same question by DFS

Spaning tree :-

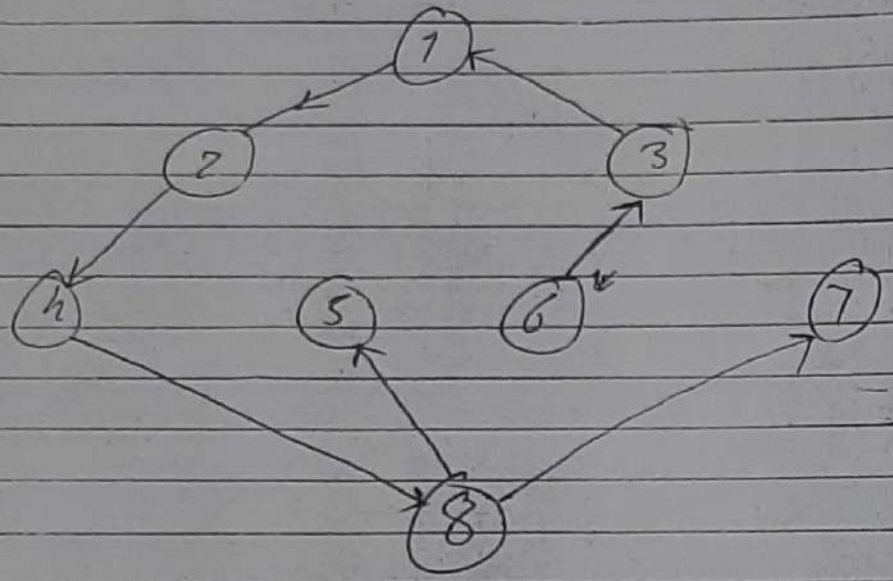




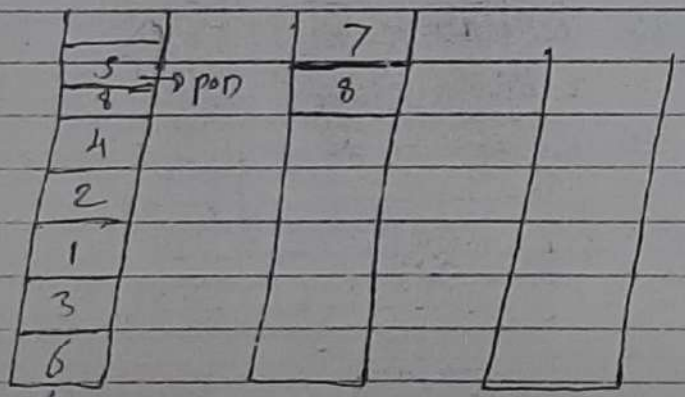
Order :- 1, 2, 4, 8, 5, 6, 3, 7

Q. Start from node (6)

A



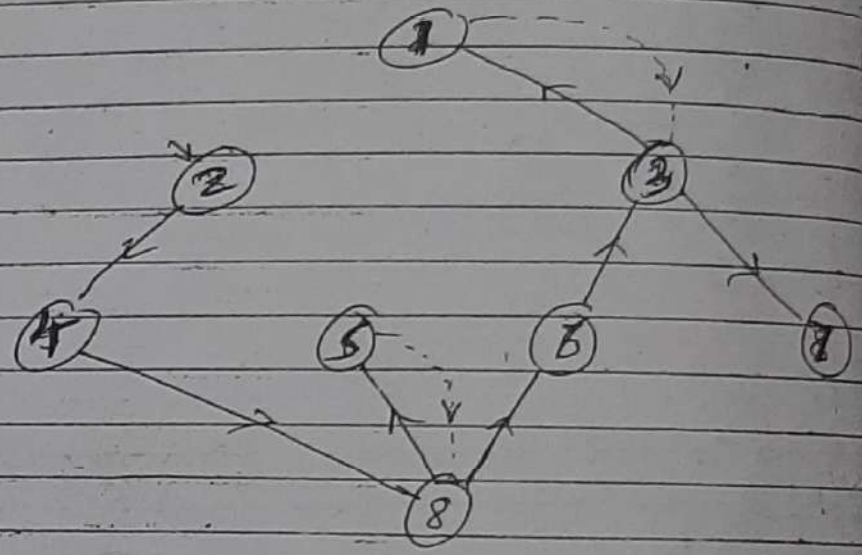
Order: 6, 3, 1, 2, 4, 8, 5, 7

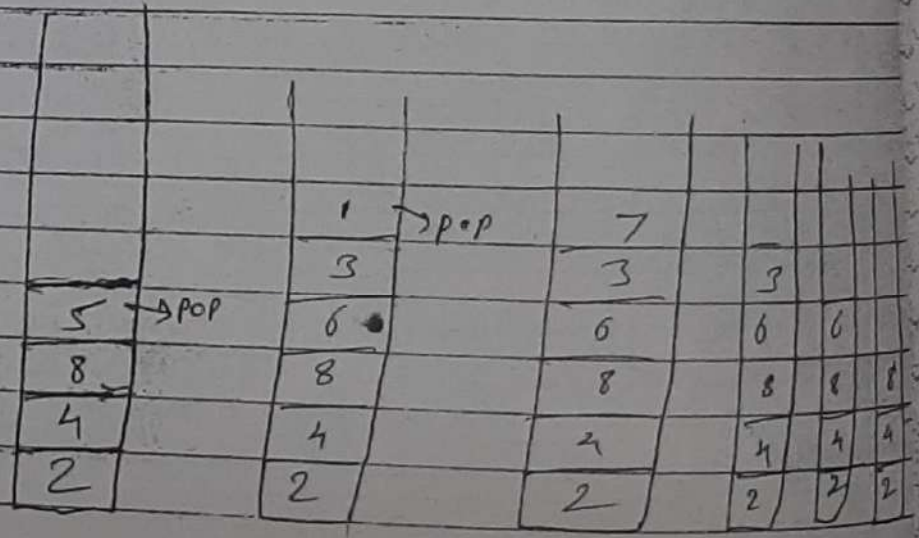


Q. Make it by (8) and (2)

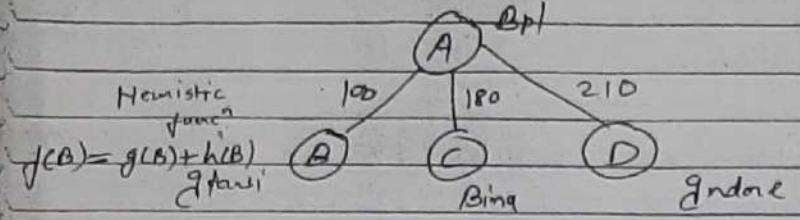
A with (2)

Order: 2, 4, 8, 5, 6, 3, 1, 7





Hemistic Search:→



(H) Goal (chennai)

Hemistic function:→

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

where.

f(A): Total cost from node A to goal node

g(A): Actual cost to reach to node A from the initial node.

h(A): It is a hemistic cost of node A i.e. estimated cost calculated by applying hemistic function to reach the goal node.

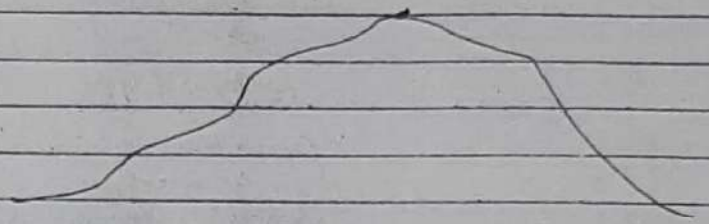
predicted cost

$f(B) = 100 + 900 = 1000$

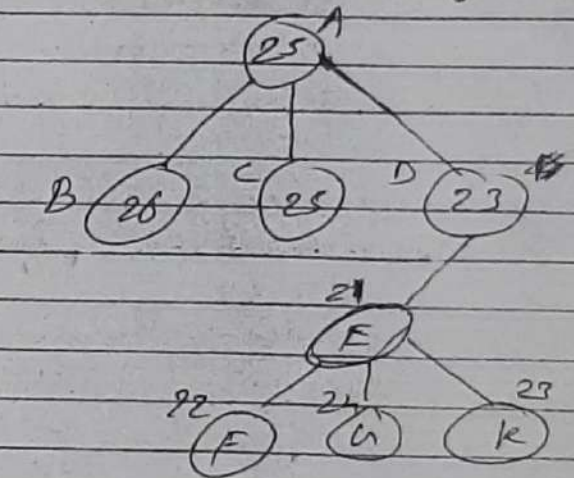
$f(C) = 180 + 800 = 980$

$f(D) = 210 + 600 = 810$

Hill climbing algo:→



Here we do the same as we do in case of making steps.



Algorithm:→

(1) Check whether initial node is the goal node if yes then stop with successful search Otherwise calculate its total cost and make it the ...

2 Repeat the following steps-

(a) Generate one possibility of current node.

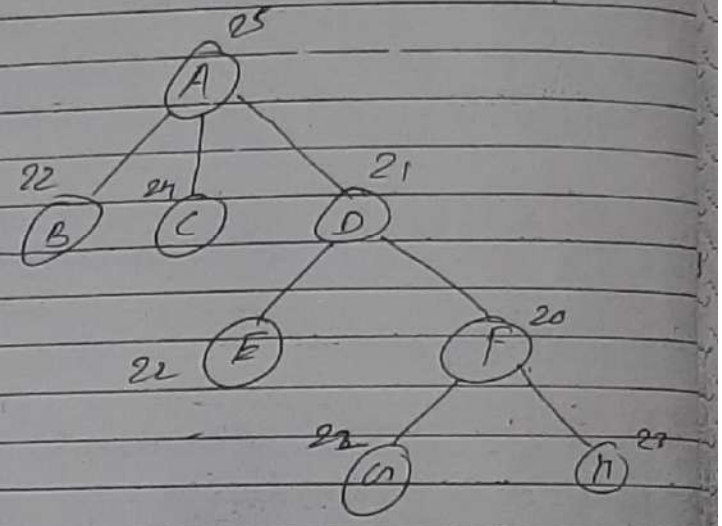
(b) Check whether this node is goal if yes then stop with successful search.

(c) If no calculate the cost of the possibility (newly generated node).
If cost is better than the current then make new node as a current node and goto step (2).

(d) Check whether the cost of newly generated node is worse than generate other option and do the same as of step (a) (b) (c)
If no option left then stop with unsuccessful search.

Disadvantage:-
There is no guarantee of solution.

Steepest Hill Climbing:-



From algorithm point of view it is more promissing algo.

So it will give better soln than hill climbing.

Its time complexity will be more because we search for the best option

Algo →

① Check whether the initial node is the goal node if yes then stop with successful search.
Otherwise calculate the cost of and make it current node.

2. For the current node generate all possible options (children).

For each option do the following.

(a) If goal node is found stop with successful search.

(b) Calculate the cost of the node.

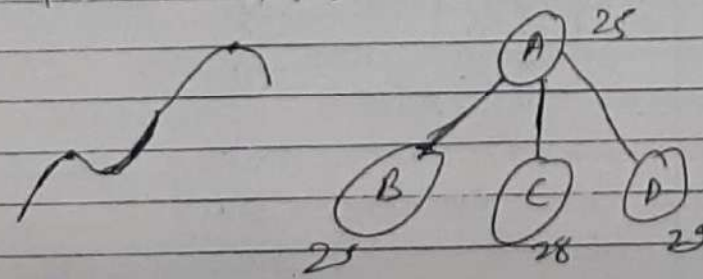
(c) Select best cost node (min cost) and compare its cost with current node.

If the cost is better than the current node it current node and goto Step-2.

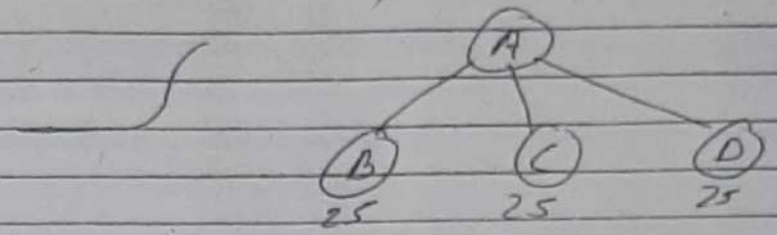
If cost is worse than the current then stop with unsuccessful search.

Problems & their Soln associated with these algos:

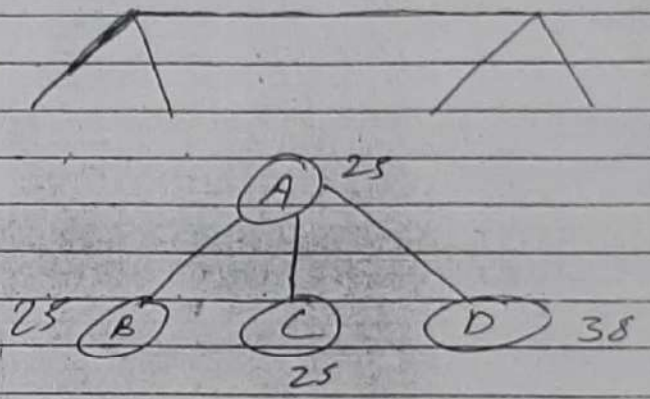
(1) Local Maxima :→



(2) Plateau :



(3) Ridge :-



Solutions :→

(1) Backtracking.

(2) Ignore some steps

But with these soln's also there is no guarantee of finding the goal

& they may also lead very high complexity ($O(2^n)$).

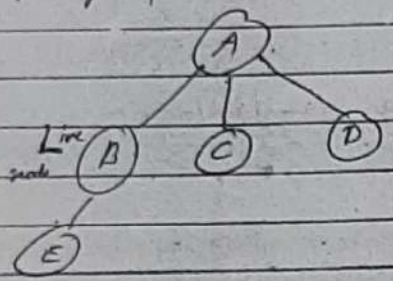
Branch and bound :→

One of the examples of it is Best First Search.

Best First Search :→

Live Nodes (Open List) :-

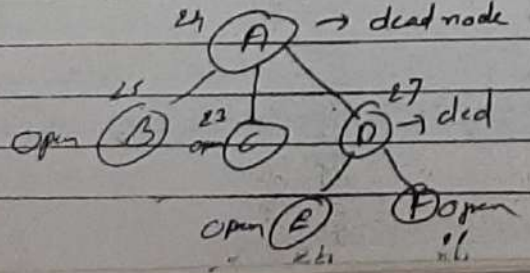
Nodes which are either partially expanded or not expanded at all are known as live nodes. Set of live nodes (list) is known as open list.



Dead nodes (Closed list) :→

Nodes which are either fully expanded and no further expansion is possible are known as dead nodes.

Set of dead node (list) is known as Closed list.



Closed list = {A, D}

Open list = {B, C, E, F}

E-Node →

Node selected from the open list which is most promising for further expansion is known as E-node.

E-Node = {C}

Algorithm :→

Step 1- Check whether initial node is goal node if yes then stop with successful search. Otherwise calculate cost and place it in open list.

2. Select most promising node from open list (Least cost node). Selected node becomes E-node.

Do the following.

a. Generate all possibilies from E-node and place it in closed list once it becomes dead node.

Do the following for each newly generated node

(a) Check whether the generated node is goal node then stop with successful search.

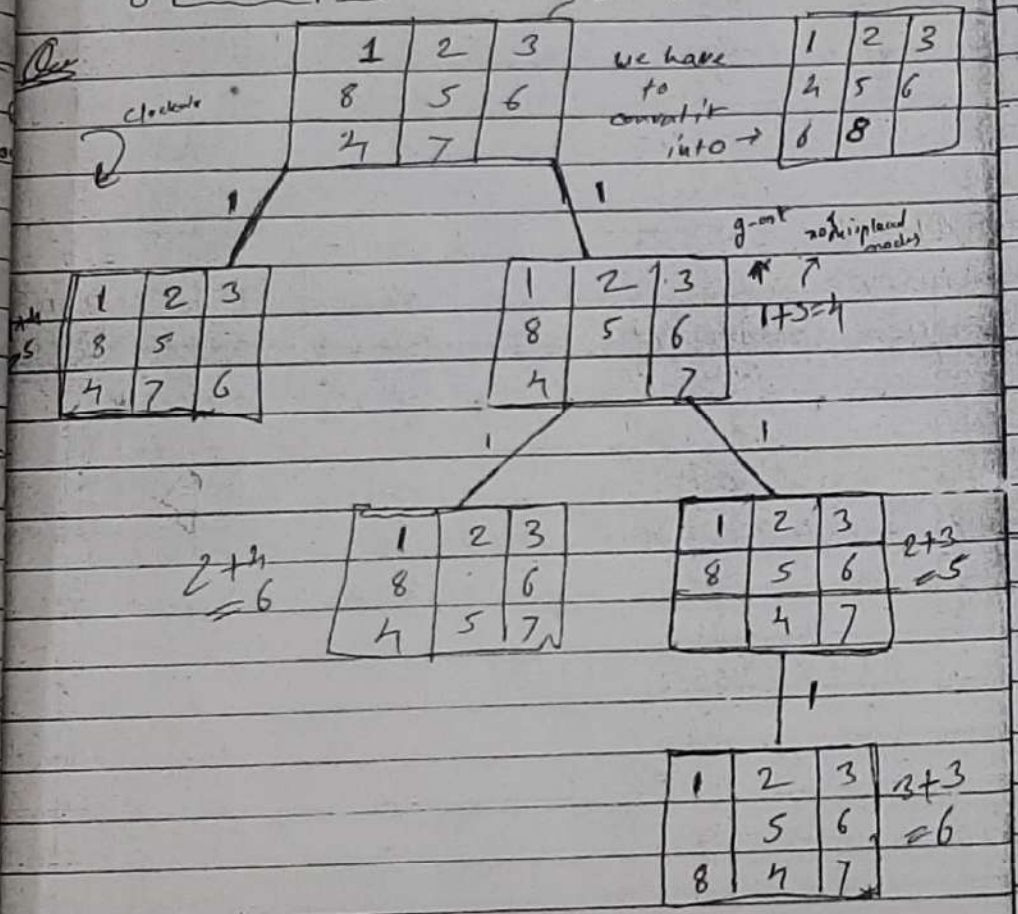
(b) Otherwise calculate the cost of it...

and place it in open list.

(c) goto step - 2

3. If open list is empty then Stop with unsuccessful search.

8-puzzle problem:-



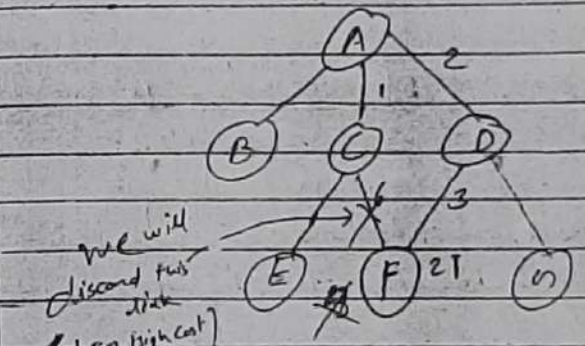
In previous example there was one initial & 1-goal

1. 1-initial, 1-goal → Ludo, Snake & ladders
2. Many - one → 15 puzzle, 24 puzzle
3. ~~Many~~ 1 to many → Chess, s.
4. Many-many → Tic-Tac-To, sudoku

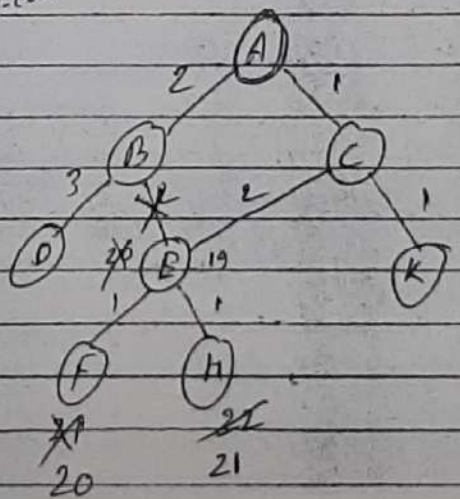
In 1-1 you can proceed from any one

A* Algorithm:-

State space diagram is a Graph here



Here the children is already in Open list



Here the child is already in close list

Here we will do same as upper above case and

a node if it exists in state space diag

A*

Algorithm :→

It is one of the branch and bound techniques and very similar to best First Search the difference is in BFS the state space diag. is assumed as tree but in A* algo stat space diagram may be graph. So A* algo is more general version of Best First Search which is applicable for not only tree but also for graph.

Algo:→

(1) Check whether the start node or initial node is goal node if yes then stop with successful search otherwise calculate its cost and make it current node. place it into open list.

(2) ~~Generate~~ Select best option (least cost node) from open list generate all its children and for each child do the following :→
(If open list is empty then stop with unsuccessful search)

(a). Check whether the node (child) is goal node then stop with successful search else

(b) Calculate its cost

(c) Is the node already in open list, if yes then compare its new cost with already stored cost. If new cost is better then change the parent of the node as per new path and delete old cost and restore new cost see the previous diags.

(d) check whether the node already in closed list then compare its new cost with old cost, if the new cost is less

(i) Change the parent link to new path
(ii) Restore new cost and delete old cost
(iii) Change the cost of all descendants see diag (Imp to show)

(e) Place the node in open list and goto step no. 2.

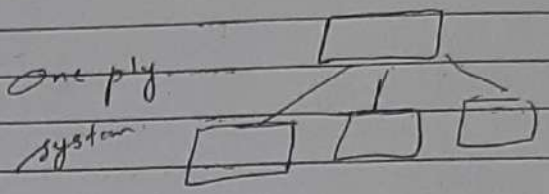
$$P(H \& E) = \frac{P(H|E)}{P(E)}$$

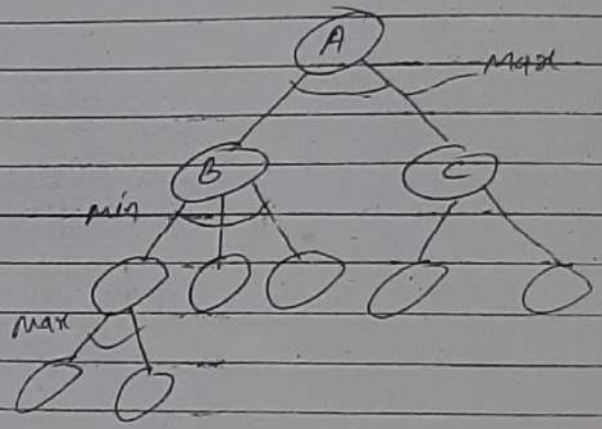
$$P(E \& H) = \frac{P(E|H)}{P(H)}$$

Game Playing-

Max-min Technique

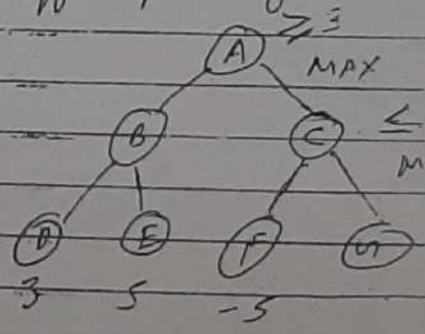
↳ α-β cutoff (Pruning)





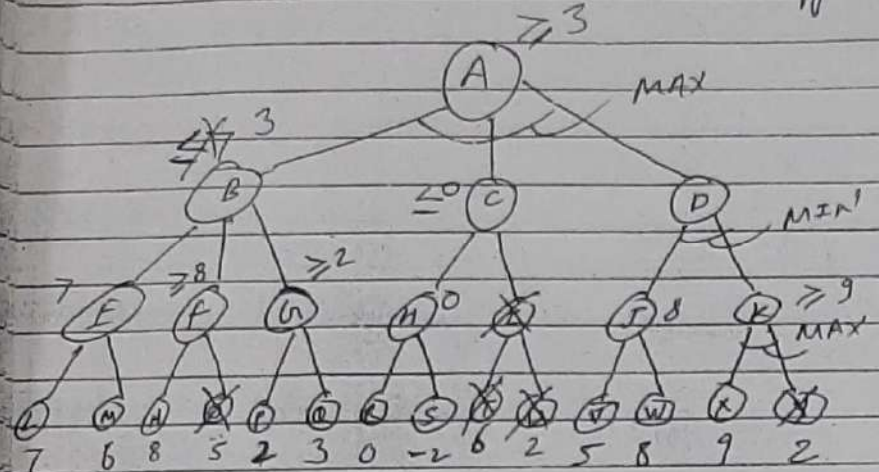
Max-min algo

α-β cutoff (pruning) :-

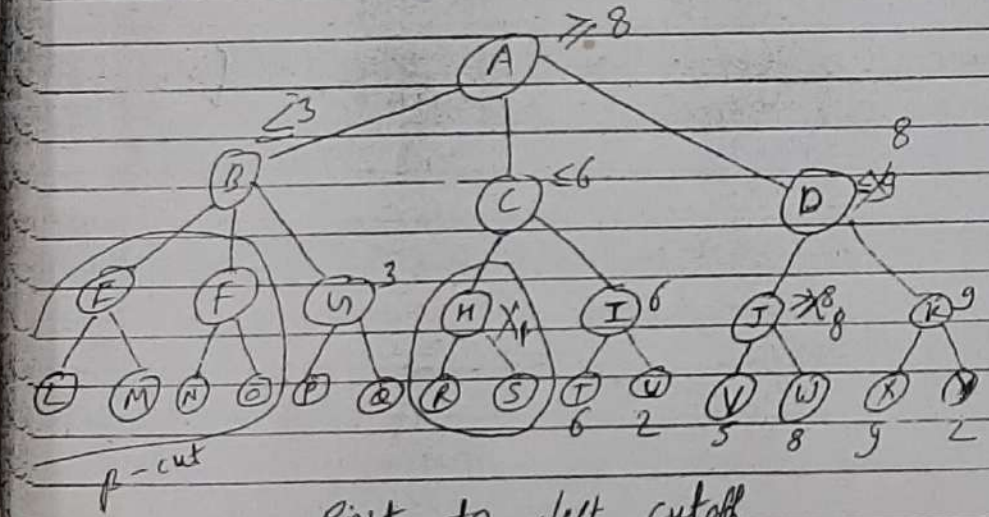


lower bound — α
Upper bound — β

Q. Calculate M by Max-Min & α-β cutoff :-



Left to Right cutoff.
So cutted nodes are → O, I, I, U, Y



Right to left cutoff

Unification Algorithm :→

Clauses:-
1. $P(x,a) \vee Q(g)$
2. $\neg P(b,g)$ — unify
Apply resolution b/w Cl. 1 & (2) with $x=b$ & $a=g$
3. $Q(a)$

This process is called Unification.

Algo

Input $W = \{P(a,y), P(a,b)\}$

Output :- $\sigma' = \{a/x, b/y\}$
(Most general unifier) (mgu)
(mgu)

Algorithm :-

Step-1 Set $k=0$, $W_k = W$ & $\sigma_k = \phi$

Step-2 If W_k is singleton (A set whose cardinality is 1), stop,
σ_k is mgu of W.
else find Disagreement set D_k

Step-3 If there exist elements V_k & t_k in D_k such that V_k is variable that does not occur in t_k, goto step 4.
else stop, W is not unifiable.

Step-4 $\sigma_{k+1} = \sigma_k \cup \{t_k / V_k\}$

5. K = K+1, goto step 2.

Ex. $W = \{P(a, x, f(g(y))), P(z, f(z), f(u))\}$?

1. $K=0$, $W_0 = W$, $\sigma_0 = \{\}$

2. $D_0 = \{a, z\}$

3. $V_0 = z$, $t_0 = a$

4. $\sigma_1 = \{\} \cup \{a/z\} = \{a/z\}$
$W_1 = \{P(a, x, f(g(y))), P(a, f(a), f(u))\}$

5. $K = 1$

Pass-2

2. $D_1 = \{x, f(a)\}$

3. $V_1 = x$, $t_1 = f(a)$

4. $\sigma_2 = \{a/z, f(a)/x\}$

5. $W_2 = \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}$

5. $K = 2$

Pass-3

2. $D_2 = \{f(g(y)), f(u)\}$

3. $V_2 = u$, $t_2 = g(y)$

4. $\sigma_3 = \{a/z, f(a)/x\} \cup \{g(y)/u\}$
$W_3 = \{P(a, x, f(g(y))), P(a, f(a), f(g(y)))\}$

Ex. $W = \{Q(f(a), g(x)), Q(y, y)\}$

Ans

1. $K=0$, $W_0 = W$, $\sigma_0 = \{\}$

2. $D_0 = \{f(a), y\}$

3. $V_0 = y$, $t_0 = f(a)$

4. $\sigma_1 = \{f(a)/y\}$
$W_1 = \{Q(f(a), g(x)), Q(f(a), f(a))\}$

5. $K = 1$

Pass-2

2. $D_1 = \{g(x), f(a)\}$

3. $V_1 = x$, $t_1 = g(x)$

4. $\sigma_2 = \{g(x)/y\}$
$W_2 = \{Q(f(a), g(x)), Q(f(a), f(g(x)))\}$

5. $K = 2$

Here there is no V_K and t_K exists for agreement set so it is not unifiable.

Planning :-

Divide and Conquer :- In divide & conquer we divide the problem in subproblems which are the same as the original problem but smaller.

While Planning is divides the problem into the sequence of subproblems.

↳ Goal Stack Planning :→

Ex. Block World Problem :-

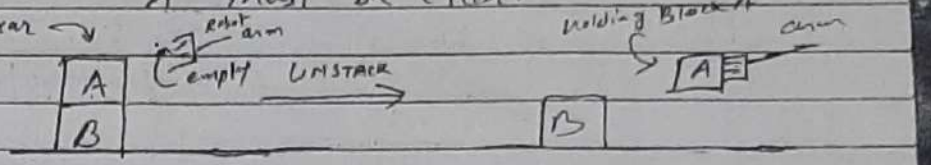
Predicates and Proposition used to describe the state :-

1. ON(A,B) : Block A is on block B
2. ONTABLE(A) : A is on the table
3. CLEAR(A) : There is nothing on the top of A
4. HOLDING(A) : The Robot arm is holding block A
5. ARMEMPTY : Robot arm is holding nothing (empty)
   ↳ Proposition

The actions performed by the Robot :-

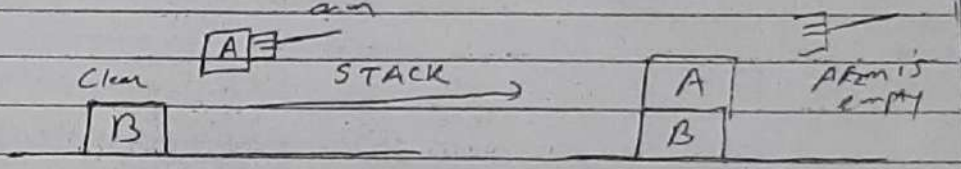
1. UNSTACK(A,B) : Pickup block A from its current position which is on B

Condition : Arm must be empty and block A must be clear.



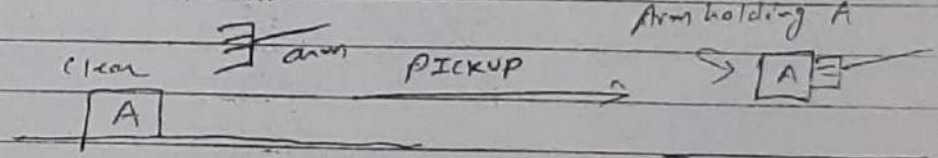
2. STACK(A,B) : Place block A on block B.

Cond : (i) The arm must already be holding block A. and
(ii) Top of B should be clear



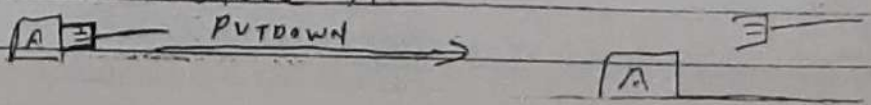
3. PICKUP(A) : Pick block A from the table and hold it.

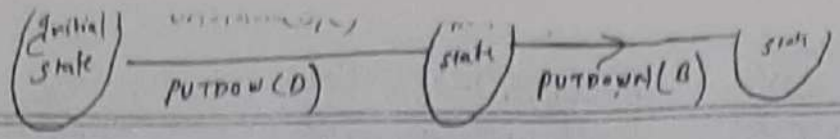
Cond : (i) Robot arm must be empty.
(ii) Top of A must be clear.

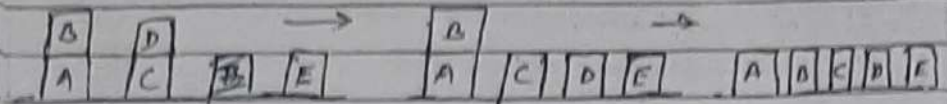


4. PUTDOWN(A) : Put block A down on the table.

Cond : Arm must be already holding block A.

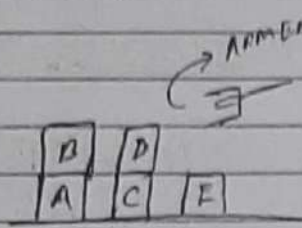




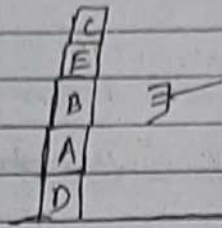


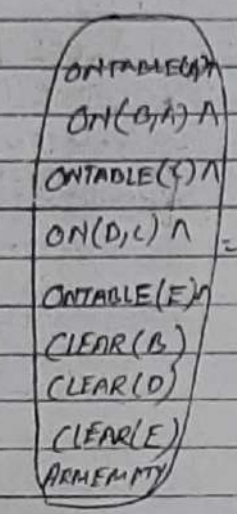
Ex.

Initial state

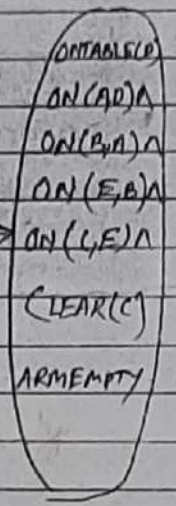


Final state





UNSTACK(B,A)
PUTDOWN(B)
UNSTACK(D,C)
PUTDOWN(D)
~~STACKPICKUP(A)~~



STACK(A,D)
PICKUP(B)
STACK(B,A)
PICKUP(E)
STACK(E,B)
PICKUP(C)
STACK(C,E)

To describe states predicates are used and for describing actions (transition) functions (actions) are used.

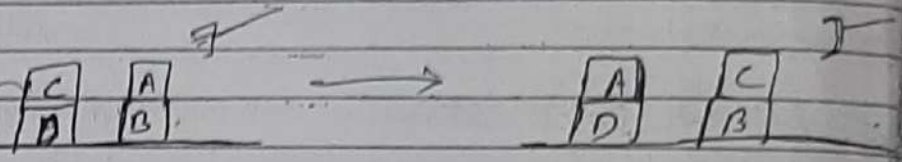
Ex

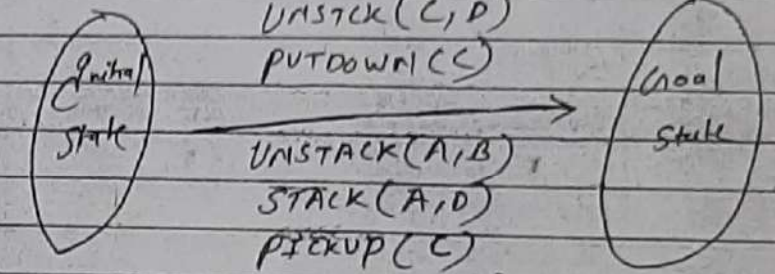
Initial State

ONTABLE(A) ∧
ON(C,D) ∧
ONTABLE(B) ∧
ON(A,B) ∧
ARMEMPTY

Goal Final State

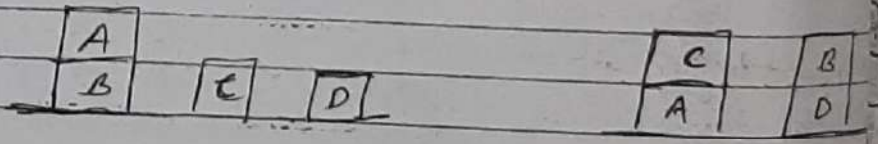
ONTABLE(A)
ONTABLE(B)
ON(A,D)
ON(C,B)
ARMEMPTY





UNSTACK(C,D)
PUTDOWN(C)
UNSTACK(A,B)
STACK(A,D)
PICKUP(C)
STACK(C,B)

Ex



Initial State

Goal State

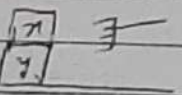
Initial	Actions	Goal
ON(A,B) ∧	UNSTACK(A,B)	ONTABLE(A)
ONTABLE(C)	PUTDOWN(A)	ON(E,A)
ONTABLE(D)	PICKUP(C)	ONTABLE(D)
CLEAR(A)	STACK(C,A)	ON(B,D)
CLEAR(C)	PICKUP(B)	CLEAR(C)
CLEAR(D)	STACK(B,D)	CLEAR(B)
ARMEMPTY	↑	ARMEMPTY

Q. Write precondition, Addlist, Deletelist for the actions considered in block world problem?

A. 1. Preconditions :- (i) STACK(x,y)

b STACK(x,y) : CLEAR(y) ∧ HOLDING(x)

2. Deletelist :-



CLEAR(y) ∧ HOLDING(x)

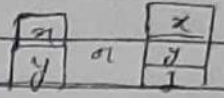
3. Addlist :-

ON(x,y) ∧ ARMEMPTY

Conf :- CLEAR(x)

(ii) UNSTACK(x,y)

Preconditions :-



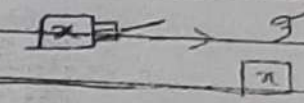
ARMEMTY ∧ CLEAR(y) ∧ ON(x,y)

D:

ON(x,y) ∧ ARMEMPTY

A: CLEAR(y) ∧ HOLDING(x)

(iii) PUTDOWN(x)



P:

HOLDING(x)

D:

HOLDING(x)

A: ARMEMPTY ∧ ONTABLE(x)

(iv) PICKUP(x) :-

P: ARMEMPTY ∧ CLEAR(x) ∧ ONTABLE(x)

D: ONTABLE(x) ∧ ARMEMPTY

A: HOLDING(x)

Knowledge Representation Techiques:→

1. By Frame :→

FRAMES :→ Frames are used to represent knowledge where declarations and procedures are combined into single knowledge representation environment. Frames are very similar to the objects, i.e. both data and procedures are packed into common structure.

STRUCTURE OF FRAME:→ A frame consists of :-

(i) The "Name" of the frame → Each frame should have unique name.

(ii) "Parent" of the Frame :- Frame uses concept of Inheritance. The properties inherited by this frame is from its Parent.

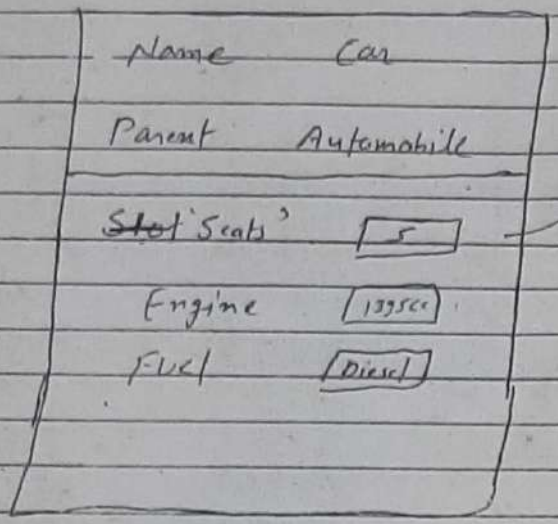
(iii) "Slot names" and "their "Values":- Each frame may have large no. of slots. Within a frame the slot names must be distinct. But 2- different frames may have same "Slot name" i.e. slot names are local to the Frame. Each slot has some value associated with it. Slots are considered to be similar as of fields of structure.

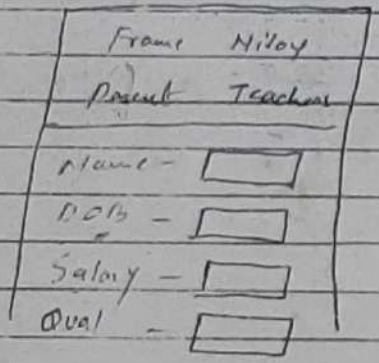
(iv) "Attached Predicates" in each slot :- This is a unique property of the Frame knowledge representation technique that each slot can be associated with 2- predicates. The purpose of these predicates are to provide security when the value of that slot is accessed or changed. These Predicates are as follows :→

(a) "If Needed" Predicate :- If a Slot has an If needed predicate then before the value of the slot is found and obtained, The If Needed predicate must be successfully proven. (Satisfied). The use of If Needed predicate is optional.

(b) If added predicate :- If a slot has an If Added predicate, Before the value of the Slot is assigned or modified, the If Added predicate must be successfully proven. The use of this predicate is optional.

Pictorial Representation of a Frame:-





in Frames. (Feature of Frame)

By Script :- It is data representation technique which is used to represent seq^n

Important components of the script are as follows

① Entry Conditions :- Set of conditions that must be satisfied before the events strictly described in script occur.

② Result :- Conditions that will be true after the events described in script have occured.

③ Props :- These are the physical objects which are essential parts of events described in script.

④ Roles :- People there are the part of the events/ scenes described in script are known as roles.

⑤ Track :- A same script may have different no. of tracks. These tracks share many but not all components of the script.

⑥ Scenes :- The actual seq^n of even

known as scenes. Practically the scenes are represented using Conceptual dependencies (CD).

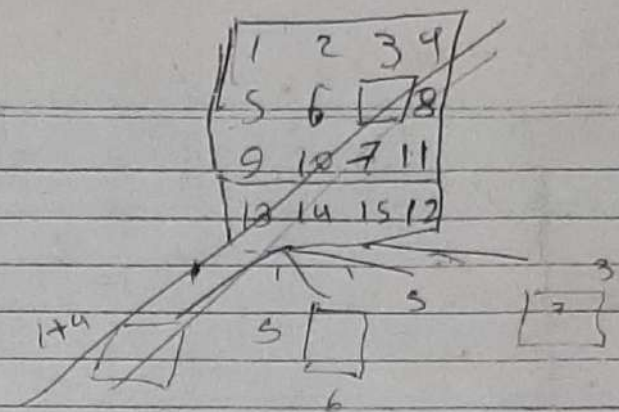
ex.

Script of Restaurant (Pictorially)

These are actually represented in CD but we write in english.

Script: Restaurant Track: Coffee house Props: Table, chair, food, Money	Scene1: Customer will go to rest. Scene2: Customer will enter rest Scene3: Waiter will welcome him Scene4: Customer will sit on table Scene5: Customer will sit to the table
Roles: Hungry customer, Manager, waiter, Chiefs.	
Entry conditions: Customer is hungry Customer has money	Scene6: Customer waiter will ask for services to customer Customer will serve water Waiter will serve the required Customer will eat.
Result: Customer is satisfied Customer has lost money Manager will have one more	

Ex: Script university examination



weak slot and filler

weak slot and filler

Semantic Net:-

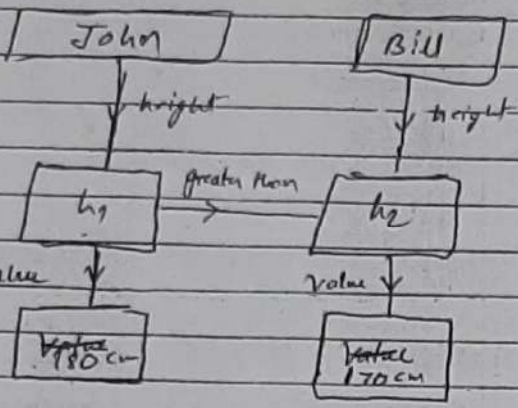
Semantic Net is a weak slot and filler knowledge representation technique. In weak slot and filler knowledge repr^n different entities are connected to each other by infinite set of natural language terms. In this knowledge repr^n, it is very difficult to interpret correct meaning by diff. AI environments.

Q. Construct semantic net for following group of sentences

1. John's height is 180cm

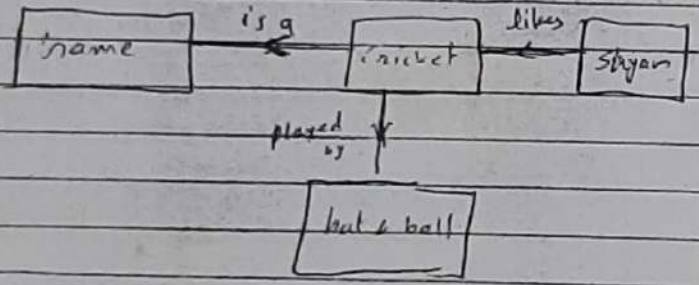
3. John is taller than Bill.

Ans

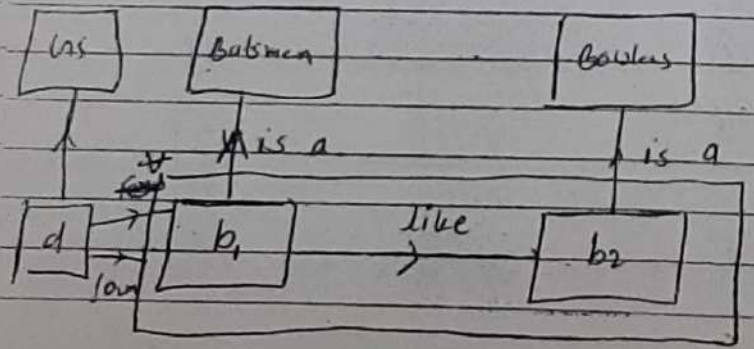


Que 1. Cricket is a game.
2. It is played by bat and ball.
3. Shyam likes Cricket.

Ans



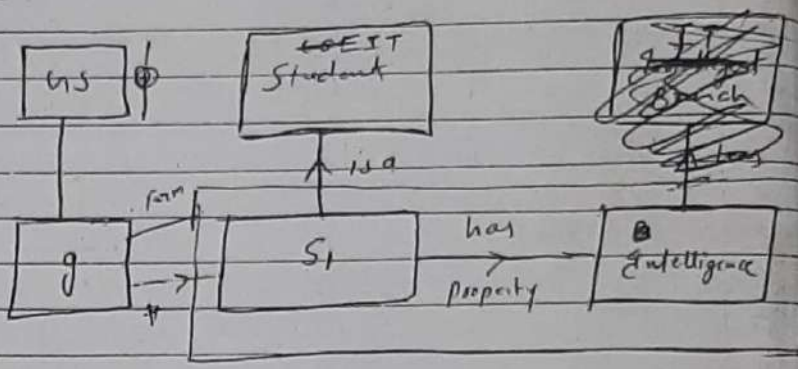
Que 1. All Batsman like bollers. bowlers



Que 1. General statements not related with other objects.

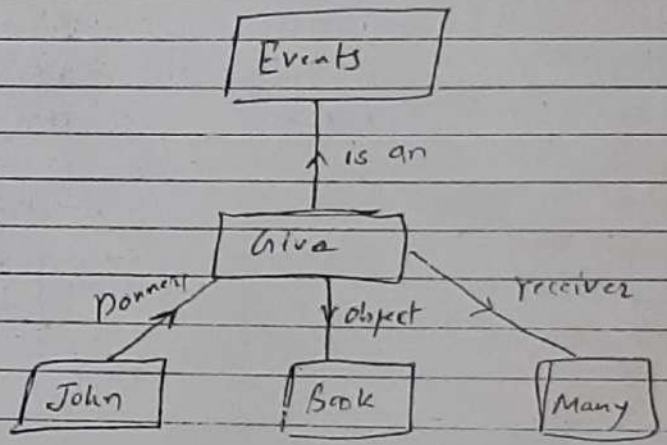
Que All IT Students are intelligent

Ans



Que John gave the book to many.

Ans



Conceptual Dependency :→ It is one of the type of strong slot & filler structure.

C.D. :→ If the knowledge is represented in the form of semantic net i.e weak slot & filler structure it is not universally acceptable i.e. the representation varies person to person.

The sol^n for the above problem is to use a strong slot & filler structure where activities used for repr^n of knowledge are from finite & predefined set. The advantage of using conceptual dependency as knowledge repr^n technique is:-

1. Simpler methods to get inference stored knowledge base.
2. The repr^n is independent of the lang in which original knowledge is given, that's why this technique is universally acceptable technique.

The actions used in C.D. are as follows:-

ATRANS :- Transfer of an abstract relationship for example give and take.

PTRANS :- Transfer of physical location of an object.
ex. go, come

PROPEL :- App^n of physical force to the object for ex. push, pull etc.

MOVE :- Movement of body part by its owner. ex. kick

GRASP :- Grasping or holding an object by an actor.
ex. Hold clutch

INGEST :- Ingestion of an object by any living thing.
ex. Eat, Drink.

EXPEL :- Expulsion of something out from the body.
ex. Cry

MTRANS :- Transfer of mental info.
ex. Teaching, learning etc.

MBUILD :- Building new info. out of old info. inference or decide

SPEAK :- Production of sound

ATTEND :- Focusing of sense organ towards particular activity.
ex - Listen, View

Q. John pushed the cart.

A. John ⟺ P → PROPEL ← O ◎ cart

; it is past activity

F- for future activity but nothing for present

Q. 1. I gave book to a man

2. John come tomorrow.

A. 2. Tomarrow

↓

John ⟺ PTRANS

1.

I ~~John~~ ⟺ P ATRANS ← O Book

↑

to ↓ from

man I

Q. John ~~I~~ took a ~~big~~ bicycle from Bill ~~man~~

A.

John ⟺ P ATRANS ← O Bicycle ← from Bill

→ John

to

Q. John went down Town to deposit money in the Bank.

A.

John ⟺ P PTRANS ← to Town

← from post

John ⟺ ATRANS ← O Money ← to Bank

← from John

Q. John begged ~~Many~~ for a pencil.

A.

John ~~Many~~ P ⟺ MTRANS ←

↑

John ⟺ P ATRANS ← pencil ← To

← from

Q. 1. John slapped Bill

2. Bill drank Coke.

A. 1.

John ⟺ P MOVE ← ~~hand~~ → Bill

← from John

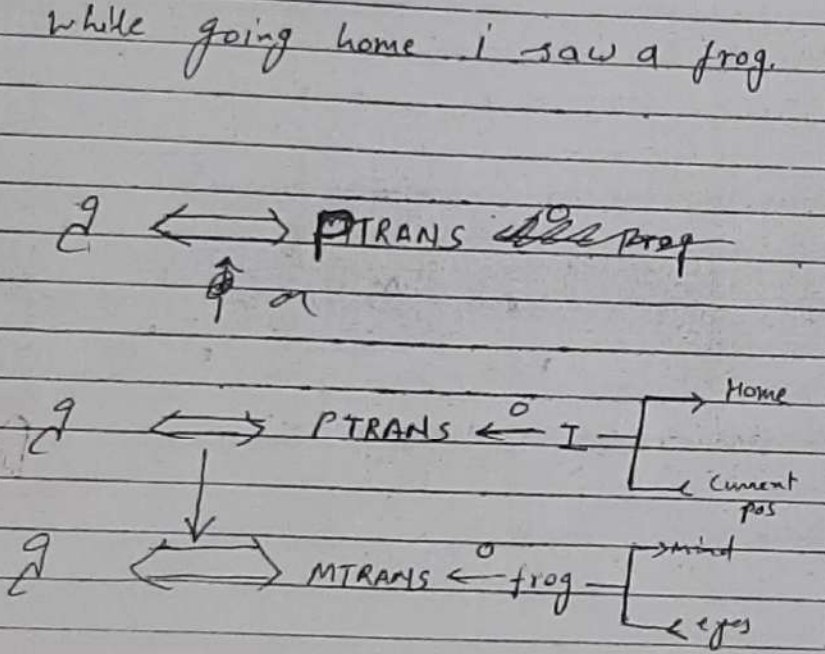
John ⟺ PROPEL ⟵ Bill

2.

Bill ⟺ INGEST ⟵ Coke

Q. While going home I saw a frog.

Ans



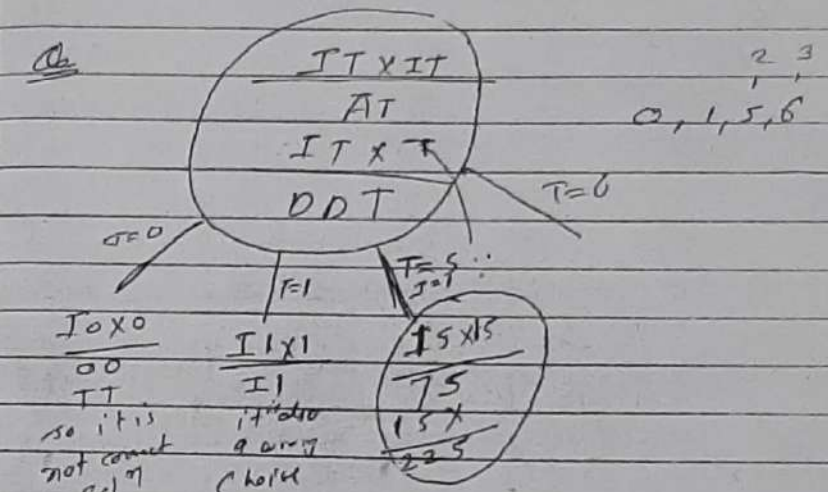
Constraints Satisfaction Problem :→

(Cryptoarithmatic Problem)

Q. SEND + MORE = MONEY

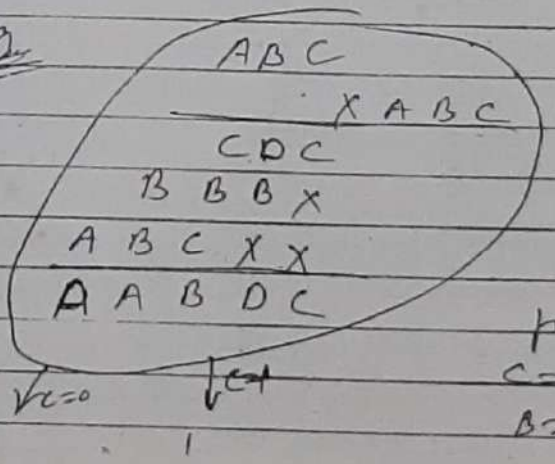
ROADS + CROSS = DANGER

Q.

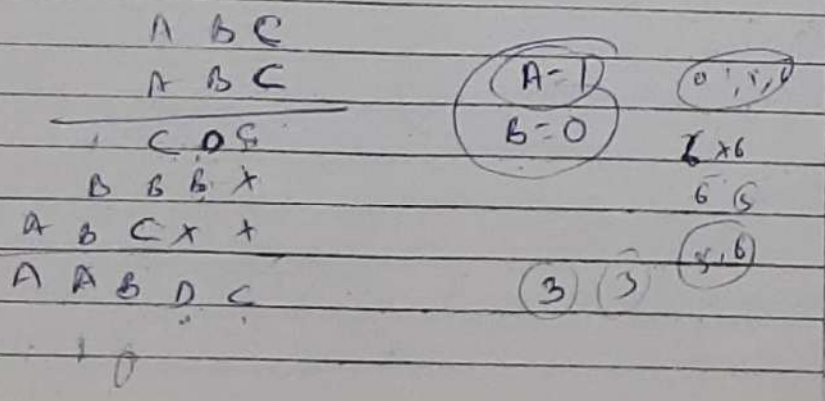


Sol^n T=5
A=7
I=1
D=2

Q.



here we can predict
C=1,6
B=0



FUZZY SET THEORY :-

$A = \{a, b, c, d\}$

$A = \{(a, 0.6), (b, 0.1), (c, 0.0), (d, 1.0)\}$
(Tall) ↑ Membership Chart func^n

Short $= \{(a, 0.1), (b, 0.7), (c, 1.0)\}$
↓ ∵ it may be other human also

Here probability (Tall) + prob(short) may not be 1.

~~Any element of fuzzy set is repeat~~

FUZZY SET THEORY :-

In conventional set theory only 2-value logic is used ie either any element is the member of given set (ie probability 1) or it is not member of the set (ie prob = 0).

In real life there are large no of app^n's / situations where partial membership of elements for a particular set is to be represented. The conventional set theory & Probability theory does not support such partial membership. Fuzzy theory is the sol^n to handle such type of situations where partial membership can also be considered eg.

Ex. Set of tall people & set of short people.

In conventional set theory
Tall $= \{a, b, c, d\}$
Short $= \{e, f, g\}$ ie here these two sets are disjoint sets, because it is assumed that either the person is tall or short.

In real life tall is a relative word (not absolute) & the def^n of tall varies person-to-person. Hence there may be no of people who are considered both tall short & may be something else in public opinion. To handle such type of sit^n the fuzzy repr^n of such type of sit^n will be -

Tall $= \{(a, 0.8), (b, 0.3), (c, 0.6), (d, 0.4), (e, 0.0), (f, 0.7)\}$

Short $= \{(b, 0.3), (c, 0.4), (d, 0.5), (e, 0.2), (f, 0.1)\}$

As shown in the above sets each element of set is represented by a

pair (

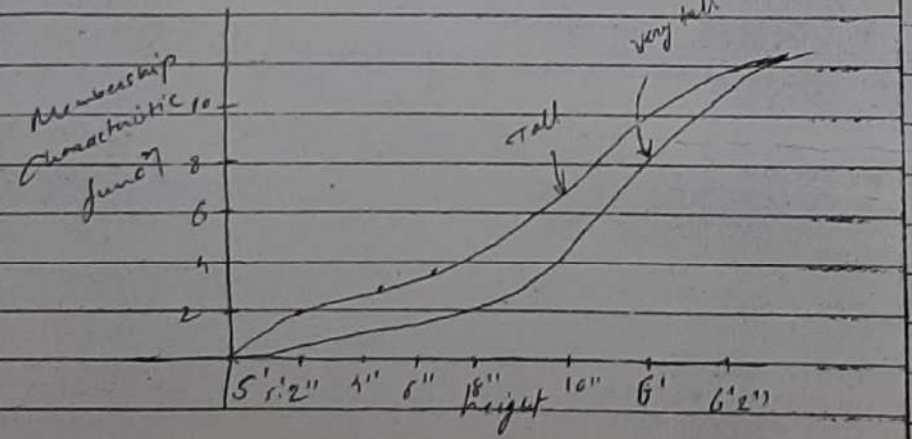
In set tall (B, 0.5) means 50% opinion is that B is tall in set short (B, 0.3) means 30% of people are saying that B is tall remaining 0.2 people may have not given any opinion or they may consider B as a member of any other set (middle set).

Fuzzy Curve :-

for short, tall & very tall

Tall = {(5', 0.0), (5'2", 0.2), (5'4"), ---, (6,2", 1.0)}

Curve



square & for V.V. it will ---

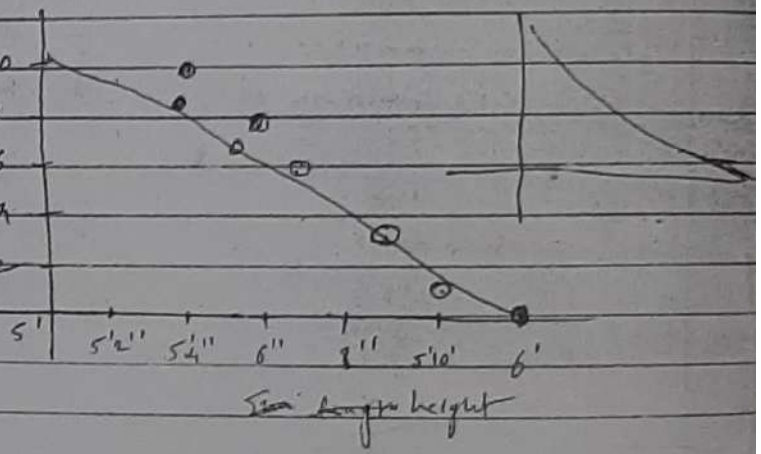
Very tall = {(5', 0), (5'2", 0.04), (5'4", 0.1), (5'6", 0.33) ---
--- (6', 0.82)

Q. Make curve for short & Very short (

Ans

Short = {(3', 1.0), (3'2", 0.8), (3'4", 0.7), (

Short = {(5', 1.0), (5'2", 0.8), (5'4", 0.5), (5'6", 0.30)
(5'8", 0.2), (6', 0.0)}



Natural Language Processing :-

Converting any natural lang. statement into M/c language instr's (High level)

Phases or steps For Translation :-
(Steps to translate, N.L. para into M/c lang)

1. Morphological Analysis :- It is similar to lexical analyser of compiler. Its basic duty is (a) Consider paragraph as a collⁿ of word
(b) Verify the validity of each & every word
(c) For each & every word pass token & attributes to next phase.

2. Syntatic Analysis :- Here paragraph is considered as collⁿ of sentences and each sentence is verified as per the grammar rules of the nat. lang.

3. Simantic Analysis :- The purpose of this is to check the meaningfulness of each and every sentence.

eg: 1. This is a dog.

2. This dog is

4. Discource integration :- In natural lang the meaning of a given sentence may be based on previous sentence for ex. (i) Vijay is SATI student
(ii) He is intelligent.
(iii) His college is situated at vidisha.

The purpose of ~~this~~ discource integration is to get correct meaning of the terms used in the statements.

5. Pragmatic Analysis :- In this it is possible to write phrases and this phrases and sentences don't have any direct meaning but they refer to any other sentence. The purpose of pragmatic analysis is to get actual meaning of the sentence.

CFGs For Natural Lang. process :→

$S \longrightarrow N_p V_p$ (Noun phrase & Verb phrase)

$N_p \longrightarrow$ (the) $N_{p_1} / P_r / N_{p_1} / P_N$ (A)

$N_{p_1} \longrightarrow Adj\ N$

$Adj \longrightarrow \epsilon / Ad\ Adj$

$V_p \longrightarrow V / V N_p$

$N \longrightarrow$ file / printer / dog

(Proper Noun) $P_N \longrightarrow$ John / Bill / Amit

$P_r \longrightarrow$ I

$Ad_j \longrightarrow$ Short / tall / good (extra) / old

$V \longrightarrow$ Created / printed / Bite

$A \longrightarrow$ the / a / an

Q 1 Parse the following sentences :—

1. Bill printed the file.
2. Sumit is a good boy.

Ans

Q 2: The dog bites (bites) the oldman.

Ans 1 1. Bill printed the file

(P_N V N)

Generate by LMD :⇒

$S \longrightarrow N_p V_p$

$S \longrightarrow P_N V_p$

$S \longrightarrow Bill\ V_p$

$S \longrightarrow Bill\ V N_p$

$S \longrightarrow Bill\ printed\ N_p$

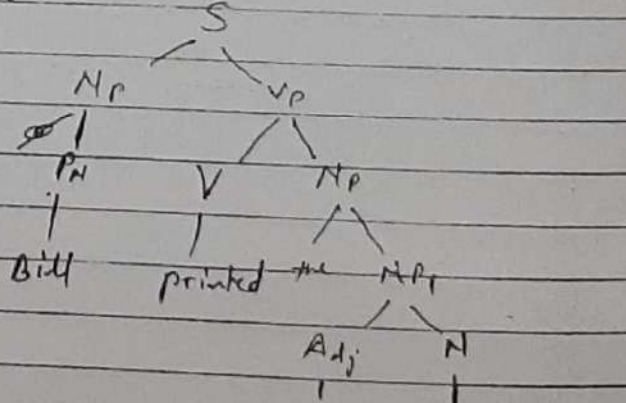
$S \longrightarrow Bill\ printed\ the\ N_{p_1}$

$S \longrightarrow Bill\ printed\ the\ Adj\ N$

$S \longrightarrow$ —"— the N

$S \longrightarrow$ —"— the file

parse tree :-



S → The dog V N/p

S → The dog bite N/p

S → The dog bites A N/p

S → The dog bites Adj N

S → The dog bites Ad Adj N

S → The dog bites old N

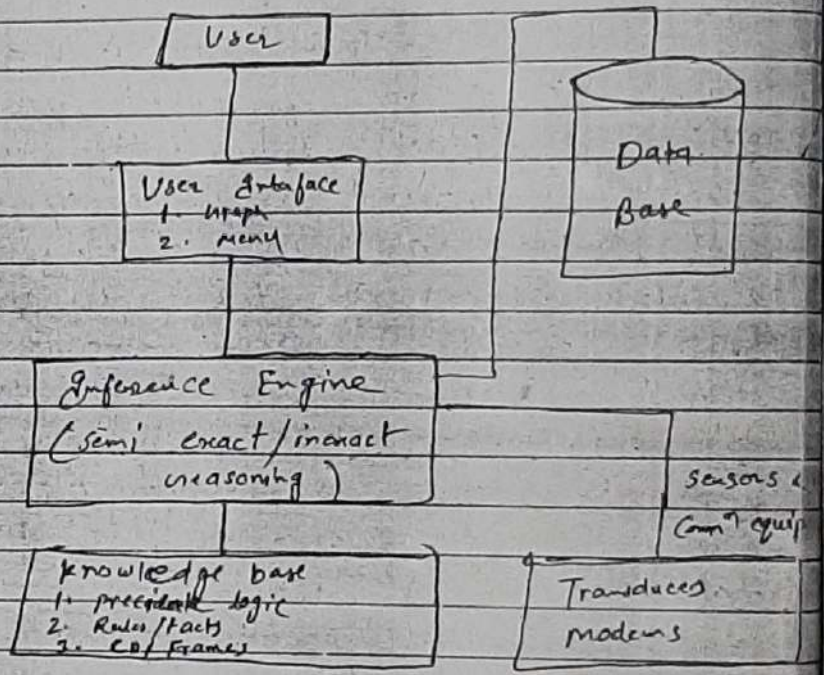
S → The dog bites old man.

Parse tree:→

(Theory)

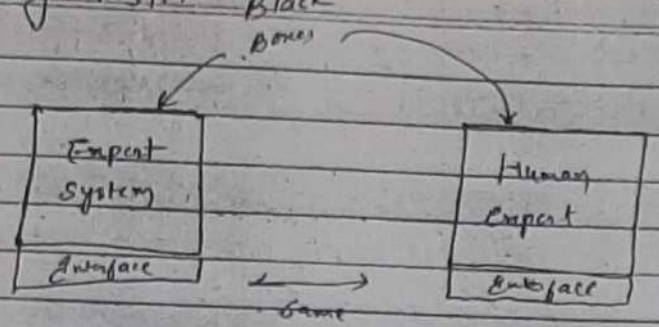
Expert Systems:-

It is generally used by experts. Data is fed to the exp^t system, which processes the data & do reasoning and then give the by and the results it will give the advice that what to do.



In Exact reasoning :→ means the decision reasoning has not been proved completely but still the decision is taken. Ex. If doctor gets few symptoms of MP parasite & the weather then inspite the proof he takes says U have the disease.

Turing Test:-



Turing Test:-

The Turing Test is used to verify the goodness of expert systems. It is a black box testing mechanism that is internal components of the exp^t systems are not tested. Here 2 - black boxes are taken in one box human expert is present and in another expert system software. The way of interaction that is interface for both the boxes is same. Now any no. of queries can be asked to the box where we don't know the position of human exp^t & to exp^t system soft. and after asking many queries it is not possible to diff. where exp^t system & where human exp^t is present then exp^t system is considered to be perfect and this type of testing mechanism is known as turing test.

Cell → the utility to

Expert System Cell:-

A program used to build expert systems is usually called expert system cell. It works very similar to word processor which is used to create text file or letter, Expert S. C. is used to create expert system.

Functions performed by expert system cell:-

(1) Assists in building the knowledge base by allowing a developer to insert knowledge into knowledge representation structure.
(2) Provides methods of inference or reasoning that can do reasoning based on the info. available in knowledge base.
(3) Provide an interface that allow the user to setup reasoning tasks and query the system about its reasoning strategy.

What is imp. about E.S. Cell:-

The guidlines for evaluation of expert system cell are as follows:-

(1) Expressing knowledge:-
   (a) Descriptive power
   (b) Ease of translation
   (c) Readability

(2) Organizing & Displaying Knowledge
(4) Reasoning with & validating k

(3) Reasoning

(4) Operating a knowledge

(5) Ability to integrate conventional software with expert system.

Mycin :→

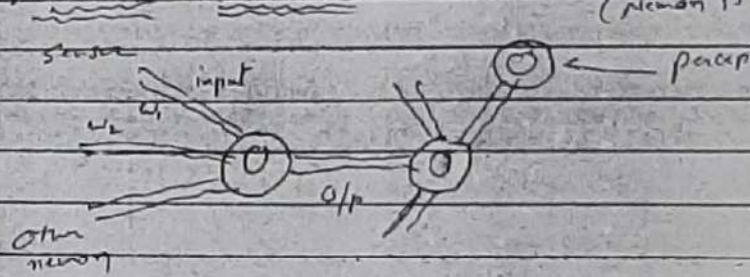
Mycin was one first successful expert system to demonstrate the feasibility of developing intelligible progs that can do activities similar to human experts. Mycin was designed to interact with a physician, collect all the relavent info about the patient and examine this info for the evidence on which to base of diagnosis and recomand a treatment.

Mycin is a rule based system of about 500 rules. In analysing each case mycin answers four basic questions
(1) Is the infection significant.
(2) What is the identity of the organism causing the infection.
(3) What are the potentially useful drugs.
(4) Which drug is best suited for patient

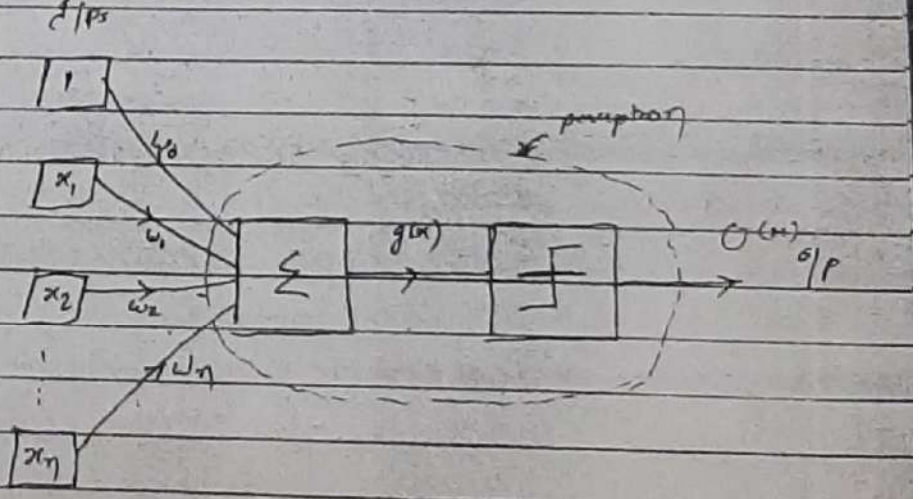
The basic opn of Mycin is controlled by backward chaining Rule interpreter. The rules are written with the use of certainity factors. This will help Mycin to do interact reasoning. The certainity value may range from -1 to +1 here '-1' indicates that complete confedence that the part rule/prop. is false. '+1' indicates the rule — true.

(In book)

Neural Networks:→



Perceptron :—

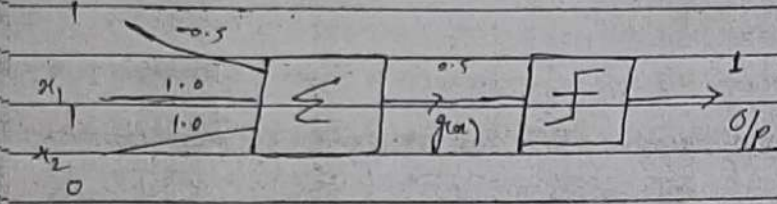


Neural N/W is a complex n/w of perceptrons

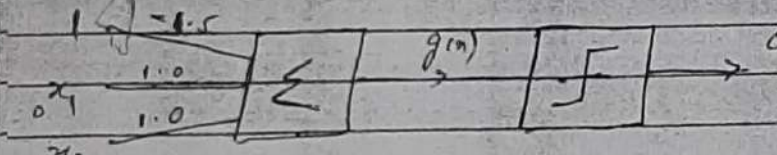
$$g(x) = \sum_{i=0}^{n} w_i x_i$$

$$O(x) = \begin{cases} 1 & \text{if } g(x) > 0 \\ 0 & \text{if } g(x) < 0 \end{cases}$$

Ex: OR - Gate for 2 i/p.

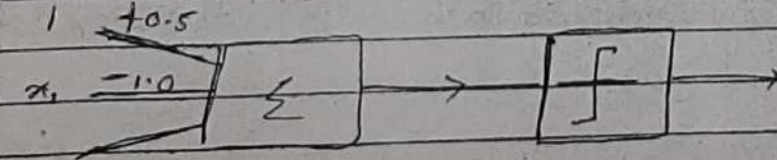


Ex: AND - Gate



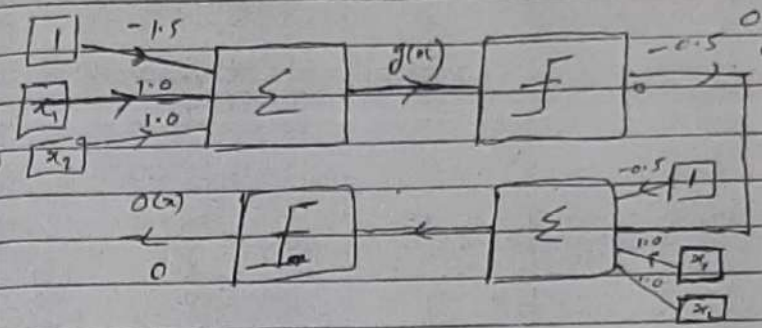
Here

Ex: EX-OR NOT - Gate.



Ex: EXOR Gate :→

Here we need two p



LEARNING :-

As human can grow his knowledge base by watching or doing new thing called learning exactly & AI system learning is provided.

AI system should remain intelligent for long period of time is an essential requirement. To make it possible there must be good knowledge aquiciti techniques (Learning tech.) to grow its knowledge base. There are five strategies or learning methods.

(1) Rote learning.
(2) Direct instn
(3) Analogy.
(4) Induction
(5) Deduction