1 Define Context Sensitive language

A: The Context Sensitive grammars are used to represent context sensitive languages

The context sensitive grammar is an unrestricted grammars in which all the productions are of form

$$\alpha \longrightarrow \beta$$

where $\alpha, \beta \in (V \cup T)^+$ and

$$|\alpha| \leq |\beta|$$

→ Where α, β are strings of Non-terminals & terminals

→ Here the length of L.H.S is restricted to length of R.H.S in CSG

2) Define Decidability?

A: A problem is said to be decidable if we can construct a TM which will in finite amount of time for every input and gives answer Yes (or) no.

A decidable problem has an algorithm to determine the answer for given input

3 Define Recursively enumerable language

A: An Regular language can be accepted (or) recognized by turing machine which means it will enter into final state for the strings of the language and may (or) may not enter into rejecting state for the strings which are not part of the language.

It means TM can loop forever for the strings which are not part of the language.

4 Define Undecidability?

A: A problem is undecidable if there is no turing machine which will halt in finite amount of time to give answer Yes (or) no.

An undecidable problem has no algorithm to determine the answes for given input.

5. Define Recursive languages

A: Recursive languages:

A language 'L' is said to Recursive if there exists a Turing Machine which will accept all the strings not in 'L'.

Scanned with CamScanner

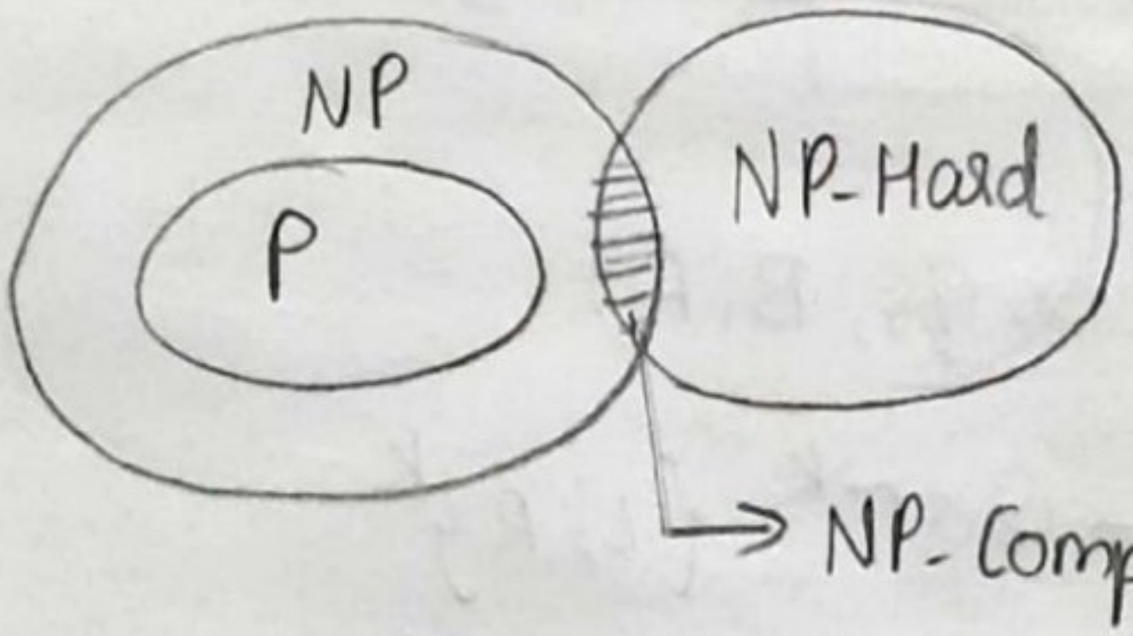
→ This turing machine will Halt every time and give an answer (accepted (or) rejected) for each and every string input.

6 Define NP- complete class

Ar NP- Complete class:

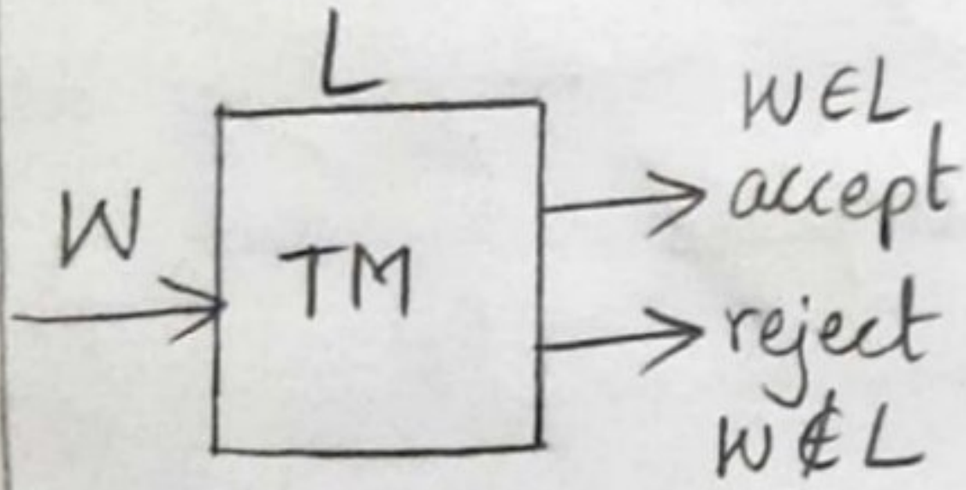
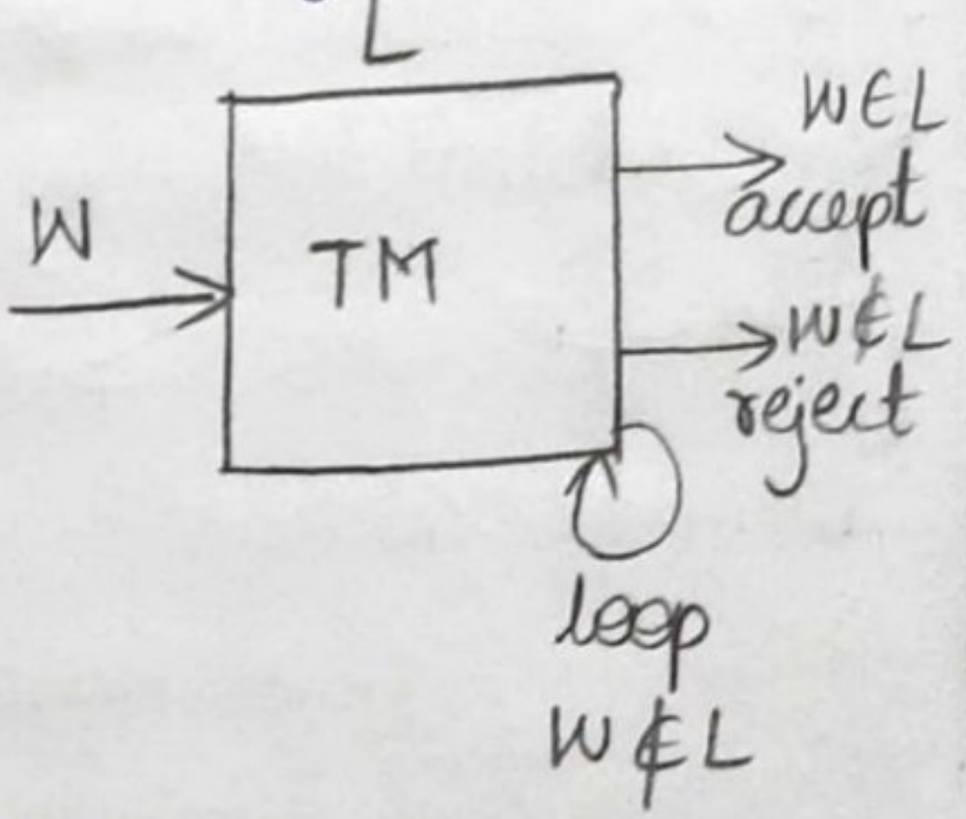
If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time soluable. These problems are called NP- Complete. The phenomenon of NP- completeness is important for both theoretical and practical reasons

NP- Complete is also called as Decision problem



7 Distinguish Recursive languages and recursively enumerable languages

Ans

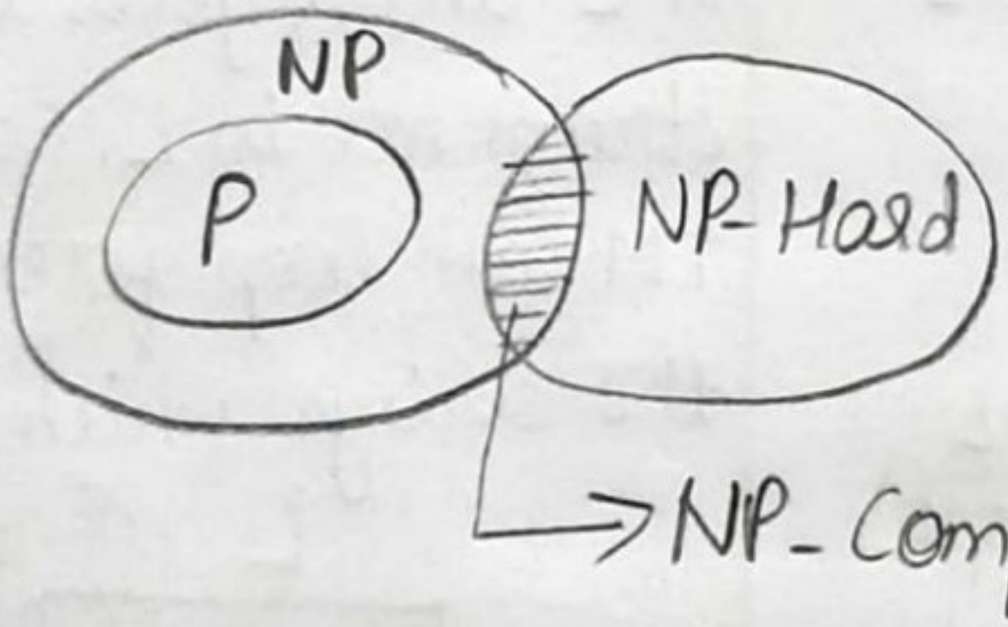
Recursive language	Recursively Enumerable lang
1) A Language 'L' is said to be recursive if there exists a TM which will accepts all the strings in 'L' and reject all the strings not in L. 	1) A Language 'L' is said to be recursive enumerable if there exists a TM which will accepts all the strings in 'L' and rejects the strings not in L. Sometims TM can loop forever for the strings which are not 
2) Recursive lang is decidable language	2) Recursive enumerable language is partially Decidable language

8 Define NP-Hard class

Ans NP-Hard class:

A problem than is NP-Hard has property that all problem that are in NP can be reduced in polynomial time to it.

→ NP-Hard problems are atleast as hard as NP-complete problems



9) Define PCP

A:- Post Correspondence Problem is an undecidable decision problem that was introduced by Emil Post in 1946

Scanned with CamScanner

The PCP problem consists of two lists of strings that are of equal length over the input Σ.

The two lists are $A = x_1, x_2, x_3 \text{-----} x_n$

and $B = y_1, y_2, y_3, \text{-----} y_n$

then there exists a non empty set of integers $i_1, i_2, i_3, \text{-----}$ in such that

$x_1, x_2, \text{----} x_n = y_1, y_2 \text{----} y_n$

$\Rightarrow$ To solve the PCP we try all the combinations of $i_1, i_2, i_3 \text{---}$ in to find the $x_i = y_i$ then we say that the PCP has a solution.

16) Compare P, NP problems

A:-

P Problems	NP Problems
1) P problems are set of problems which can be solved in polynomial time by deterministic algorithms	1) NP problems are the problems which can be solved in non-deterministic polynomial time.
2) P problems can be solved and verified in polynomial time	2) Solutions to NP problems can not be obtained in polynomial time, but if the solution is given, it can be verified in polynomial time.
3) P problems are subset of NP problems	3) NP problems are superset of P problems
4) All P problems are deterministic in nature	4) All NP-problems are non-deterministic in nature
5) Example: Selection Sort linear search	5) Example: TSP, knapsack problem

11 a) Discuss in brief about NP-Hard problems

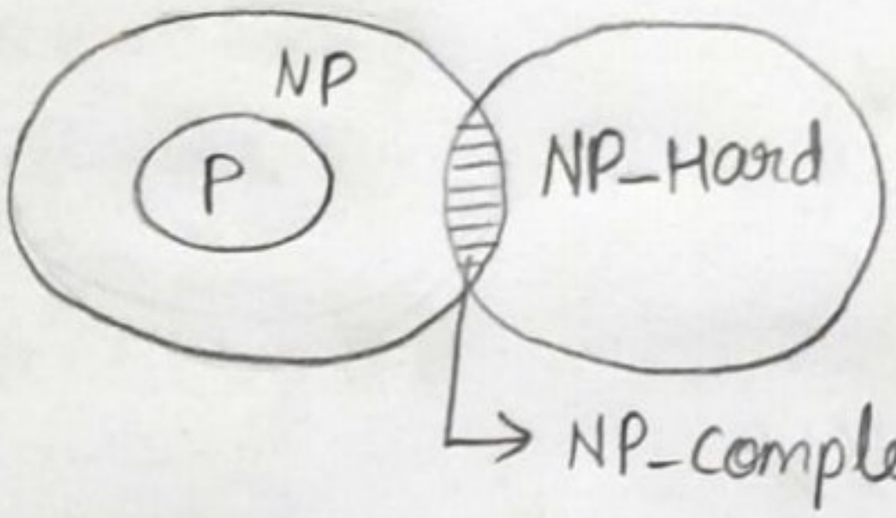
A:- **NP Hard Problem:**

A Problem that is NP Hard has property that all problem that are in NP can be reduced in polynomial time to it.

→ NP Hard problems are atleast as hard as NP-complete problems

→ NP Complete problems can be said to soluable in polynomial time if and only if P = NP.

→ NP-Hard problems can't be solved in polynomial time unless P = NP



Ex:- Integer linear programming is NP-Hard.

→ NP-Hard problem is called as Optimization problem.

b) Explain about the Decidability and undecidability problems

A:- **Decidability**: A problem is decidable if we can construct a TM which will in finite

amount of time for every input and give answer as Yes (or) No.

A Decidable problem has an algorithm to determine the answer for given input.

Example:

1) Equivalence of two regular languages:

Given two regular languages there is an algorithm and Turing machine to decide whether two regular languages are equal (or) not.

i) Finiteness of regular language:

Given a R.L, there is an algorithm and TM to decide whether R.L is finite (or) not.

ii) Emptiness of CFL:

Given a CFL, there is an algorithm whether CFL is empty (or) not.

Undecidability:

A problem is undecidable if there is no turing machine which will halt in finite amount of time to give answer as Yes (or) no.

An Undecidable problem has no algorithm to determine the answer for given input.

Example:

1) Ambiguity of CFL: Given a CFL, there is no TM which will always halt in finite amount of time and give answer whether language is

ambiguous (or) not

i) Equivalence of 2 CFL's: Given CFL, there is no TM which will always halt in finite amount of time and give answer whether 2 CFL's are equal (or) not.

ii) Everything (or) Completeness of CFG: Given a CFG and input alphabet, whether CFG will generate all possible strings of input alphabets (Σ^*) is undecidable.

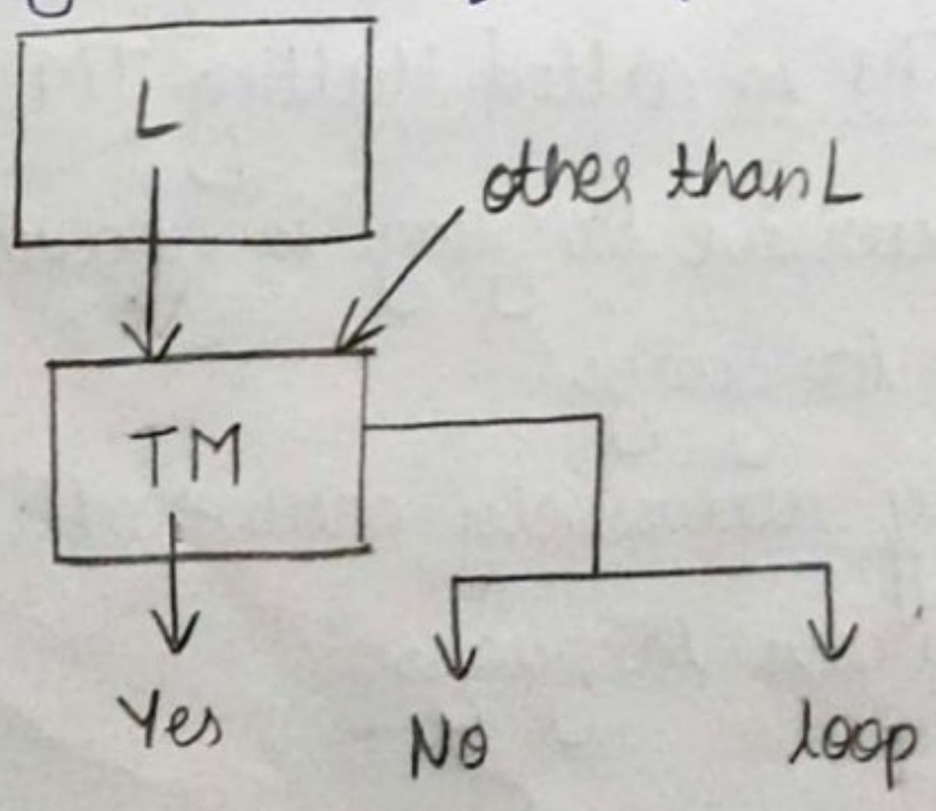
12 a) Explain about recursively enumerable language

A: A language 'L' is said to be a recursively enumerable language if there exists a turing machine which will accepts for all the input strings which are in 'L'.

Here Σ → set of input symbols

Σ^* → set of string

Language → set of all possible strings



→ If the set of strings are present in language, all that strings are accepted by TM, then set of strings
i.e. Language is recursively Enumerable

→ If the string is not present in the language then
i) The TM HALT in any non final state and string is not accepted by TM
ii) The TM may goto infinite loop.

→ To overcome the problem of infinite loop in recursively enumerable language. restrict the TM in such a way that the string is present in TM then string is accepted and halts at final state

→ If the string is not present in TM, then the string is not accepted by TM & Halts at some other state (but not going to infinite loop)
Such a TM is called Halting TM.

→ Every recursive language is recursively enumerable language

→ But every recursively enumerable language is not recursive language

b) Compare P, NP problems

A:

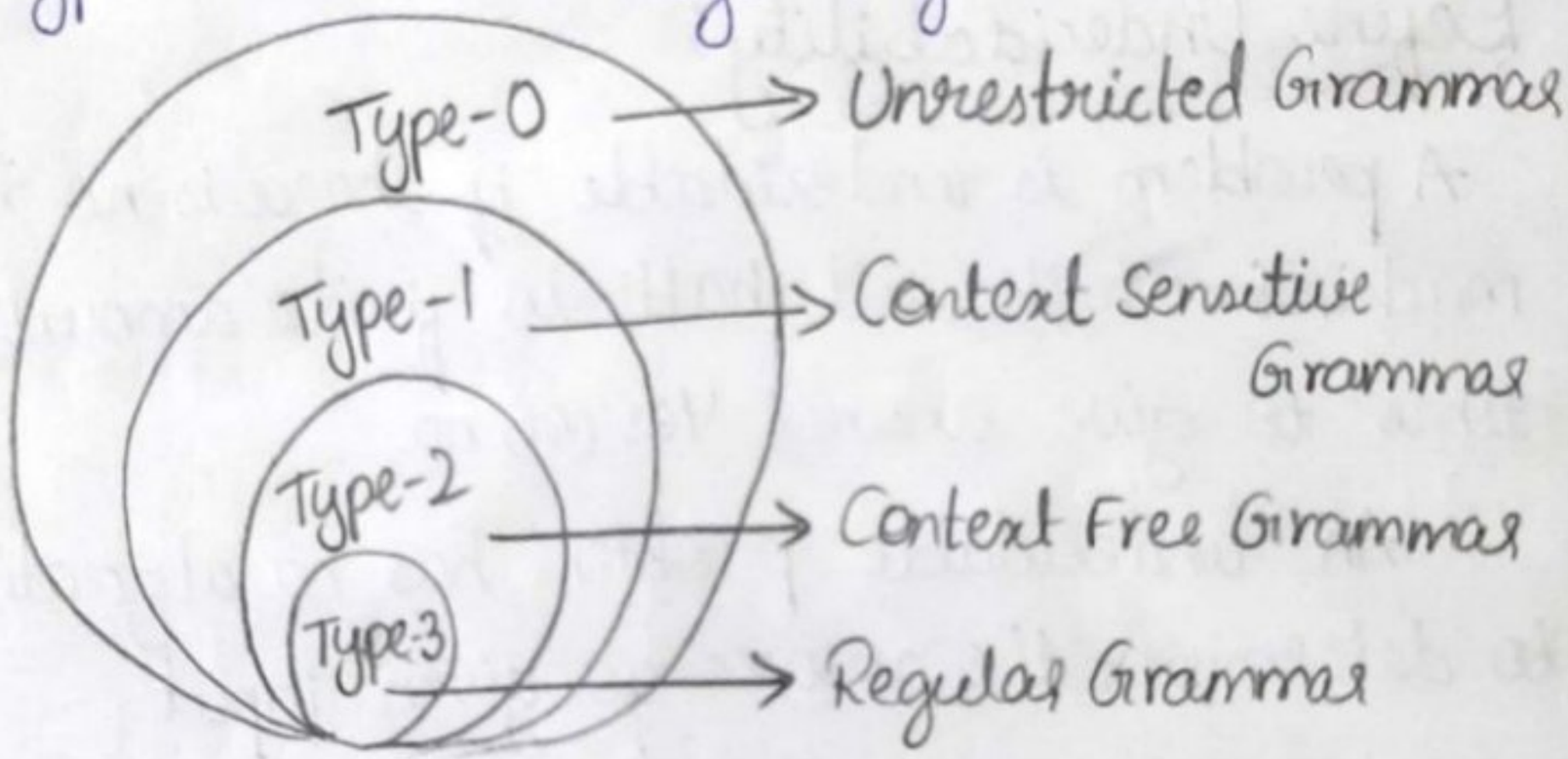
P Problems	NP Problems
1) P problems are set of problems which can be solved in polynomial time by deterministic algorithms	1) NP problems are the problems which can be solved in non-deterministic polynomial time.
2) P problems can be solved and verified in polynomial time	2) Solutions to NP problems can not be obtained in polynomial time, but if the solution is given, it can be verified in polynomial time.
3) P problems are subset of NP problems	3) NP problems are superset of P problems
4) All P problems are deterministic in nature	4) All NP-problems are non-deterministic in nature
5) Example: Selection Sort linear search	5) Example: TSP, knapsack problem

13 a) Explain the chomsky's hierarchy of languages

A: Chomsky hierarchy represents the class of languages that are accepted by the different machine.

The category of language is

1) Type 0 known as unrestricted grammar
2) Type 1 known as Context sensitive Grammas
3) Type 2 known as Context Free Grammar
4) Type 3 known as Regular grammar



Type 0: Unrestricted Grammar:

It includes all formal Grammar. It is recognized by turing machine, these languages are also known as recursive enumerable language.

Grammar production in the form of

$\alpha \rightarrow \beta$

where α is $(VUT)^+ V (VUT)^+$

V → Variables

T → Terminals

Type-1 Grammar: Context Sensitive Grammars:

This grammar generates the context sensitive language. This language accepted by linear bounded Automata

Production: $\alpha \longrightarrow \beta$

where $\alpha, \beta \in (V \cup T)^+$ and

$|\alpha| \leq |\beta|$

Type2 Grammar: Context Free Grammar:

This Grammar generates context Free language. This language accepted by PDA

production: $A \longrightarrow \alpha$

where $\alpha = (V \cup T)^*$ and $A \in V$

Type-3 Grammar: Regular Grammar

This grammar generates regular language. This language accepted by finite automata.

b) What is Post Correspondance problem. Verify whether the following PCP has a solution (or) not?

$A = \{ba, ab, a, baa, b\}$, $B = \{bab, baa, ba, a, aba\}$

A: Post Correspondence Problem:

The PCP problem is an undecidable decision problem that was introduced by Emil Post in 1946

The PCP problem consists of two lists of strings

that are of equal length over the input Σ.

The two lists are $A = x_1, x_2, x_3, \ldots\ldots x_n$

and $B = y_1, y_2, y_3, \ldots\ldots y_n$

then there exists a non empty set of integers $i_1, i_2, i_3, \ldots\ldots$ in such that

$$x_1, x_2, \ldots\ldots x_n = y_1, y_2, \ldots\ldots y_n$$

→ To solve the PCP we try all the combinations of $i_1, i_2, i_3, \ldots\ldots$ in to find the $x_i = y_i$ then we say that the PCP has a solution

$A = \{ ba, ab, a, baa, b \}$
$\quad\quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5$

$B = \{ bab, baa, ba, a, aba \}$
$\quad\quad y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5$

then

$x_1 x_5 x_2 x_3 x_4 x_4 x_3 x_4 = y_1 y_5 y_2 y_3 y_4 y_4 y_3 y_4$

$ba\,ba\,ba\,baa\,baaa\,baa = ba\,ba\,ba\,baa\,baaa\,baa$

Hence the solution is 15234434

14 Define the following and give examples

i) P-type

ii) NP-Hard

A: i) P-type:

P problems are set of problems which can be solved in polynomial time by deterministic algorithms

→ A deterministic machine at each time executes an instruction, depending on instruction, it then goes to next state which is unique.

A language L is said to be in class P if there exists a (deterministic) TM.

If the turing machine M has some polynomial P(n) and the machine M never makes more than P(n) moves when an input of length n is given to M then such a TM is said to polynomial time TM

Example: Transitive closure, Kruskal's algorithm for minimum spanning tree.

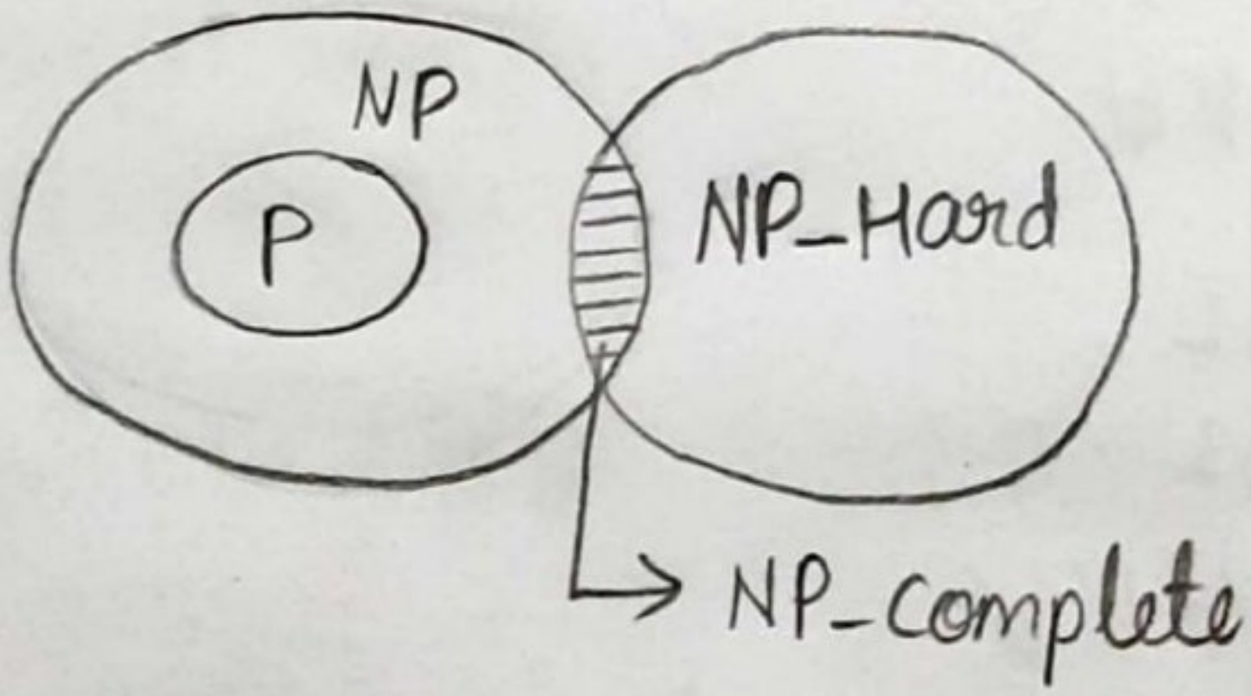
NP Hard Problem:

A Problem that is NP Hard has property that all problem that are in NP can be reduced in polynomial time to it.

→ NP Hard problems are atleast as hard as NP-complete problems

→ NP Complete problems can be said to soluable in polynomial time if and only if P = NP.

→ NP-Hard problems can't be solved in polynomial time unless P = NP



Ex: Integer linear programming is NP-Hard.

→ NP-Hard problem is called as Optimization problem.

15 Define Post Correspondence problem (PCP).
Explain in brief about PCP with an example

Ans: Post Correspondence problem:

The PCP problem is an undecidable decision problem that are introduced by Emil Post in 1946.

The PCP problem consists of two lists of strings that are of equal length over the input Σ.

The two lists are $A = x_1, x_2, x_3 \ldots\ldots\ldots x_n$

and $B = y_1, y_2, y_3, \ldots\ldots y_n$

then there exists a non empty set of integers $i_1, i_2, i_3, \ldots\ldots$ in such that

$$x_1, x_2, \ldots\ldots x_n = y_1, y_2 \ldots\ldots y_n$$

$\Rightarrow$ To solve the PCP we try all the combinations of $i_1, i_2, i_3 \ldots\ldots$ in to find the $x_i = y_i$ then we say that the PCP has a solution.

Example: $x = \{11, 100, 111\}$

$y = \{111, 001, 11\}$ has a solution.

i	x_i	y_i
1	11	111
2	100	001
3	11	11

$x_1 x_2 x_3 = y_1 y_2 y_3$

1110011 = 11100111

Solution sequence is $\Rightarrow$ 123.

16 Explain in detail about NP-complete and NP-Hard problems

A: Definition of NP-Completeness:

A language B is NP-complete if it satisfies two conditions

* B is in NP
* Every A in NP is polynomial time reduciable to B

→ If a language satisfies the second property, but not necessarily the first one, the language B is known as NP-Hard. Informally, a search problem B is NP-Hard if there exist some NP-Complete problem A that turing reduces to B.

→ The problem is in NP-Hard cannot be solved in polynomial time, untill P = NP.

→ If a problem is proved to the NP-Complete there is no need to waste time on trying to find an efficient algorithm for it. Instead, we can focus on design approximation algorithm

Ex: Satisfiability problem

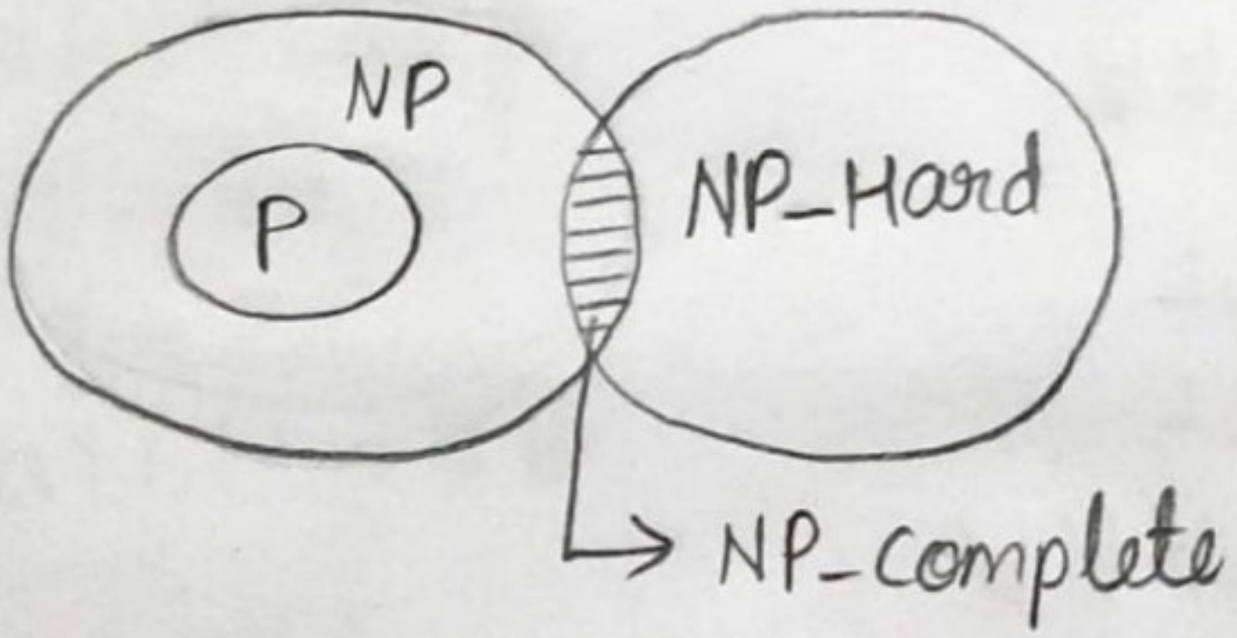
NP Hard Problem:

A Problem that is NP Hard has property that all problem that are in NP can be reduced in polynomial time to it.

→ NP Hard problems are atleast as hard as NP-complete problems

→ NP Complete problems can be said to soluable in polynomial time if and only if P = NP.

→ NP-Hard problems can't be solved in polynomial time unless P = NP

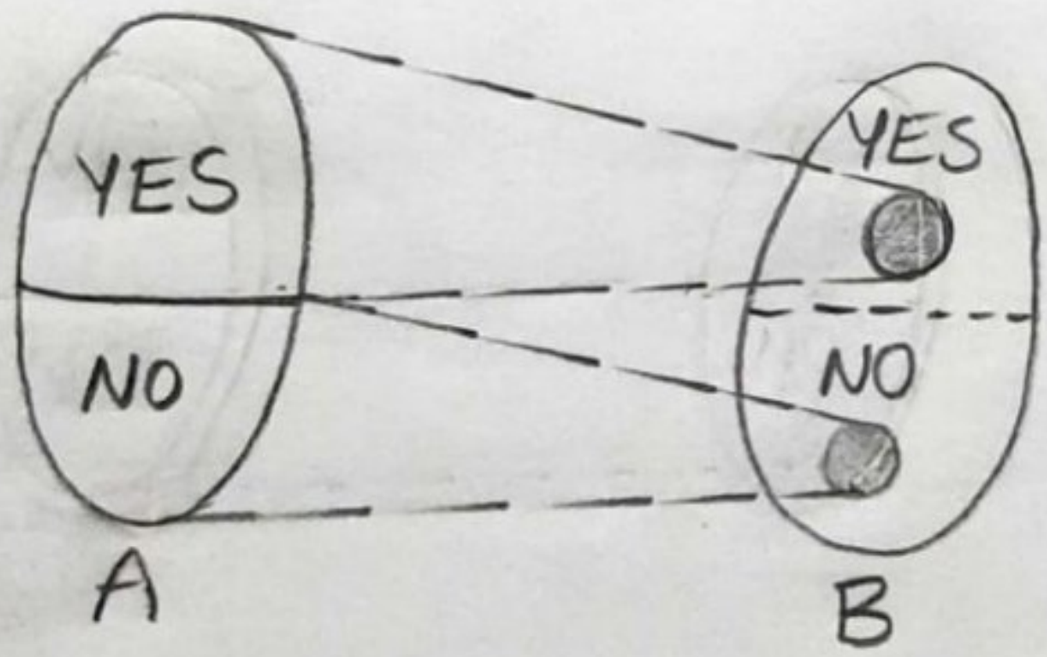


Ex: Integer linear programming is NP-Hard.

→ NP-Hard problem is called as Optimization problem.

17 What is Turing Reducible problem? Give an example.

A:- Reduction is a technique in which if a problem A is reduced to problem B then any solution of B solves A. In general, if we have an algorithm to convert some instance of problem A to some instance of problem A to some instance of problem B that have same answer here it is called A reduces to B



Definition of turing Reducibility:

Let A and B be the two sets such that A, $B \subseteq N$ of natural numbers. Then A is Turing reducible to B and denoted as $A \leq_T B$
If there is an oracle machine that computes the characteristics function of A when it is executed with oracle machine for B.

This is also called as A is B-recursive and B-computable. The oracle machine is an abstract machine used to study decision problem. It is also called as Turing machine with black box. We say that A is turing equivalent to B and write $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$

Some Properties:

1) Every set is Turing equivalent to its complement
2) Every computable set is Turing equivalent to every other computable set
3) If $A \leq_T B$ and $B \leq_T C$ then $A \leq_T C$

18 Define PCP problem? Does the PCP problem $P = \{(10,101)(011,11)(101,011)\}$ has a solution if so, give the solution

A: Post Correspondence problem:

The PCP problem is an undecidable decision problem that are introduced by Emil Post in 1946.

The PCP problem consists of two lists of strings that are of equal length over the input Σ.

The two lists are $A = x_1, x_2, x_3 \ldots\ldots x_n$

and $B = y_1, y_2, y_3, \ldots\ldots y_n$

then there exists a non empty set of integers $i_1, i_2, i_3, \ldots\ldots i_n$ such that

$$x_1, x_2, \ldots\ldots x_n = y_1, y_2 \ldots\ldots y_n$$

$\Rightarrow$ To solve the PCP we try all the combinations of $i_1, i_2, i_3 \ldots\ldots i_n$ to find the $x_i = y_i$; then we say that the PCP has a solution.

$P = \{(10, 101)\ (011, 11)\ (101, 011)\}$

1	10	101
2	011	11
3	101	011

$$\Rightarrow \frac{M}{N} = \frac{10}{101}\ \frac{011}{11}\ \frac{101}{011}$$

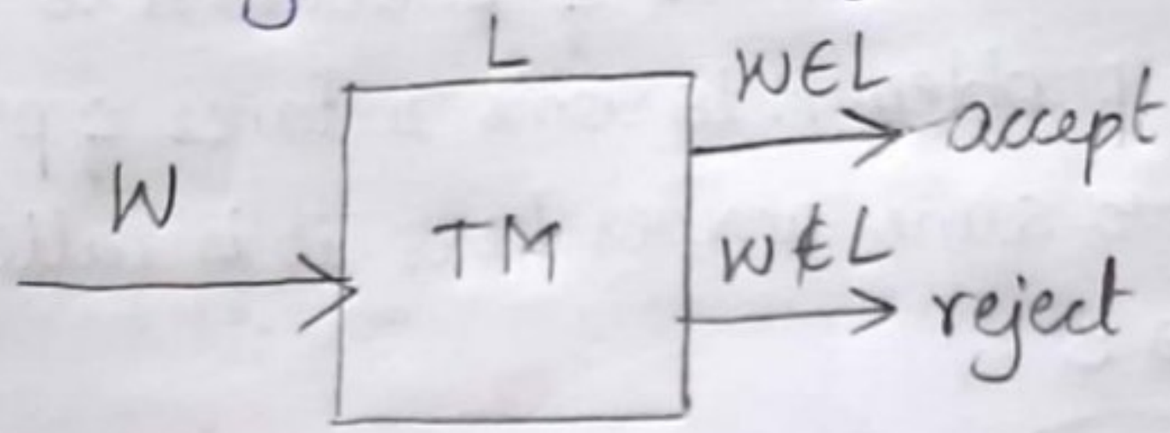
$$\frac{1010110}{10101111101}$$

numerator and denominator does not match
Then, It has no PCP Solution.

19. a) Write short notes on Recursive and recursively Enumerable languages.

Ans Recursive language:

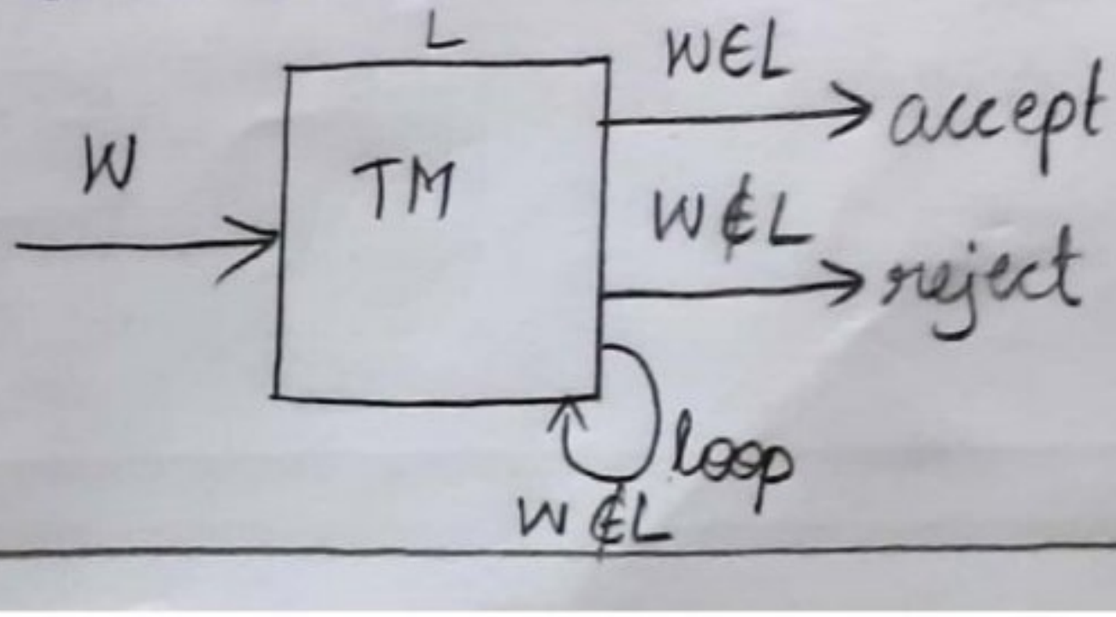
A language 'L' is said to be recursive if there exists a TM which will accepts all the strings in 'L' and reject all the strings not in L.



→ Recursive language is decidable language because a Turing machine can decide membership in those languages (It can either accept (or) reject a string)

Recursively enumerable languages:

A language 'L' is said to be recursive enumerable if there exists a TM which will accepts all the strings in 'L' and rejects the strings not in L Sometimes TM can loop forever for the strings which are not in 'L'.



→ Recursive enumerable language is partially Decidable language

→ Recursively Enumerable languages are also called Recognizable because a Turing Machine can recognise a string in the language (accept it) It might not be able to decide if a string is not in the language since the machine might loop for that input

b) Explain the following i) Decidability
ii) Undecidability

i) Decidability: A problem in decidable if we can construct a TM which will in finite

amount of time for every input and give answer as Yes (or) No.

A Decidable problem has an algorithm to determine the answer for given input.

-Example:

1) Equivalence of two regular languages:

Given two regular languages there is an algorithm and Turing machine to decide whether two regular languages are equal (or) not.

i) Finiteness of regular language:

Given a R.L, there is an algorithm and TM to decide whether R.L is finite (or) not.

ii) Emptiness of CFL:

Given a CFL, there is an algorithm whether CFL is empty (or) not.

iii) Undecidability:

A problem is undecidable if there is no Turing machine which will halt in finite amount of time to give answer as Yes (or) no.

An Undecidable problem has no algorithm to determine the answer for given input.

-Example:

1) Ambiguity of CFL: Given a CFL, there is no TM which will always halt in finite amount of time and give answer whether language is

ambiguous (or) not

i) Equivalence of 2 CFL's: Given CFL, there is no TM which will always halt in finite amount of time and give answer whether 2 CFL's are equal (or) not.

ii) Everything (or) Completeness of CFG: Given a CFG and input alphabet, whether CFG will generate all possible strings of input alphabets (Σ^*) is undecidable.

20. Define NP. Complete and NP-Hard Problems

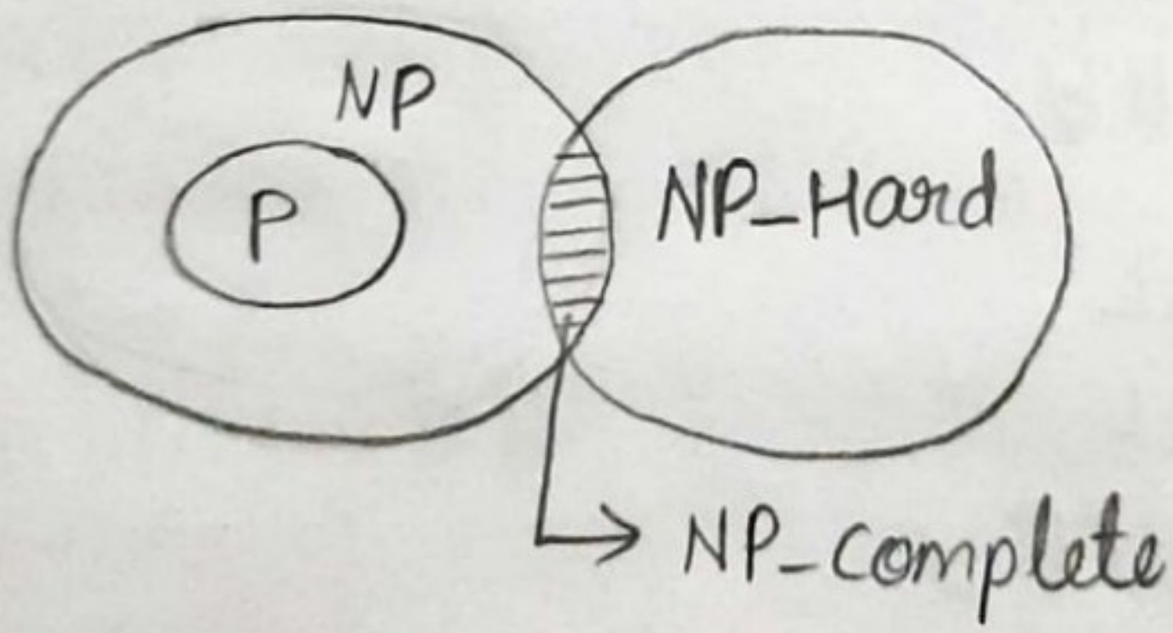
NP Hard Problem:

A Problem that is NP Hard has property that all problem that are in NP can be reduced in polynomial time to it.

→ NP Hard problems are atleast as hard as NP-complete problems

→ NP Complete problems can be said to soluable in polynomial time if and only if P = NP.

→ NP-Hard problems can't be solved in polynomial time unless P = NP



Ex: Integer linear programming is NP-Hard.

→ NP-Hard problem is called as Optimization problem.

Definition of NP-Completeness:

A language B is NP-complete if it satisfies two conditions

* B is in NP
* Every A in NP is polynomial time reduciable to B

→ If a language satisfies the second property, but not necessarily the first one, the language B is known as NP-Hard. Informally, a search problem B is NP-Hard if there exist some NP-Complete problem A that turing reduces to B.

→ The problem is in NP-Hard cannot be solved in polynomial time, untill P = NP.

→ If a problem is proved to the NP-Complete there is no need to waste time on trying to find an efficient algorithm for it. Instead, we can focus on design approximation algorithm

Ex: Satisfiability problem