UNIT VI COMPUTABILITY AND COMPLEXITY THEORY

COMPUTABILITY THEORY.

1. DECIDABLE PROBLEMS:

A problem is decidable if it has an algorithm or TM that can always halt and give a correct yes/no answer for any given input
i.e A language is decidable if it accepts strings in language and reject others.
The class of decidable language is also known as recursively enumerable or recursive language.

2. UNDECIDABLE PROBLEMS:

- A problem is undecidable if it has no algorithm or TM that can decide if arbitrary input belongs to language The machine might not halt and reject even if input not in language.
- The halting problem is classic example of undecidable problem. There is no general algorithm that can decide whether an arbitrary TM will halt on given input.

3. CHURCH - TURING THESIS:

- The church Turing Thesis is the hypothesis that anything ~~computable by a TM~~. Can be computable by a algorithm can be computable by a TM.
- It suggest that the informal notion of computability (such as algorithmic problem solving) is equivalent to what can be computed by a TM

RECURSIVELY ENUMERABLE PROBLEM (RE):

A language is RE if there exist a TM that will accept strings in language and may either reject or run forever for string not in RE language.
While RE problem is not ~~un~~decidable (since it may never halt for string outside language.) we can atleast verify whether a string is in the language by running the TM

SIMPLE UNDECIDABLE PROBLEM:

The halting problem itself is strong example of undecidable problem.

- Given a TM (M) an input w, determine whether M halts on input w.
- Its undecidable becoz no general algorithm exist that can solve this problem for all possible TM inputs.

COMPLEXITY CLASSES.

1.] TIME AND SPACE COMPLEXITY

- TIME: amount of computational time a problem requires relative to input size.
- SPACE: amount of ~~com~~ memory / tape space required relative to input size.

2.] CLASS.

i] class P:

It consist of problems which can be solved by a DTM (deterministic TM) in polynomial time.
i.e a problem can be solved efficiently.
example: Sorting O(nlogn)
shortest path.

ii] class NP:

It consist of decision problem where a solution if given can be ~~solved~~ verified in polynomial time
Although we might not know how to solve quickly if we are given a candid solution then we can solve it in polynomial time.
example: knapsack problem (weight, value) pair
find the most valuable combination of items.

P vs NP Problem :

This asks whether every problem whose solution can be verified in polynomial time (NP) can also be solved in polynomial time (P)

If P = NP, then every problem for which we can verify a solution in polynomial time can also be solved in polynomial time.

If P ≠ NP, then there are problems in NP that cannot be solved in polynomial time, though their solutions can be verified in polynomial time.

NP complete problem :

A class of problem in NP that are informally the "hardest" problem in NP

If NP-complete problems can be solved in polynomial time then that means all NP problems can be solved in polynomial time (P = NP)

example : Travelling Salesman Problem :

NP hard problem :

They are as hard as NP complete problems at least but they do not necessarily belong to NP. This means there may ~~be~~ not be a quick way for verification of solution for these problem.

Some NP-hard may not even be decidable.