Unit V

P class:

The set of all languages that are accepted by some deterministic TM in polynomial time. $O(n^k)$

order of n^k

k can be any number

NP - class:

The set of all languages accepted by non-deterministic TM in polynomial time is NP.

make copies of themselves

A problem is decidable, ⇒ there exist a TM or Algorithm.

Some algorithms → we may not able to implement as program at practise.

For some problems → even though there exist algorithm we may not implement it as pgm.

To do: Revise.

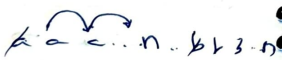
Finding out Alg which can be easily implemented as practical pgm.

All TM are equivalent in power.

ie., lang accepted in one TM will also be accepted in another type. Some TM are better than other TM for some problems.

Time Taken to accept

$a^n b^n$ – single Tape TM ⇒ $O(n^2)$ order of n^2.
Deterministic

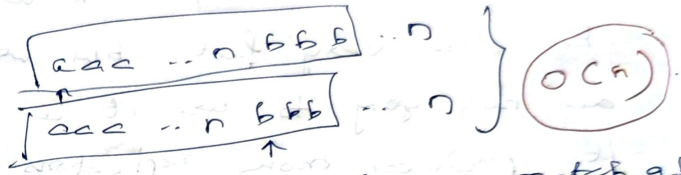


no. of cells $O(n)$
i. must cross it

how many a's & b's are compared ⇒ n

$n \times O(n) = O(n^2)$.

some problem in multitape TM.

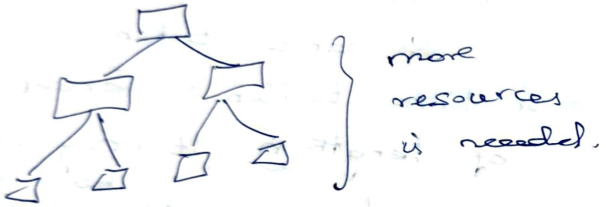


in one scan of can match a,b.
∴ So, time taken to solve or time taken to accept string of length n, it may vary significantly based on the type of TM.

→ Every TM accepts the lang. so all TM are equal in power. But Depending on the type of TM, some TM do some work faster so do it slower.

NDTM → do any work faster compared to Deterministic.

But in Non-deter TM, work will be done very fast but we are not going to use it as it produces more replications.



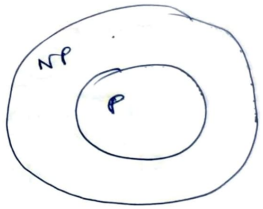
more resources is needed.

Work done fast & easy → But ↑ resources.

So, we are going to do problems which are equivalent to deterministic.

Hard are

If a problem is solved by Deterministic TM in polynomial time, definitely it will be solved by non-deterministic TM in polynomial time is NP.



P = NP ?

Assume P ≠ NP - in future there can be any proof

NDTM → DTM

$O(n^k)$ → $O(2^n)$

polynomial time → Exponential time.

Diagonalization method for Analyzing Infinite sets.

Analyzing finite set is very simple since the elements may be explicitly enumerated, whereas analyzing an infinite set is a little bit difficult task.

"Georg Cantor" - proposed a method in the year 1873, to answer the question "whether the two given infinite sets S_1 and S_2 are of equal size or not".

- Cantor stated that the given 2 sets are of equal size when there exists a bijective function to map the elements

of one set to another.

- A function is said to be bijective when it is both one to one and.

Tractability

- Some problems are decidable and others are not and so on.
- Even though some problems are decidable theoretically, they are unsolvable in practice when they grow large due to inordinate amount of time and space requirement.

* Tractability of a problem is the ability to control and manage the problem with reduced space and time.

* Complexity of the problem is analyzed by designing an abstract machine to solve the problem.

NP- Complete

Theorem: SAT is NP-Complete.

Proof:

To prove that SAT is NP-complete, two things are to be proved.

1. SAT has to be proved to be in NP
2. It has to be proved that all the languages in NP are reducible to SAT.

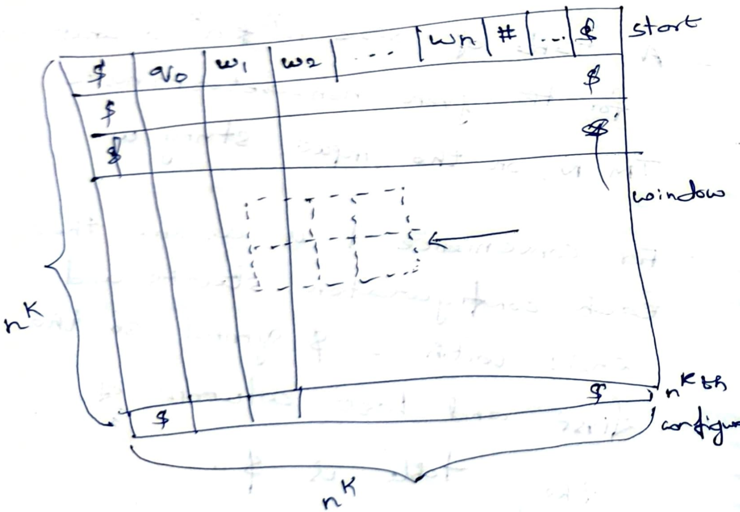
A Nondeterministic Turing Machine is constructed to prove that SAT is in NP.

Designed NDTM guesses an assignment to the given Boolean formula ϕ and moves to accepting state if the assignment values satisfies the Boolean formula ϕ.

- The TM designed finds the solution in polynomial time.
- It is assumed that there exists a NDTM N that decides A in n^K time for some constant K, where n is the length of the string.
- A table of size $n^K * n^K$ is constructed for the given non-deterministic TM N, on the input string w.
- For convenience it is assumed that each configuration starts and ends with a $ symbol so the first and last columns of the table is $.

- The first row of the table is the initial configuration of N on w and each row follows the previous one according to the transitions of N.

- The table is accepting if any row of the table is an accepting configuration.



Constructing a Boolean formula for the given TM and the i/p w

- Every accepting table for N on w corresponds to a computation branch of N on w.

- The problem of determining whether N accepts w now reduces to the problem of determining whether an accepting table exists for N on the input w.

- To reduce the constructed table to SAT, ~~SAA~~ SAT consists of Boolean formulas (ϕ) made up of Boolean variables and operators.

- To construct Boolean formula, first the variable of ϕ are formed as follows.

- If Q and Γ are the set of states and Tape alphabet of N respectively

→ $C = Q \cup T \cup \{\$\}$ then for each i and j between 1 and n^k and for each s in C there is a variable $x_{i,j,s}$.

→ Each of the $(n^k)^2$ entries of the table is called a cell and the cell in row i and column j is called cell $[i, j]$ and contains a symbol from C.

The Boolean formula Φ consists of the following 4 parts connected by AND operators. They are,

(i) Φ_{cell}

(ii) Φ_{start}

(iii) Φ_{move}

(iv) Φ_{accept}

Formula Φ_{cell} is formed as follows and ensures that it is expressing a Boolean formula.

$$\Phi_{cell} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right]$$

The symbol $\wedge$ denote "logical AND" and $\vee$ denote logical OR. The first part Φ_{cell} of the Boolean formula Φ is a very long expression that contains a fragment for each cell in the table.

- There are two parts in Φ_{cell}, as given below.

1) The first part of each fragment implies that at least one variable is turned on in the cell.

2) The second part of the fragment implies that not more than one variable is turned on in the corresponding cell.

Classification of Decision Problems:

Decision Problems – Problems which can be solved by a Turing machine, and the TM halts on every input by saying "yes" or "no".

- Some of the problems that are decidable are intractable.
- A problem is said to be intractable when it is not possible to solve them in reasonable time as they grow large in input size.
- The problems are grouped under different classes depending on the time required to solve and the type of TM used. They are

1) class P
2) class NP
3) class NP- complete
4) class NP- Hard

Class P Problems:

P is the complexity class that contains decision problems that are solvable by a Deterministic Turing machine in polynomial time.

Eg: class P problems are adding 2 nos, sorting a set of numbers, finding maximum of a list of n nos, etc.,

Eg:

Task of finding a directed path from one node to another in the given directed graph.

- one of the issues in solving this problem is to represent the graph for which the path between the nodes has to be found.
- one of the way to represent a graph is using a 2-dimensional array of size $n \times n$.

n = is the number of nodes in the given graph.

The $(i, j)^{th}$ entry is made as 1, when there is an edge from i^{th} node to the j^{th} node and $(i, j)^{th}$ entry is made to be 0, when

there is no edge between the i^{th} and j^{th} node.

Task is to find whether there is a path from S to T.

Graph to find path from node S to T.

Theorem:

The Problem of finding a path from node P to node Q in a given graph G is a class P problem.

Proof Idea:

There are several ways to prove the above statement.

The basic brute and force algorithm is to determine all the paths in the given graph G and find out whether there is any path that starts in P and ends in Q.

The polynomial time algorithm M for Path is as follows:

Assume that the adjacency matrix of the graph G is on the tape TM. To find the path from node S to T.

1) A mark is placed on node S.
2) Third step is repeated till no more nodes are marked.
3) All the edges of G from the marked node A are scanned and node B is marked when B is

reachable from A.

4) If node T is marked the input is accepted and rejected otherwise.

Now the time required by the algorithm is being discussed.

→ Step 1 and 4 are executed only once, and Step 3 runs atmost m times, where m is the number of nodes in graph G.

Thus, totally the algorithm takes 1 + 1 + m steps, giving a polynomial time solution.

Class NP

- The class NP consists of decision problems which can be solved in polynomial time by a non-deterministic Turing machine.
- we have n number of solutions, and we have to choose one best solution.

Eg:

3 SAT problem: SAT - SATISFIABILITY

$$(x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3)$$

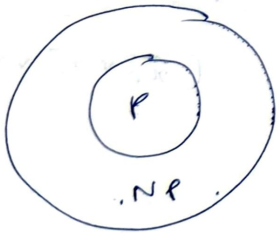
Another way to define NP problems:

A problem is said to be NP if

1) its solution comes from a finite set of possibilities

2) It takes polynomial time to verify the correctness of a candidate solution.

Since it is known that there is a non-deterministic solution for class P problems too clearly P is a subset of NP.



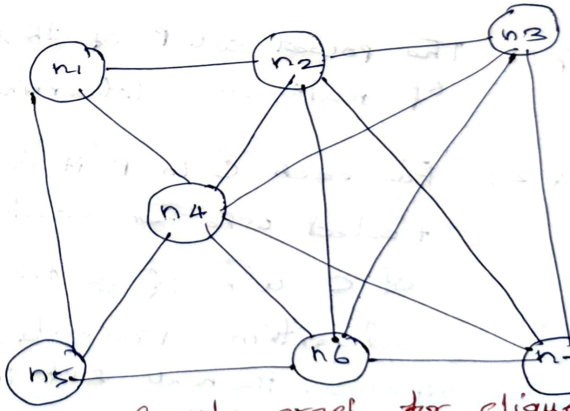
Clique Problem:

clique - complete graph.

- The input of this Problem is a graph G1 and a number K.

- The output is a subgraph of G with K nodes in which every two nodes are connected by an edge.

For Eg:- consider the following graph G with seven nodes n1 to n7.



Sample graph for clique Problem

A 5-clique may be found for this graph with 5 nodes n_2, n_3, n_4, n_6 and n_7 each node in this clique is connected to each other.

- The problem may be Proved to be a NP problem by providing an algorithm which can be simulated using a non-deterministic TM.

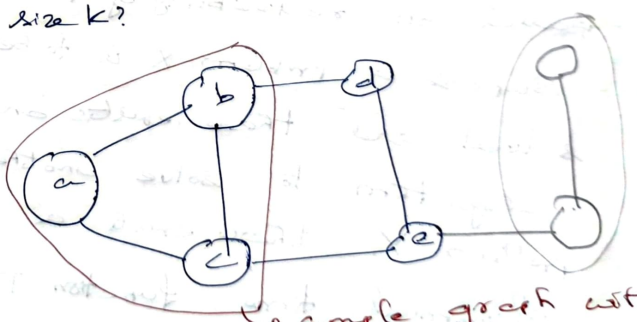
Step 1: The power set P of the set of nodes is determined.

Step 2: For each C in P it is tested whether cardinality of C is K if so the algorithm proceeds, otherwise the next set in P is chosen.

Step 3: It is checked whether all nodes are connected.

Step 4: The subset which succeeds all the tests are declared as output and the TM halts, if there is no such subset the input is rejected and the TM halts.

Is the graph contain a clique of size K?



↳ Simple graph with 3 nodes.

A clique C, in an undirected graph G = (V, E) is a subset of the vertices, C ⊆ V, such that every two distinct vertices are adjacent. This is equivalent to the condition that the induced subgraph of G induced by C is a complete graph.

NP complete problems:

- Based on reducibility of problems.
- Suppose a problem X is to be solved and there exists an algorithm to solve another problem Y, there exists a polynomial time function T to reduce problem X to problem Y then problem X can be solved with the same algorithm as that of Y and problem X has the same complexity as that of Y.
- If the function T takes polynomial time to reduce problem X to problem Y, then T is said to be polynomially bounded and written as $X \leq_p Y$. If X is polynomially reducible to Y then the implication is that Y is atleast as hard as to solve as X. Therefore it may be concluded that if $X \leq_p Y$ and $Y \in P$, the $X \in P$.

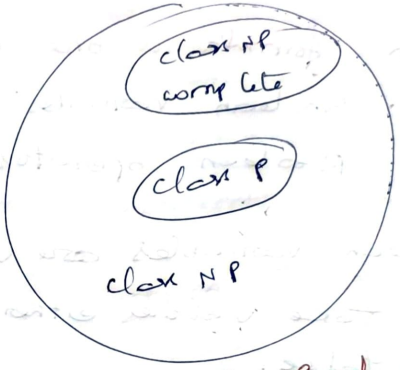
Definition of NP completeness:

A decision problem E in class NP is NP-complete if every problem in the class NP is polynomial-time reducible to E.

A language B is NP complete if it satisfies two conditions

(i) B is in NP

(ii) Every problem A in NP is reducible to B in polynomial time.



Relationship b/w P, NP and NP complete Problems

Satisfiability Problem: (SAT)

- most important Problem in tractability.
- The task is to test whether the given Boolean formula is satisfiable.

- Boolean formula is one formed with Boolean variables and Boolean operators.

- Boolean variables are variables that take values either true or false.

- The possible Boolean operations are AND, OR and NOT represented by the symbols $\wedge$, $\vee$ and $\neg$ respectively.

- A Boolean formula is said to be satisfiable if some assignment of true and false to the variables makes the Boolean formula to evaluate to 1 (true).

Cook-Levin theorem:

- Theorem about satisfiability (shortly referred as SAT problem) is as follows:

> SAT $\in$ P iff P = NP.

- To demonstrate a polynomial time reduction, 3 SAT problem is introduced.

- Terms involved in 3 SAT are

(1) literal - a Boolean variable or a negated Boolean variable.

(2) clause - several literals connected with OR's (ie.) $\vee$'s

eg: $(x_1 \vee x_1' \vee x_2' \vee x_4)$.

A Boolean formula is in conjunctive normal form if it comprises several clauses connected with $\wedge$s.

For eg:

$(x_1 \vee x_2) \wedge (x_3 \vee x_6)$ is a conjunctive normal form.

A CNF is said to be a 2 CNF formula if all the clauses in it have three literals.

For eg, the Boolean expression $(x_1 \vee x_2' \vee x_3') \wedge (x_3 \vee x_5' \vee x_6) \wedge (x_3 \vee x_6 \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$ is 3 CNF. In a satisfiable CNF formula, each clause must contain atleast one literal to be true.

Theorems:

① 3 SAT is polynomial time reducible to CLIQUE.

② If a problem B is NP complete and $B \leq_p C$ for any problem C in NP, then C is NP complete.

③ SAT is NP-complete.

④ Vertex-cover Problem is NP-complete.

⑤ NP-hard.

A problem A is said to be NP hard when all the NP problems are reducible to A and ~~A ∈ NP~~ $A \notin NP$.

(i) $P_1 \leq A$ for all $P_1 \in NP$

(ii) $A \notin NP$.

For eg: travelling salesman problem is NP-hard.

Class NP:

- Polynomial time
- Non-deterministic TM.

~~3 SAT~~:

Satisfiability Problem:

- most impor problem in tractability
- Task of SAT is to test whether the given Boolean formula is satisfiable.
- Boolean Formula: is one formed with Boolean variables and Boolean operators.

Boolean Variables: Variables that take values either true or False.

Boolean operations → AND, OR, NOT
$\wedge$, $\vee$, $\neg$

A Boolean formula is said to be satisfiable if some assignment of true and false to the variables make the Boolean formula to evaluate to 1 (true).

Cook levin theorem about satisfiability (SAT problems) is as follows:

SAT $\in$ P iff P = NP

3 SAT is a special case of the satisfiability problem and the terms involved in 3 SAT are:

① Literal - Boolean variable (or) a negated Boolean variable

② Clause - several literals connected with OR's (ie) V's

eg: $(x_1 \vee x_2' \vee x_3' \vee x_4)$

③ Conjunctive Normal Form (CNF):-

A Boolean formula is in conjunctive normal form if it comprises several clauses connected with $\wedge$'s.

Eg: $(x_1 \vee x_2) \wedge (x_3 \vee x_4)$ is a CNF.

A CNF is said to be a 3 CNF formula if all the clauses in it have 3 literals:

Eg: The Boolean expression

3 CNF $\{ (x_1 \vee x_2' \vee x_3') \wedge (x_3 \vee x_5' \vee x_6) \wedge (x_3 \vee x_6 \vee x_4) \wedge (x_4 \vee x_5 \vee x_6)$

(literals)

is 3CNF.

* In a satisfiable CNF formula, each clause must contain atleast one literal to be true.

Theorem:

3 SAT is polynomial time reducible to ~~CLIQUE~~ CLIQUE.

Proof:

This theorem is proved using Proof by contradiction construction.

Steps involved are:

(i) The given 3SAT formula is converted to a graph.

(ii) It is determined whether there is a K-clique for the constructed graph, where K is the number of clauses in the given Boolean formula.

(iii) The Boolean formula is concluded as satisfiable, when there exists a K-clique.

The Boolean formula Φ is considered with K-clauses as

$\Phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots \wedge (a_k \vee b_k \vee c_k)$ and reduced to clique.

Since this is a 3SAT problem each clause consists of 3 literals. The procedure to construct the graph corresponding to the Boolean formula is as follows:

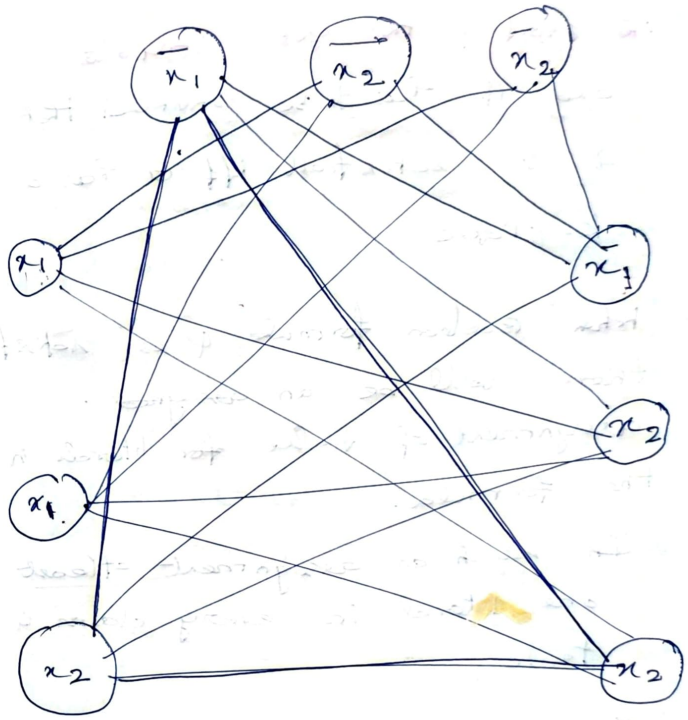
① one node is provided for each literal in the formula, therefore there will be 3*K nodes in the graph. (repetitions of variables are allowed).

(2) An edge is added ~~from~~ from each node to all the nodes except for the nodes that belong to the same clause and nodes with opposite value (ie) node x_2 do not have an edge to x_2'.

With this idea a graph is constructed for the Boolean formula

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (x_1' \vee x_2' \vee x_2') \wedge (x_1' \vee x_2 \vee x_2),$$

The graph is as follows:



Since there is 3 clause, the problem may be concluded as satisfiable if there is a 3-clique for the constructed graph.

→ The proof is not yet complete and it has to be convinced that Φ is satisfiable iff G has a k-clique.

→ When Boolean formula Φ is satisfiable, there will be an assignment of value for literals in the formula.

→ In such an assignment atleast one literal in every clause is true.

- The nodes corresponding to the literals of the same clause form a group triple in the graph G.

- In each triple of G, one node corresponding to a true literals is selected in the satisfying assignment.

- If more than one literal is true in a particular clause, one of the true literals is chosen arbitrarily. Proceeding in this way the nodes selected form a k-clique.

- Since nodes of the same triple are not connected, only one literal (node) is considered from one triple (clause) and 3SAT problem is reduced to CLIQUE.

- 3SAT problem has been reduced to a k-clique problem, hence it may be stated that if CLIQUE is solvable in polynomial time, so is 3SAT.

Even though these two problems appear different at the first glance, they have some complexity since there exists a polynomial reduction process to convert 3SAT to a CLIQUE.

Classification of Decision Problems.

Problems are grouped under,

1) class P
2) class NP
3) class NP- complete
4) class NP- hard.

class P: finding path from one node to other. ⇒ shortest Path Problem.

NP:

↳ Popular NP problem is determining Hamiltonian path in a given graph.

↳ clique problem.

↳ Traveling Salesman Problem.

NP complete Problems.

→ SAT

→ Vertex cover problem.