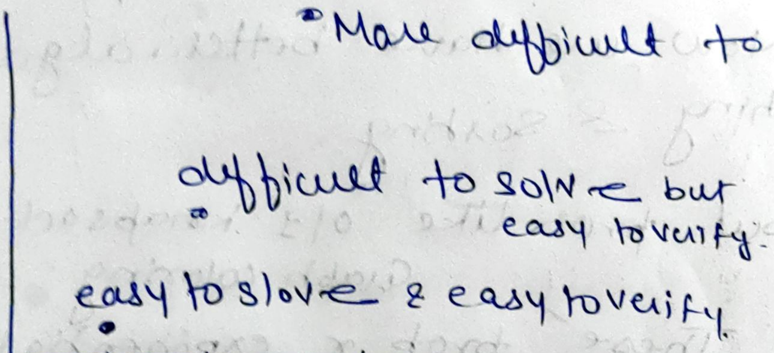
Algo we have seen so far are

Polynomial		Exponential	
Linear Search	n	0/1 knap	2^n
Binary	$\log n$	TSP	$2^n n^2$
Insertion Sort	n^2	Sum of subset	2^n
Merge Sort	$n \log n$	Graph Coloring	2^n
Matrix Mult	n^3	Hamiltonian cycle	2^n



for $n=10$
$n^{10} < 2^n$
2^n
for large values of n.

Now for searching: Linear $- n$, binary $\log n$ (less time)

Sorting: Insertion $- n^2$, Quick $- n \log n$

So if linear seach is already there what is the need of binary seach.
or if Insertion Sort is already there what is the need for Merge / Quick.

So the ans: is we ~~as a~~ human always wanted to do the things fast

Our aim is design a Algo for a prob which takes less time. So we always want to improve the algo.

Like Linear $\longrightarrow$ binary

Insertion $\longrightarrow$ Merge | Quick

So Now we have better algo for Searching & Sorting.

but prob like 0/1 knapsack, TSP Graph coloring, ...

These prob r exponential prob.

We have already seen order of growth funct^n with respect to i/p size n.

it is very clear that 2^n function (For n= ~~40~~ 100) will takes $2^{40} = 1.1 \times 10^{12}$ (abt trillion op^n.)

$2^{100} \approx 1.27 \times 10^{30}$

if we assume a high end computer which execute 10^{12} (trillion) operation / sec.

then also $\frac{2^{100}}{10^{12}} = 1.27 \times 10^{18}$ sec

$\approx$ 40.3 billion Year.

which is nearly 3 times the age of universe

So we may conclud that utility of algo with expo. time complexity is limited to small value of n (typically $n < 60$)

So basically complexity theory determines the limits of computer. what computer can and can not do.

There is no optimization algo exist for such problem (for large values of n)

So the algo whose complexity is expo. is is as good as nothing.

From this we can classify problems into 2 category.

Computable prob :- if an algo solve the problems into finite no. of steps (or time) then it is called as Computable problem.
We also know that there are 5 properties of an algo. & but of that 1 is finiteness - means algo. should finish in finite no. of steps/time.

if the algo fail to achieve this property i.e infinite no. of steps then the prob is Non Computable prob.

All expo. algo are Non computable prob.

Prob

Computable	Non Computable.
- finite time	↳ Infinite time
- Complexity will be polynomial	↳ ~~exponential~~
↳ algo exist	No algo
sorting / searching	Halting prob
TM solvable	No.

Halting - Given a prog & i/p determine wether the prog will halt or run forever.

There is another category of prob

optimization

- Solvable
- More time
- ex: Max-Min
- can be converted in to decision prob
- finding the best solut?

decesion

- Yes/No decesion
- less time as compared to optimization
- Verifiable prob

Can a graph is colored with 3 colors

Is there a path b/w Node a & b.

or for TSP is the shortest route is < k.

- can be converted into optimization prob

* Approximation algo are used to convert hard optimization prob (NP) into decision prob. to ↑ efficiency.

TSP :- is there a Hamiltonian cycle of cost ≤ 100.

optimization - What is the shortest Hamiltonian cycle.

Problem

P

if a DTM solve prob in Polynomial time then the prob is p prob

These are tractable prob.

Complexity will be upto n^k

NP

* NDTM - Solve in P time but
* DTM can not solve
* DTM can only verify in P time

Non tractable

finding a solut^n takes expo. time in DTM

ex:- SAT, sum of subset.

DTM :- Single computational path (system)

NDTM :- can only guess the correct solut^n in P time & verify in P time.

Que is :- How NDTM. Solve Prob in P time ?

Or how to solve NP prob ?

You can do 2 things

1. Try to relate NP problems.

atleast solve 1 NP problem and solve the similarities b/w all the prob. so that if one problem can be solved all other can also be solved using the same algo. This concept is called Reduction.

for ex: if we solve knapsack then we will try to relate TSP using knapsack. & will prove that if 1 can be solved other can also be solved.

- Show the relation / association b/w them

means either you solve the NP prob or relate it.

ex:-

Number scrabble

1 2 3 4 5 6 7 8 9

Select 3 no. that add up to 15.
2 person Game.
who adds the no. 15 will win the game.

1^{st} user → 3

2^{nd} → 7

1^{st} → 8

Now instead of arranging the no. in a single line we will add the no in ~~order~~. Magical Square

2	7	6
9	5	1
4	3	8

276
951
438.

every row/column & diagonal gives the result 15.

is it look something similar
Same as tic tac toe.
so here ~~Mag~~ Magic square is same as tic tac toe. they are similar. if ~~two~~ we have reduced Magic sq → tictactoe prob.

if two problem can be reduced to same prob. they can be solved with the same strategy.

So any strategy can solve tic tac toe can also solve numbers scrabble prob.

strategies are nothing but algorithms

depending on the prob algo can be more or less complex.

Problem — how complecated they are

classify the prob into complexity classes is the area called computational complexity

P

$1, n, n^2, n^3 \ldots$

NP

verifiable in P time

Solvable in exponential time

SAT, TSP, Sudoku

Candy Crush

↳ NP Hard.

if P = NP then all of our security password would be a free game for a world.

(2) When you are unable to write deterministic algo, then write a ND Algo for the same prob.

if polynomial time algo is not possible then write ND algo of P time.

How to write ND algo ?

Uptill now whatever algo we have written are deterministic means each & every step is clear, how it works we know it clearly.

Deterministic — Everything is known.

Non — " — Something we are knowing & something not knowing.

so here for any prob like 0/1 knapsack you may be knowing many stmts but some of the stmts takes exp. time or you are unable to figure it out how to make them polynomial, then leave them blank.

so in ND algo some stmts are determ[inistic]
some are Non deterministic.

whatever is ND just leave them blank
so in future some body else will work
on ND part. In this way we can
preserve our work.

~~eg:~~ Knapsack algo — ~~ND~~ algo.

ND algo has 3 functions:-

(1) choice — arbitrarily choose 1 from mor[e]
(2) Success — signal successful completion
(3) failure → signal unsuccessful completion

& we assume that all these stmt will take
1 unit of time.

* A ND. algo comes to failure only when
there exist no set of choices leading
to a successful signal.

ND algo for searching.

Searching an element x from a given set of element A[1:n]

Nsearch (A, n, ~~key~~ x) – ~~(1)~~
J = choice [1, n) – (1)
if A[J] = x then
{ write (J); – (1)
success();
}
write (0); – (1).
failure();

Complexity : O(1) → ND algo
where as DA algo → O(n)

Choice – means randomly choose 1 of the ele. from the set

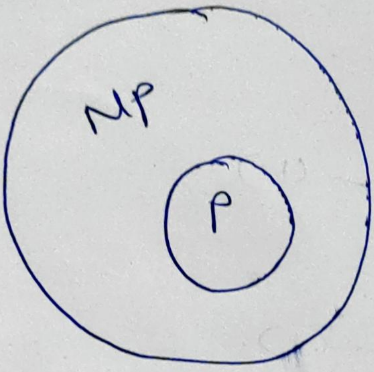
So choice can be either 0 or 1. choice is a ND stmt. ~~we~~ Right now we are not knowing how choice is selecting any ele from array. once we will know it will become deterministic.

choice is a <u>Magic</u> – becoz we are not knowing how it is working

* 0/1 knapsack – PPT.

P – all the algo who takes polynomial ti
& there algo is deterministic ca
P time.

NP – algo who takes polynomial time
& the algo Non deterministic
then it ND Polynomial time alg



$P \subseteq NP$.
or $P \nsubseteq NP$.

7 Millenial Price
prob. from
Clay university

earlies all algo are ND. out of which
some algo becomes deterministic after the
research work. So ~~x~~ out of NP prob we
have converted some into P prob.

Some are beliving that $P = NP$ but no
one has a proof, which eq^n is correct.
but if $P = NP$ then all our cryptography
concept, security password
will be easily crackable at P time
world would be in a different place.

As we discussed we can do two things for and ND prob.

(1) Write an ND algo (temporary)

(2nd) relate 1 ND prob with another so that if one can be solved all can be solved.

To relate one prob with another we need one base prob.

Let CNF is the base prob.

<u>CNF satisfiability</u>

3 CNF → 3 variable & the relation of variable.

$$x_i = \{x_1, x_2, x_3\}$$

$$CNF = \underbrace{(x_1 \vee \bar{x}_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)}_{C_2}$$

OR/conjuction

disjuction

Satisfiability prob is the where we need to find for what values of x_i the result will be true.

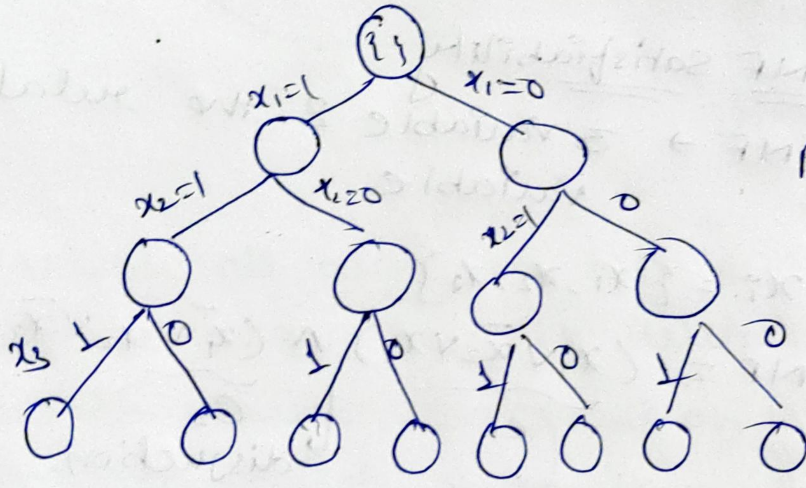
x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

8 possibility

$2^3 \Rightarrow 2^n$

we need to try 8 possibilities to check for which it is giving true result

exponential prob.



Any one path will give u true result

Now we have to relate this prob with all the exponential time taking algo.
So if any prob can be solved all prob can also be solved

if SAT can be solved in P time all can be solved in P time.

relation of SAT with 0/1 knapsack prob.

0/1 knapsack

$n=3$ $P = \{ \}$
$w \{ \}$
$x = \{0/1, 0/1, 0/1\}$

x_1	x_2	x_3
0	0	0
0	0	1
⋮		
1	1	1

this is also $\underline{\underline{2^n}}$ prob

if we prepare a state space tree it will be same as SAT prob.
both are having same tree. so if 1 can solve in P time then the other can also solve in P time.

NP Hard & NP Complete prob

Satisfiability $2^n \rightarrow$ - Now lets say SAT is NP Hard

0/1 knapsack 2^n
TSP
Sum of subset 2^n
Graph coloring 2^n
Hamiltonian - 2^n

Hard prob.

if SAT is solved in P time all can be solved in P time

Reduction :-

SAT α 0/1 knapsack

$I_1 \ \alpha \ I_2$

it takes polynomial time to reduction

if 0/1 knap. is solved in P time with the same algo can also solve SAT or vica-versa.

Conversion itself is P time. if it takes expo. time then no use!

NP HARD $\xrightarrow{\text{Poly}}$ NP Hard

SAT α 0/1 knap

SAT α L_1
↳ NP Hard

$L_1 \alpha L_1$
↳ NP Hard. (transitivity property)

So for ND prob
we are doing 2 things
(1) Reductn (2) ND algo

Do we have ND algo for any prob?
Yes for SAT we have ND algo.

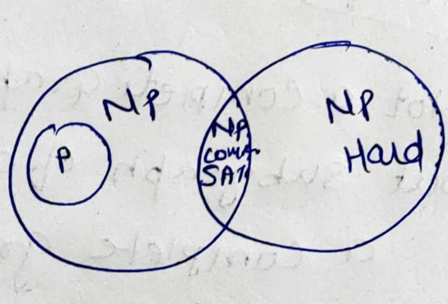
~~So any prob whose~~ ~~an~~ ~~is NP Hard~~ ND algo exist
is called NP complete

So any prob who is NP Hard & whose ND algo exist is called NP complete

So SAT is NP Hard & NP Complete.

for NP Hard prob if ND algo is also there then it is called NP complete

if a prob is reduced by SAT & a ND algo is also there then it is NP comple

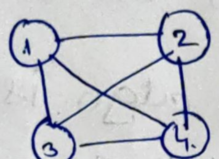


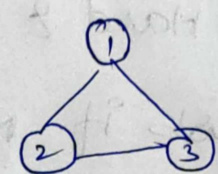
P is growing – & we belive that at some stage it will be equivalent to NP.

if we are able to prove that P = NP whatever ND we are writing tomorrow it will become deterministic.

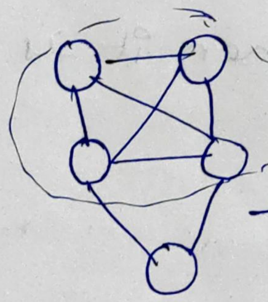
Cooks :- if SAT is laying in P then only P = NP

Clique decision Prob.





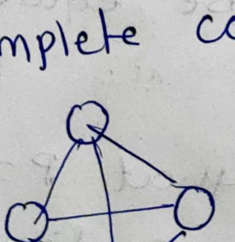
→ Complete Graph has
$|V| = n$
$|E| = \frac{n.(n-1)}{2}$



→ clique

→ Not a complete graph but subgraph of a graph is a complete graph

A subgraph of a graph which is complete called clique.



clique
K = 4
K = 3
K = 2
all are present

CDP – is there a clique of size 2/3 – Ans is Yes