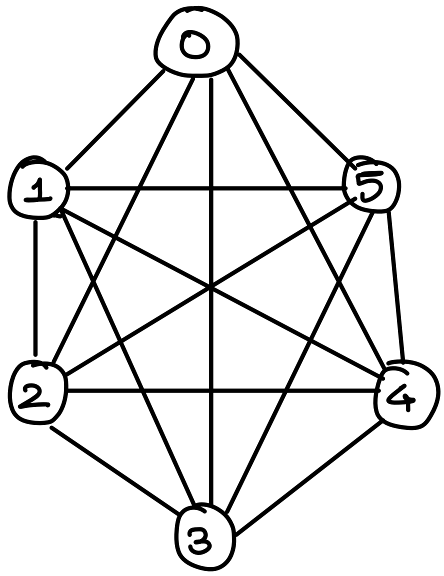
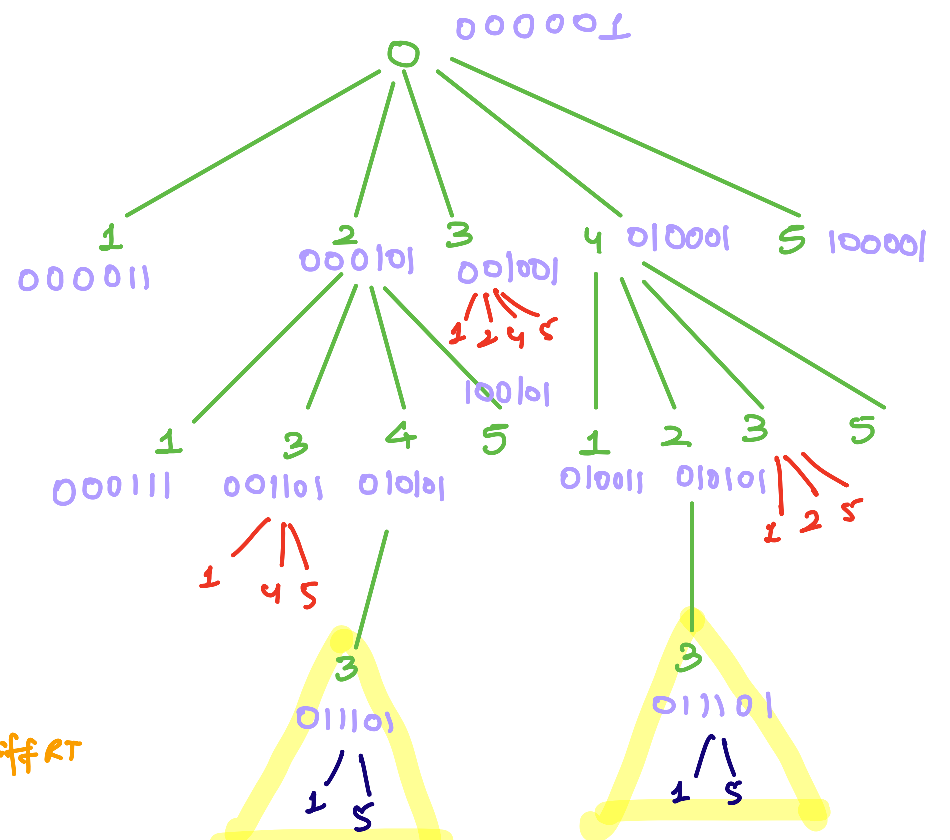
TRAVELLING SALESMAN PROBLEM (TSP) using DP





1. only vertex problem ?
   Different 3 are having diff RT

(3, 29) cell ans store

32 16 8 4 2 1
0 1 1 1 0 1 : 16+8+4+1
: 29

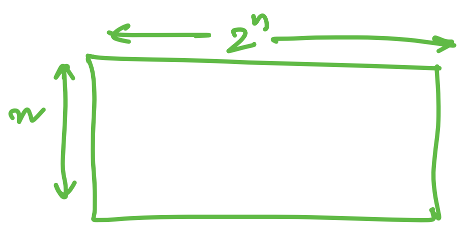
5	4	3	2	1	0
1	1	1	1	1	1

: $2^6 - 1 = 63$ → 63 index : array size : 64 → 2^6

$1 << 6$

	8	4	2	1		
2bit:			1	1	: 3	: 2^2-1
3bit:		1	1	1	: 7	: 2^3-1
4bit:	1	1	1	1	: 15	: 2^4-1

TC:



$(n \times 2^n)$ cells fill

TC: $n^2 \times 2^n$ (exponential)

P/NP :

Since we have looked at many algorithms, an interesting question to ask is - is every problem easy to solve ? or are their some problems which are difficult to solve.

Scenario:

You are s/w Engineer, your manager has asked you to solve a problem.
— After 1 month —
You run your program on given input. Program is running and everyone is waiting for your program to stop and show the result. But your program is not stopping.
— After few hours —
Still your program is running.

Manager says we will give you more time but your program should definitely halt in 1hr.

You tried again. But still you are not able to solve in 1hr.

You are given a problem to solve and you are not able to solve it in "Polynomial time".

→ Generally polynomial time algorithm will run faster.

"P" → $O(n^k)$ X

→ when you go to your manager donot say I can't solve it quickly. Instead say till now no one has been able to solve it in polynomial time.

It is very easy to prove a problem is solvable → solve it
It is difficult to prove a problem is not solvable → for manager its not

possible to recruit 100 people and show no one is able to solve it.

To prove something is not solvable we have some theories.

Problems

Decidable Problems
(algo exists)

procedure which will halt after sometime.

Tractable

If there exist at least one polynomial bound algorithm.

$O(n^k)$

k can be 10, 100, 1000

Intractable

If a problem is not tractable then it is intractable.

$O(n^{\log n})$ $O(c^n)$

Eg: TSP $n^2 2^n$

Here we compromise

In order to give exact answer it will take a lot of time.

Heuristics : we won't solve the problem exactly. we will try to give the approximation. we go for approximate algos which will not solve the problem completely but will give you close answers.

Approximation Algos

Undecidable
(no algo exists)

Like halting problem of Turing machine.
(given an input to TM whether it will stop or not)

intractable

Problems which are hard to solve :

- TSP : In a graph G, shortest path covering all vertices exactly once.

- 0/1 Knapsack : Given cap, profit and weight find out the maximum profit.

- LCS : given 2 sequences find LCS.

All these are optimization problems min, max

Generally talking about these problems directly is difficult.

If you have to say a problem is hard : take a problem easier than this and prove that easier problem is hard therefore this problem is hard.

Finding simpler problems for TSP :

Instead of using original problem, frame a different problem. Answer me in yes/no.

Is there any shortest path covering all vertices of length atmost k.

Yes/NO

Decision Problems

answer find out | Yes/NO

Convert : optimization problem → Decision problem

If decision problem itself is hard then optimization problem will even be harder.

Decision problem for 0/1 knapsack:

Is there any solution whose profit is atleast k.

Decision problem for LCS:

Is there any subsequence whose length is atleast k.

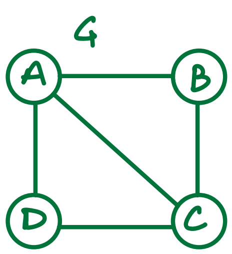
Optimization $\longrightarrow$ Decision Problem

TSP: finding out the shortest path (CP) — whether there is a shortest path of length atmost 10 (End Term)

If optimization problem is easy ($O(n^k)$) then decision problem is also easy

If decision problem is hard then optimization problem is hard.

Verification Algorithms:



Is this graph hamiltonian? → Decision Problem

$A \to B \to C \to D \to A$ → Solution

You are given a graph, question and also the answer.

Verification Algo:

You need to verify whether it is correct answer or not.

- Covering all vertices exactly once
- There should be a path from
  A → B → C → D → A

Yes it is hamiltonian cycle
and hence it is a hamiltonian graph

P NP Introduction (Decision Problems)

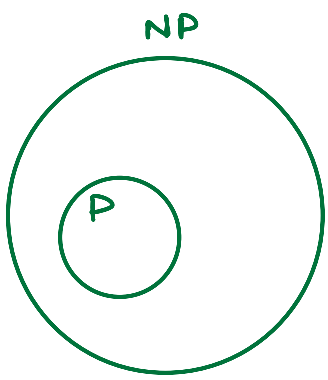
P Class: : Set of all decision problems which have polynomial time algorithms to solve them.

NP Class: : Set of all decision problems which have polynomial time verification algo.

Prim/Kruskal P
$E \log V$

In G, MST whose weight is atmost 10? $E \log V$ Yes/NO

In FKS, profit is atleast 10. $n \log n + n$



Polynomial Time Reduction:

A problem 'A' is said to be polynomial time reducible to a problem 'B' if :

i) Every instance 'α' of 'A' can be transformed to some instance 'β' of 'B' in polynomial time.

ii) Answer to 'α' is 'YES' if and only if answer to 'β' is 'YES'.

$$A \xrightarrow{\text{"Poly"}} B$$
$$\alpha \qquad\quad \beta$$

if 'B' is easy then 'A' is easy.

if 'B' is in 'P' then 'A' is also in 'P'.

$$\begin{array}{c} A \xrightarrow[O(n^k)]{} B \\ \downarrow \\ \text{Not P} \end{array}$$

if 'A' is not in 'P' then 'B' is not in 'P'.

Example:

A: Given 'n' boolean variables with values $x_1, x_2 \cdots x_n$ does atleast one variable have value "True"?

B: Given 'n' integers $i_1, i_2 \cdots i_n$ is $\max(i_1, i_2, \cdots i_n) > 0$

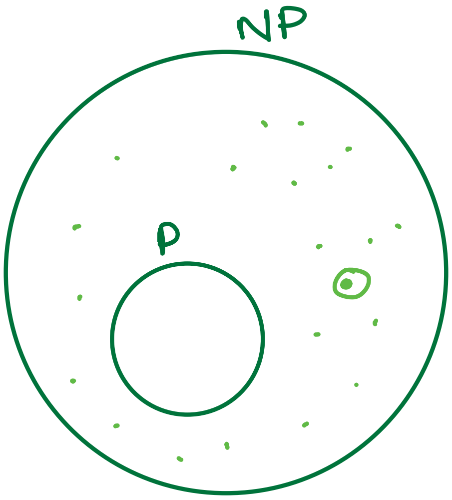
Example: $n = 4$

A: (T F F T)

B: (−30, 10, 0, 2)

T	F	F	T	A
1	0	0	1	B

$$A \xrightarrow[O(n)]{} B \quad O(n)$$

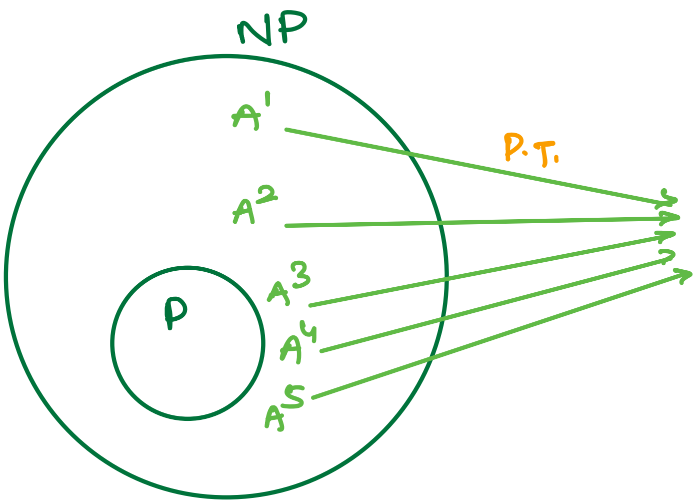


$P \stackrel{?}{=} NP$

$P = NP$ → take every problem (NP-P) and show that they have polynomial time solution.

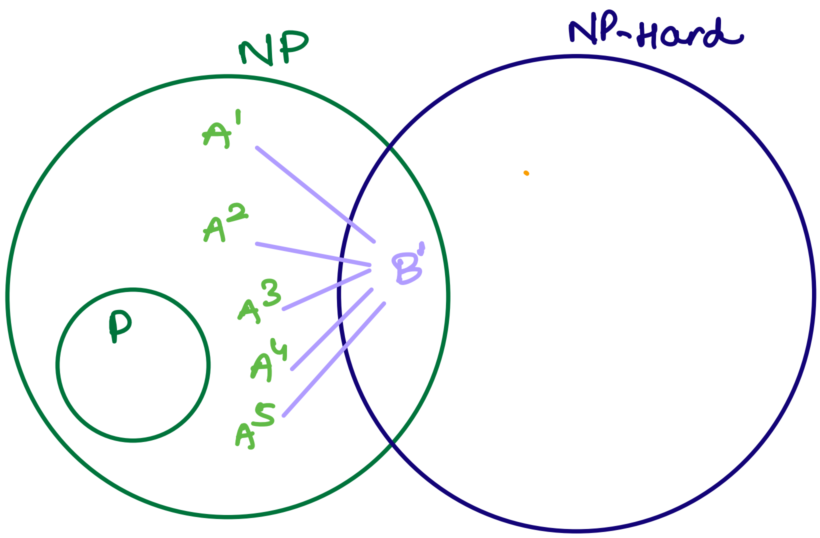
$P \neq NP$ → Prove that there is at least one problem in (NP-P) which is not polynomial time solvable.

NP-Hard:

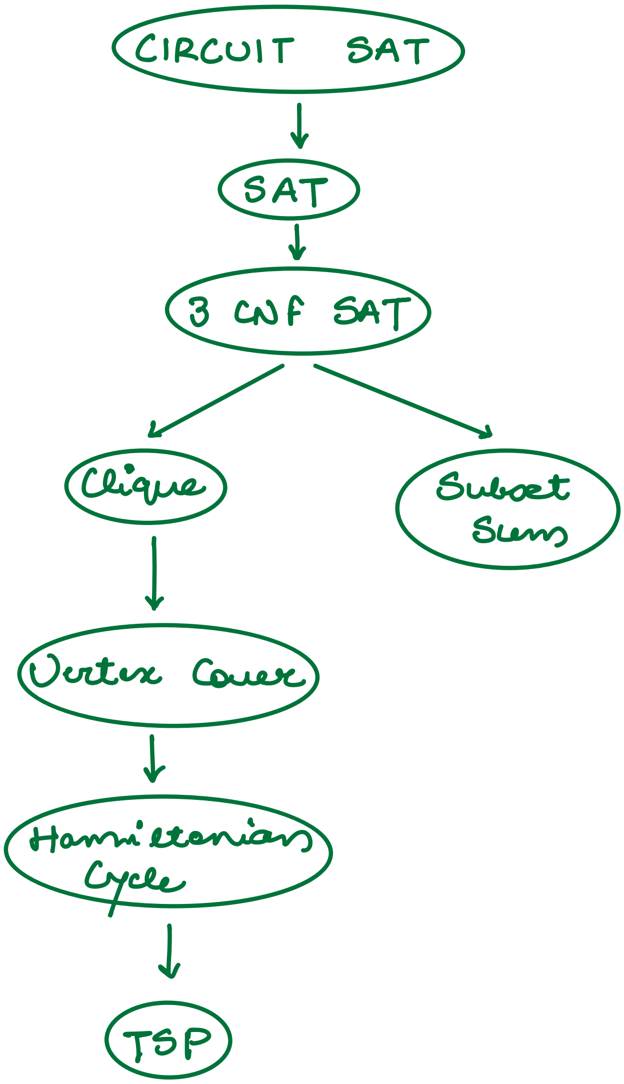


If every problem in NP can be polynomial time reducible to a problem 'B' then B is called NP Hard.

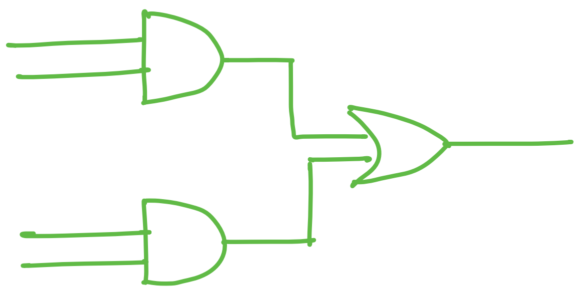
NP-Complete → NP and NP-Hard



If "B" lies in NP then it is NP-Complete



CIRCUIT - SAT:



Is there any combination of input which could give a value of True in output.

SAT:

Kind of formula

$$(x+y+z) \cdot (yz) \cdot (xy)$$

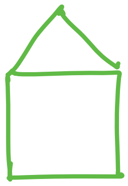
3 CNF SAT:

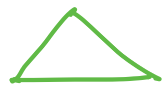
Conjuctive Normal Form

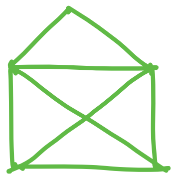
$(x+y+\bar{z}) \cdot (y+z+a)$

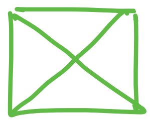
Clique:

maximum subgraph of a graph G which is complete.

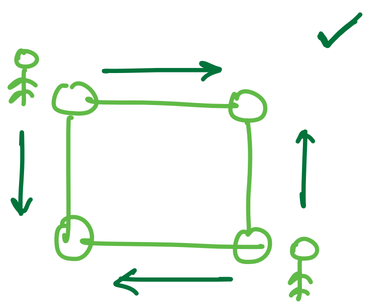
G: 

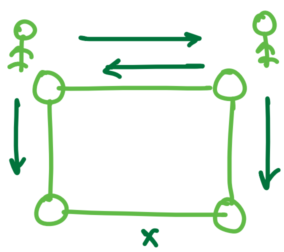
subgraph which is complete: 





Vertex Cover:





How many watchmans should be placed so that every edge is covered.

Subset Sum:

Given a subset, is its sum so and so?

Hamiltonian Cycle:

Visit every vertex exactly once and come back to same vertex.

TSP :

Find out hamiltonian cycle of least cost.