P, NP, NP-complete and NP-Hard Problems.

P:

P → Polynomial Time.

> P refers to the class of decision problem that can be solved by a deterministic Turing machine in polynomial Time. In simple terms, A problem is solvable easily in the polynomial Time based on the Algorithms efficiency and input Taken it is called as P-type problem.

> Time complexity is less. ($O(n^k)$).

Example:

1) Searching ($O(\log n)$)
2) Sorting ($O(\log n)$)
3) Matrix Multiplication

NP:

NP → Non-Deterministic Polynomial Time.

Note: NP does not stand for 'non-polynomial'

> NP refers to the set of decision problems for which a given solution can be verified in polynomial Time. while a solution to an NP problem might be difficult to find, if someone provides a solution, it can be checked relatively.

> In other words, It is a type of computational problem that if a solution is given it can be

Verified in polynomial Time. However, there may not necessarily be an efficient algorithm to find the solution itself in polynomial Time.

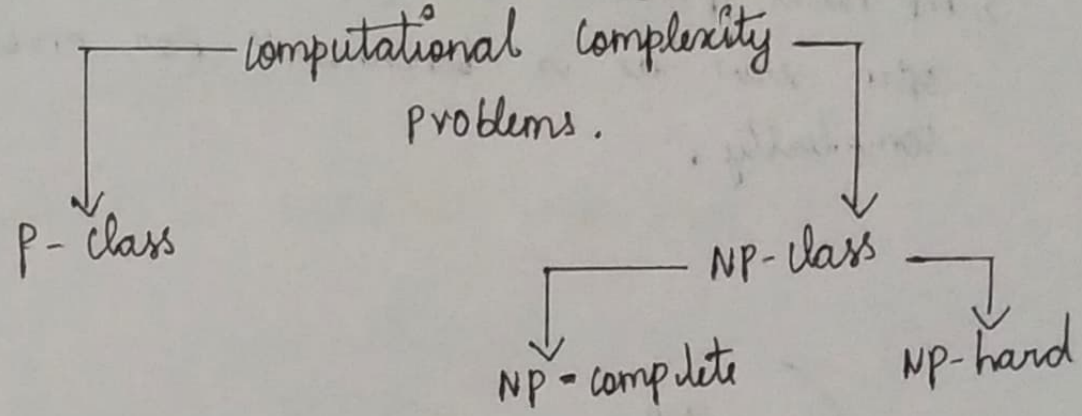
> The Time complexity for problems in NP is exponential. $(2^{O(n^k)})$.

Example:

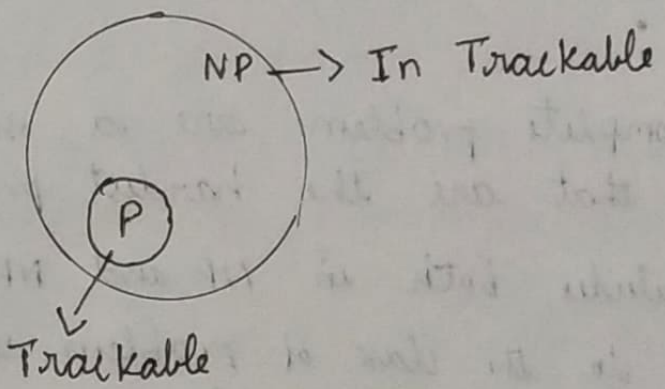
1) Travelling sales person $[O(n^2 2^n)]$.
2) Knapsack problem. $(O(2^{n/2}).)$
3) Graph coloring (2^n)

Difference between P & NP problem.

P class problems	NP class problems
Problems in P can be solved by algorithms in polynomial Time.	Problems in NP can be verified in polynomial Time. but might not have known polynomial-time Algorithm For their solutions.
All the P class problems are basically deterministic.	All the NP class problems are basically non-determi-nistic
Every problem which is a P class is also in NP class.	Every problem which is in NP is not the P class problem.
P class problems can be solved efficiently.	NP class problems can not be solved efficiently.
Eg: Binary search, bubble sort	Eg: Knapsack problem, TSP.



Relation between P & NP:



Trackable : Problems can be solved in polynomial Time.

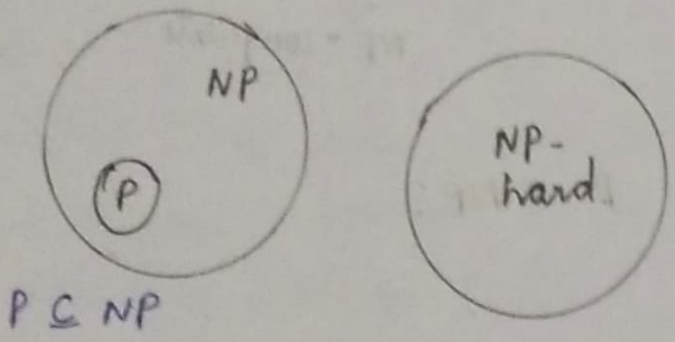
InTrackable : Cannot be solved in polynomial Time.

NP-Hard :

> A problem is NP-hard if its at least as hard as the hardest problem in NP.

> A polynomial-time algorithm for any NP-hard problem, it would imply that P = NP.

> NP hard problems are difficult to solve and often serve as a benchmark for problem complexity.

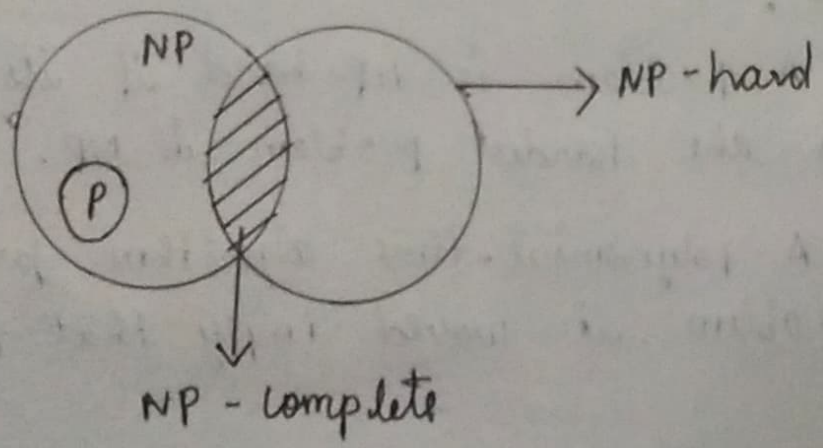


NP-complete:

> NP-complete problem are a subset of NP problems that are the hardest problem in NP.

> It includes both in NP and NP-hard, they belongs to the class of problem where solutions can be quickly verified.

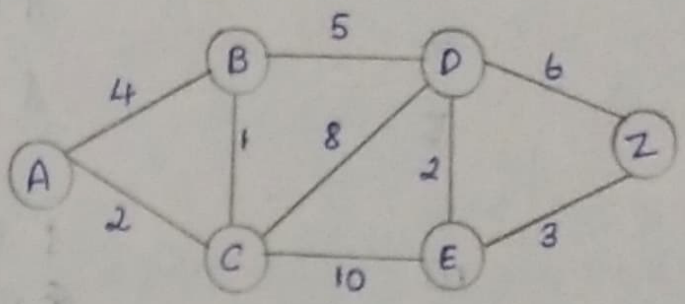
> Solving any NP-complete problem in polynomial Time would imply that every problem in NP could also be solved in polynomial time (ie, P=NP).



Example:

P class Problem:

Let us Take Dijkstras Algorithm:



$a \to c = 2$

$a \to c \to b \to 2+1 = 3$, $b = 3$

$a \to c \to b \to d \to 3+5 = 8$, $d = 8$

$a \to c \to b \to d \to E \to 8+2 = 10$, $e = 10$

$a \to c \to b \to d \to e \to z \quad 10+3 = 13$, $z = 13$.

$[a \to c \to b \to d \to e \to z]$ → is the minimum path

Time complexity: $O((V+E) * \log(V))$.

NP class problem:

Knapsack problem:

1. Find the most valuable subset of item that fit into the knap-sack.

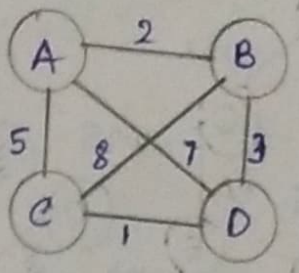
Item	weight	Value	
1	7	$42	W = 10
2	3	$12	
3	4	$40	
4	5	$25	

Item	weight	Value
One possible.		
1	7	$42
2	3	$12
3	4	$40
4	5	$25
Two possible		
1,2	10	$54 * } N.F.S
1,3	11	*
1,4	12	
2,3	7	* $52
2,4	8	* $37
3,4	9	* $65
Three possible		
1,2,3	14	X } No feasible solution
1,2,4	15	X
1,3,4	16	X
2,3,4	12	X
Four possible		
1,2,3,4	19	X

Optimal solution = (3,4) $W = 9$ $V = \$65$

Time complexity = $O(nW)$.

NP-Hard problem: (Travelling salesmen problem)



a → b → c → d → a = 2+8+1+7 = 18

a → b → d → c → a = 2+3+1+5 = 11

a → c → b → d → a = 5+8+3+7 = 23

a → c → d → b → a = 5+1+8+2 = 11

a → d → b → c → a = 7+3+8+5 = 23

a → d → c → b → a = 7+1+8+2 = 18

→ optimal solution.

Time complexity: $O(n^2 * 2^n)$

NP-Complete problem: (subset problem)

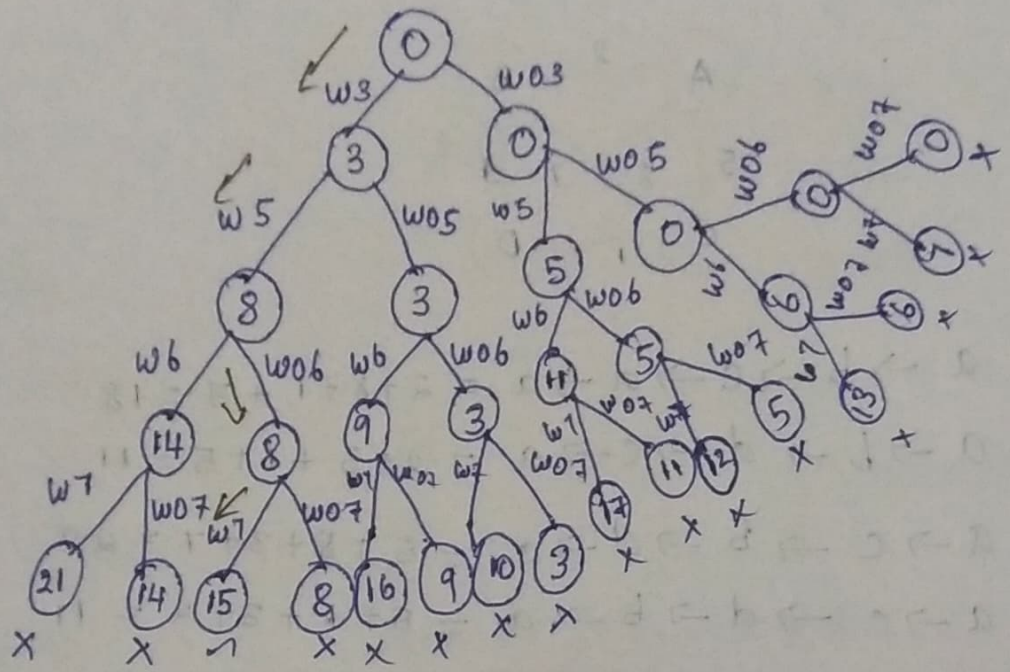
nodes → {3, 5, 6, 7}

D → 15

↓

Destination (or) Required Value.

> Initially start with 0 node.



Steps :

> Insert the values on both side of the nodes untill all the nodes insert.

> Choose the way to find the Target value node.

Time complexity : $O(n * sum)$

Short cut method to check answer :

D = 15

Add the nodes which will give this Solution.

D = { 3, 5, 7 }

Here, w06 is not calculated.