Module - 5

Backtracking

Nithin Kumar
Asst prof Dept of CSE
VVCE, Mysuru

Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.

* General Method of Backtracking → In order to applying backtracking to a specific class of problems, one must provide the data "p" for the particular instance of the problem that is to be solved & six procedural parameters. using State-Space Tree.

→ These procedures should take the instance data p as a parameter & should do the foll

* Root (p) → return the partial candidate at the root of search tree

* Reject (p,c) → return true only if the partial candidate c is not worth completing.

* Accept (p,c) → return true, if c is soln of problem, & false otherwise

* First (p,c) → generate the first extension of candidate c.

* Next (p,s) → generate the Next alternative extension of a candidate, after the extension s.

* Output (p,c) → use the solution c of p, as appropriate to the application.

Here, we will discuss three problems using Backtracking.

* Subset - Sum problem
* N - Queen's problem
* Hamiltonian Cycle problem

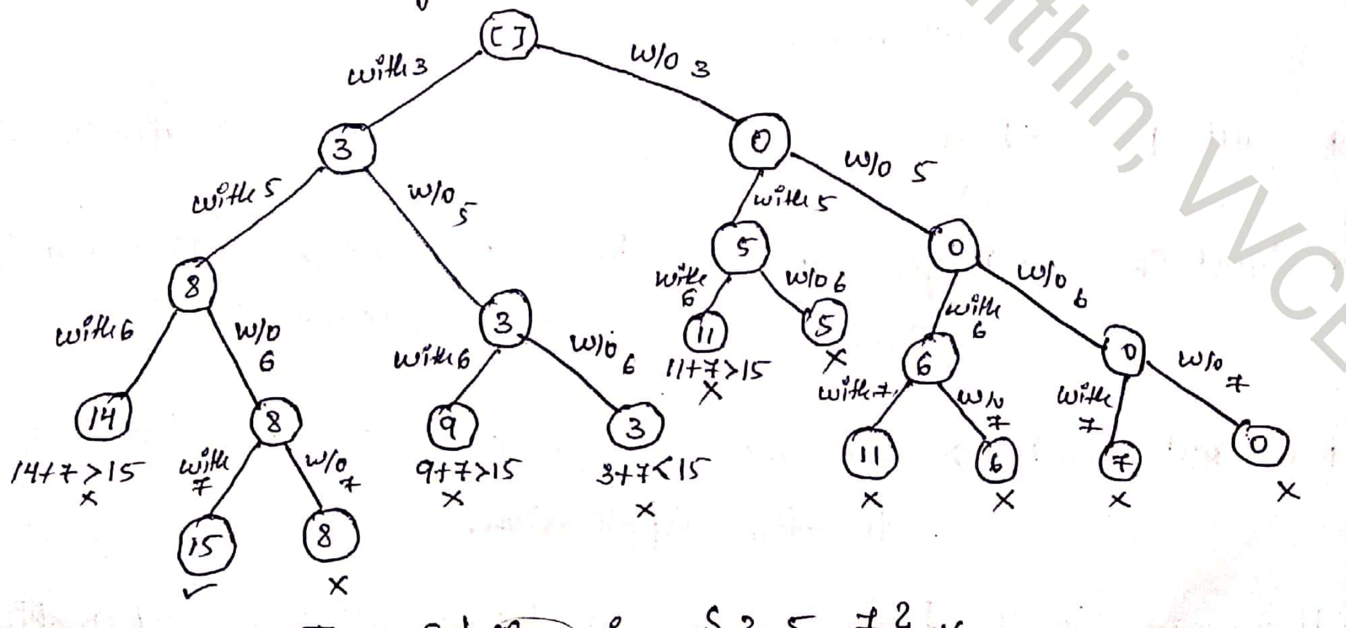
* Subset - Sum problem :- The task is to find a Subset of a given Set whose Elements add-up to a given Integer, that is 'd'.

→ Steps to Solve the given Sum-of-Subset problem
  * Initial Set only contains the Empty set.
  * We loop through Each Element in the Array & Add it to Every Element in power set.
  * We check to See if the Sum Equals to our goal 'd'.

* Consider the given $S = \{3, 5, 6, 7\}$ & $d = 15$, find the Subset for given 'S' using State-Space tree

⇒ Firstly check the Elements in the given Set S Must be Sorted
  * Value of First Element in S = 3 Must be less than d
  * Sum of all Elements in S Must be greater than d

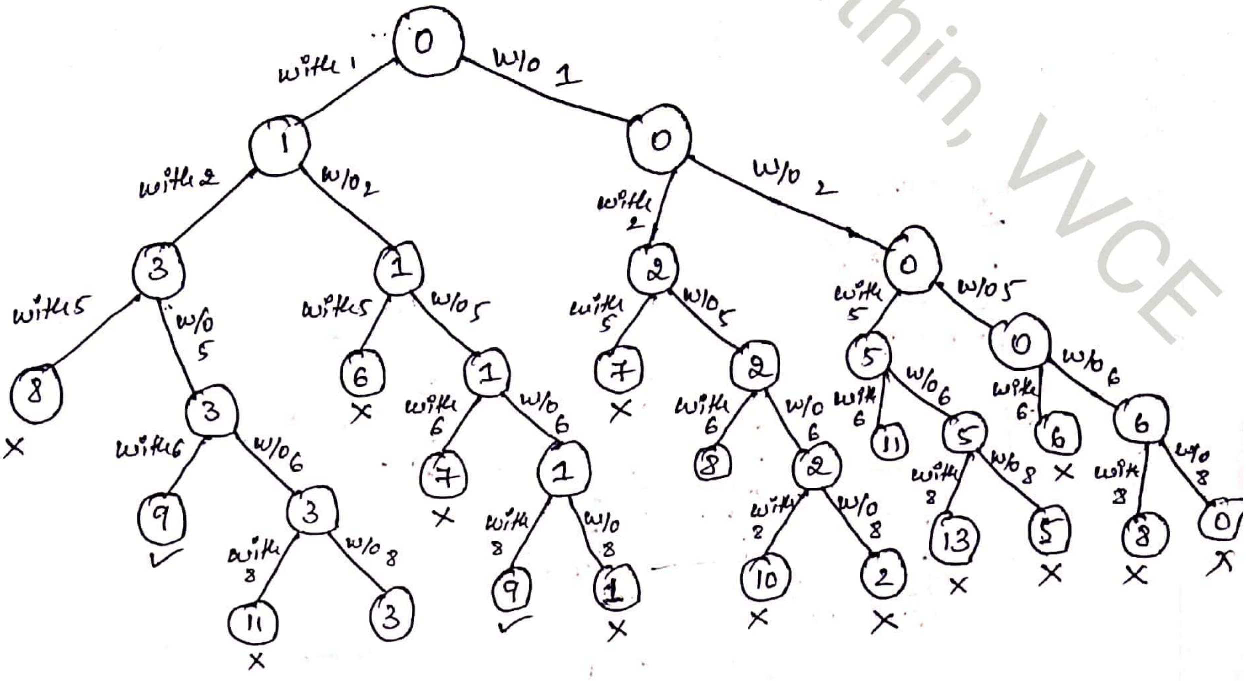
→ Start Constructing the State-Space tree



∴ The Solution is $\{3, 5, 7\}$

* Given $S = \{1, 2, 5, 6, 8\}$ & $d = 9$ find the Subsets of S with -9, State-Space tree

⇒ Firstly, Add all elements '22' should be greater than '9' & first Element $S[1] = '1'$ should be less than '9'



∴ The Solution is $\{1, 2, 6\}$ & $\{1, 8\}$ //

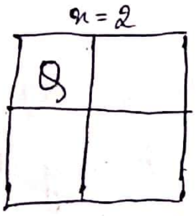
* N-Queen's problem :-

The problem is to place 'n' Queens on a $n \times n$ Chessboard So that no two Queens attack Each other by being in the Same row, Same Column or on Same diagonal

* For $n = 1$, the Solution would be $\boxed{Q}$

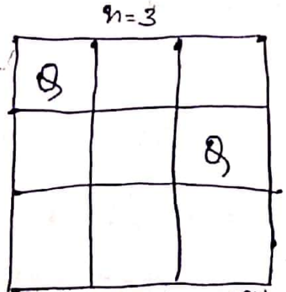
* For $n = 2$ & $n = 3$, there is no Solution because

$n = 2$



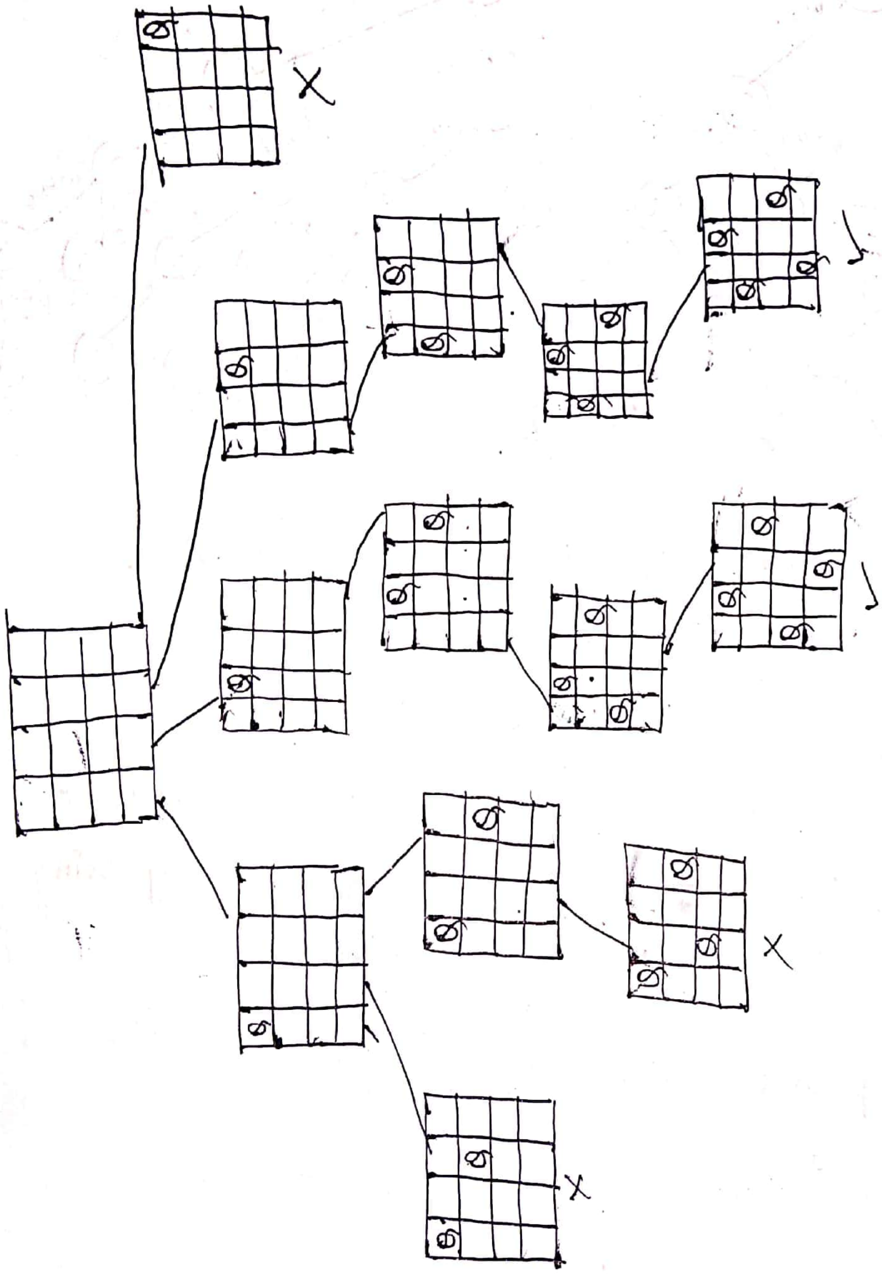
No place for 2^{nd} Queen

$n = 3$



No place for 3^{rd} Queen

* For n=4, there are two Solutions as Shown below



(4- Queen's problem)

* Branch & Bound :- is an improved version of Backtracking. here, we deal with optimization (Minimization or Maximization) problem.

→ A "feasible solution" is the one which satisfies the constraints of the problem.
For Ex, a subset of items whose total weight do not exceed the capacity of knapsack.

An "Optimal Solution" is a feasible Solution with the best value

→ Branch-and-Bound requires two additional items compared with backtracking

* For every node of a State Space tree, a way to provide a bound (either lower or upper) on the best value of the objective function.

* The Value of the Best soln found so far.

→ Here we will discuss 3 problems

* Assignment problem
* Knapsack problem
* Travelling Salesman problem

* Assignment problem :-

Here, the problem is to Assign 'n' Jobs to 'n' people so that the total cost of the Assignment is Minimum.

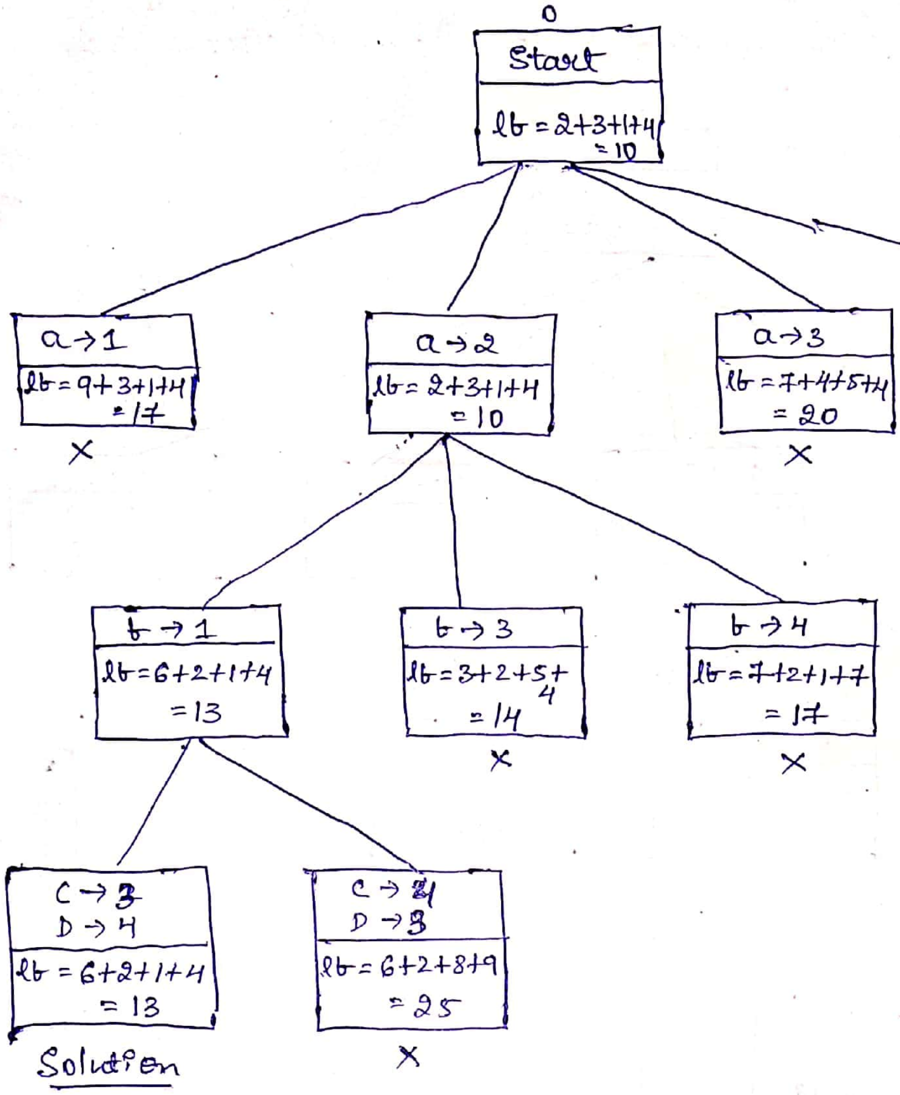
→ Consider the foll Cost Matrix

Jobs / Person	J_1	J_2	J_3	J_4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

⇒ Firstly, Compute the lower bound by adding the Smallest Elements in Every row.

* lower Bound would be (lb) = 2 + 3 + 1 + 4 = 10 //

→ Now at Every Step, we have to keep finding Minimal Soln without Violating the Constraints of the problem



$$A\begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

$$B\begin{bmatrix} \cancel{9} & \cancel{2} & \cancel{7} & \cancel{8} \\ 6 & \cancel{4} & 3 & 7 \\ 5 & \cancel{8} & 1 & 8 \\ 7 & \cancel{6} & 9 & 4 \end{bmatrix}$$

$$\begin{array}{c} \\ \\ C \\ D \end{array}\begin{bmatrix} \cancel{9} & \cancel{2} & \cancel{7} & \cancel{8} \\ \cancel{6} & \cancel{4} & \cancel{3} & \cancel{7} \\ \cancel{5} & \cancel{8} & 1 & 8 \\ \cancel{7} & \cancel{6} & 9 & 4 \end{bmatrix}$$

→ ∴ the cost of given Assignment problem is '13' & the Job Allocation would be

person A → Job 2
person B → Job 1
person C → Job 3
person D → Job 4 //

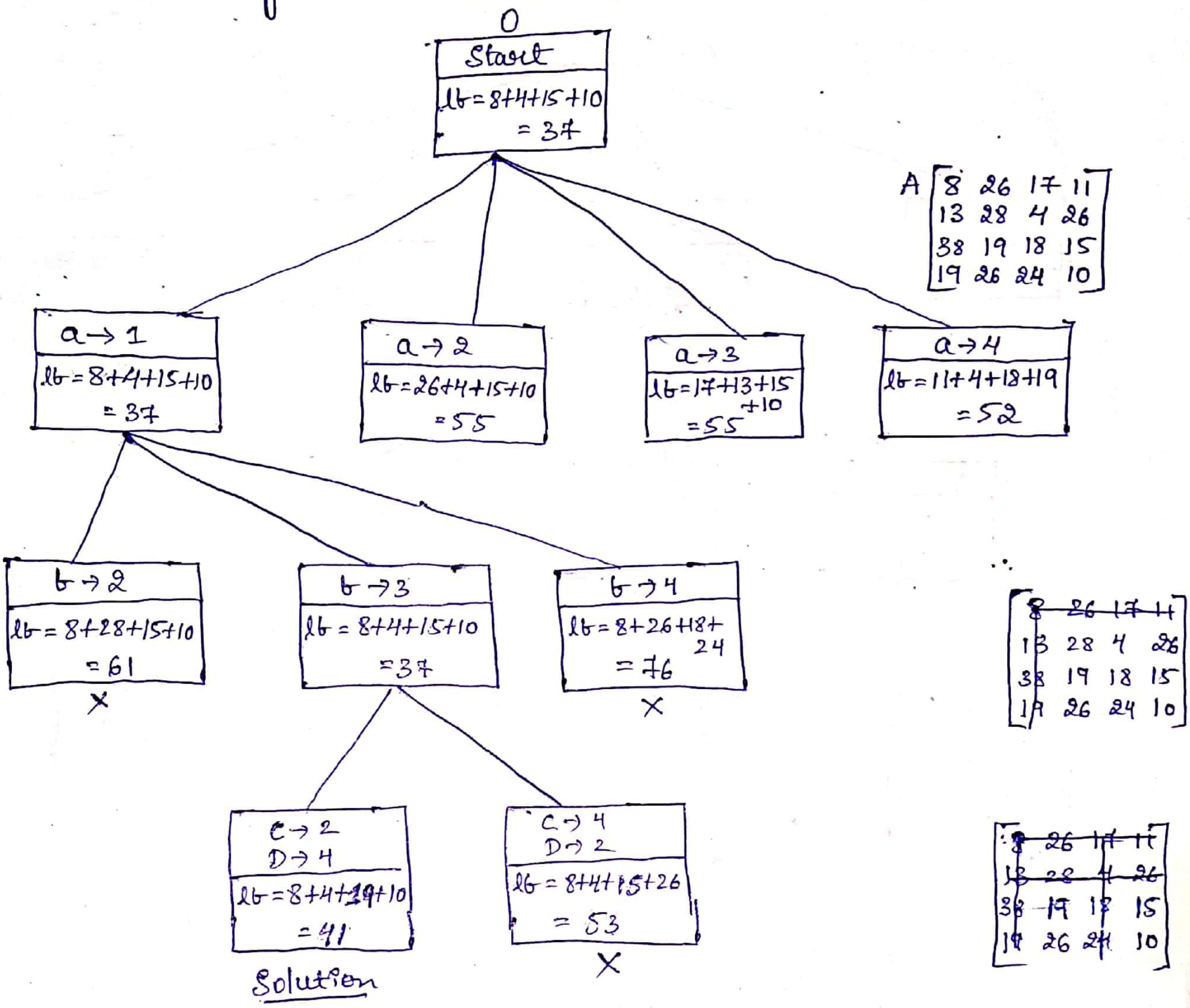
* Solve the given Assignment problem

$$\begin{bmatrix} 8 & 26 & 17 & 11 \\ 13 & 28 & 4 & 26 \\ 38 & 19 & 18 & 15 \\ 19 & 26 & 24 & 10 \end{bmatrix}$$

⇒ Firstly, Compute the lower bound by adding the Smallest Elements in Every row.

$\ast$ lower bound (lb) = 8+4+15+10 = 37 //

→ Now at Every step, we have to keep finding Minimal Soln without Violating the Constraints of the problem.



→ ∴ The cost of given Assignment problem is "41" & the Job Allocation would be

Person A → Job 1

person B → Job 3

Person C → Job 2

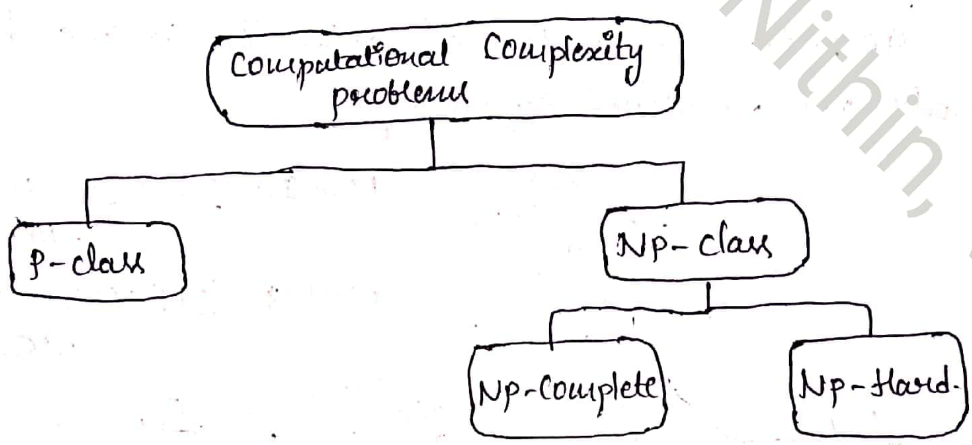
person D → Job 4 //

* P, NP, NP-Complete & NP-Hard Classes :-

The Algorithm in which Every operation is uniquely defined is called "Deterministic Algo".

→ The Algorithm in which Every operation May not have unique result, rather there can be Specified set of possibilities for Every operation Such an Algorithm is called "Non-Deterministic Algo".

→ There are two groups in which a problem can be classified



* **P-class** :- Class-P is a class of decision problems that can be Solved in "polynomial" time by Deterministic Algorithms. This class of problems is called "polynomial".

→ Some of the class-P Algorithms include Sorting, Searching, GCD, Multiplication of two Integers Etc.
There are Algorithms for which no polynomial time Algo has been found & hence Cannot be Categorized as class-P are TSP problem, Knapsack problem, Hamiltonian Circuit Etc.

* **NP-class** :- Class-NP is a class of decision problems that can be Solved by Non-deterministic polynomial Algo. This class of problems is called "Non-deterministic polynomial (NP)" Class.

→ All the problems which are of class-P are also included under the class-NP.

However, Np also Contain knapsack problem, TSP problem, Hamilto-nian cycle Etc.

* Np-Complete problems :- A decision problem 'D' is said to be Np-complete if
    * It belongs to class - Np
    * Every problem in Np is polynomially reducible to D.

* Np-Hard problems :- A problem is Np-Hard, if an Algorithm for Solving it can be translated into one for solving any Np-problem.
  Np-Hard therefore Mean "atleast as hard as any Np-problem"

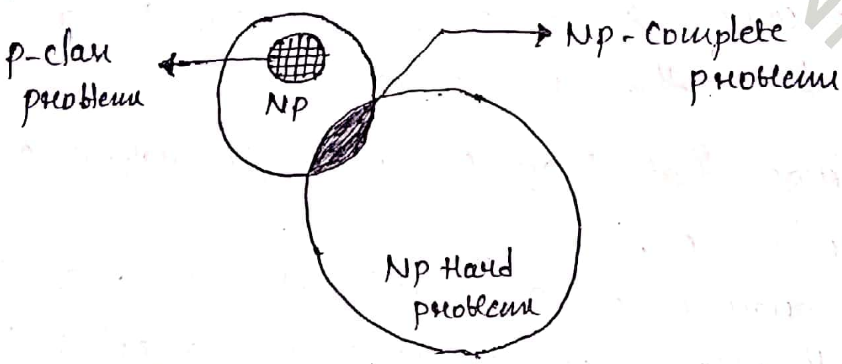


Fig: Relationship b/w P, Np, Np-Complete & Np-Hard

* Challenges of Numerical Algorithms :-
  Numerical Algorithms refers to Such Algorithms that are used for Solving Mathematical problems Such as
    * Evaluating Sin x, log x Etc
    * Evaluating Integrals Etc

However, numerous challenges are Encountered while Solving Mathematical problems Such as

* Most Numerical problems Cannot be Solved "Exactly", they

have to be Solved "Approximately". This is usually done by repl-acing an Infinite object by a finite Approximation.

Ex: $e^x \simeq 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!}$

* Due to Such Approximations, "Truncation Errors" would Occur. One of the Major challenges in Numerical Analysis is to Estimate the Magnitude of Truncation Error.
This is done using Calculus tools.

* Other type of Error that Could Occur is the "round-off Error". This is caused due to limited Accuracy while representing real nos in digital Computers.
Most Computers permit 3 levels of precision namely Single prec-ision, Double precision & Extended precision. Using Extended precision Slows down the Computation.

* Another challenge that May Occur is "overflow" & the "underflow" phenomenon.
An overflow occurs when an Arithmetic operation yields a result outside the range of Computer floating point No.
Underflow Occurs when an Arithmetic operation yields a result of Such a Small Magnitude that cannot be represented.

Nithin, VVCE

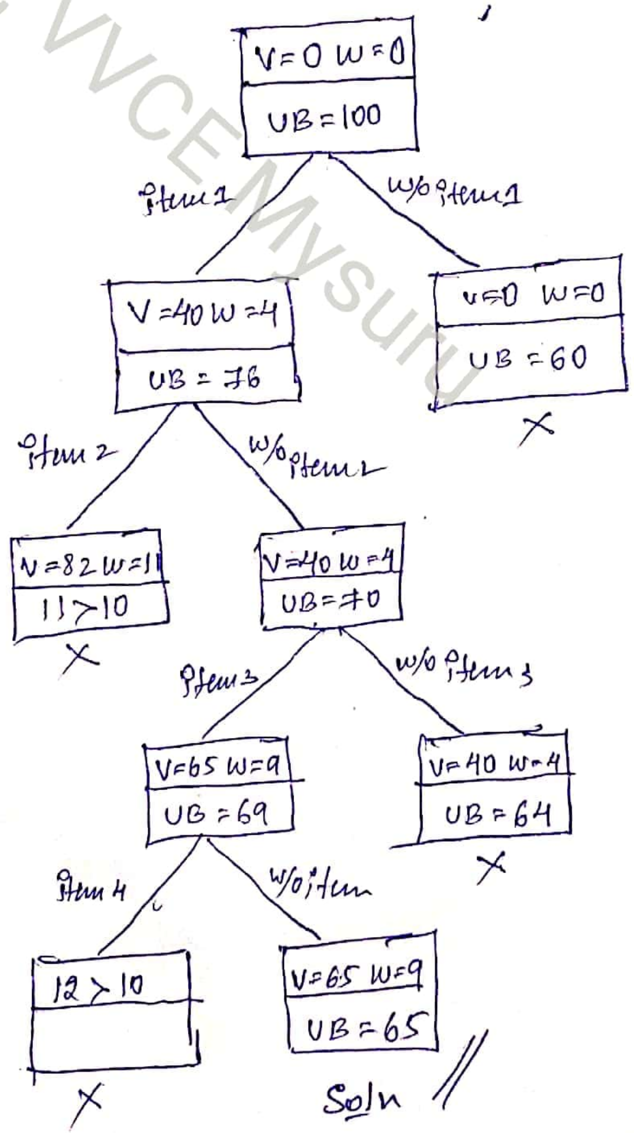
0/1 knapsack using Branch & Bound :-

$$UB = v + (M - w) * \frac{v_{i+1}}{w_{i+1}}$$

v → profit, M → Capacity, w → weight of item, v_{i+1}/w_{i+1} → ratio

→ M = 10 w = {4, 7, 5, 3} V = {40, 42, 25, 12}

Item	weight	value	$P.V_i/w_i$
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4



UB = 0 + (10 − 0) * 10 = 100

UB = 40 + (10 − 4) * 6 = 76

UB = 0 + (10 − 0) * 6 = 60

UB = 40 + (10 − 4) * 5 = 70

UB = 65 + (10 − 9) * 4 = 69

UB = 40 + (10 − 4) * 4 = 64

UB = 65 + (10 − 9) * 0 = 65

65 = {I_3, I_1} //

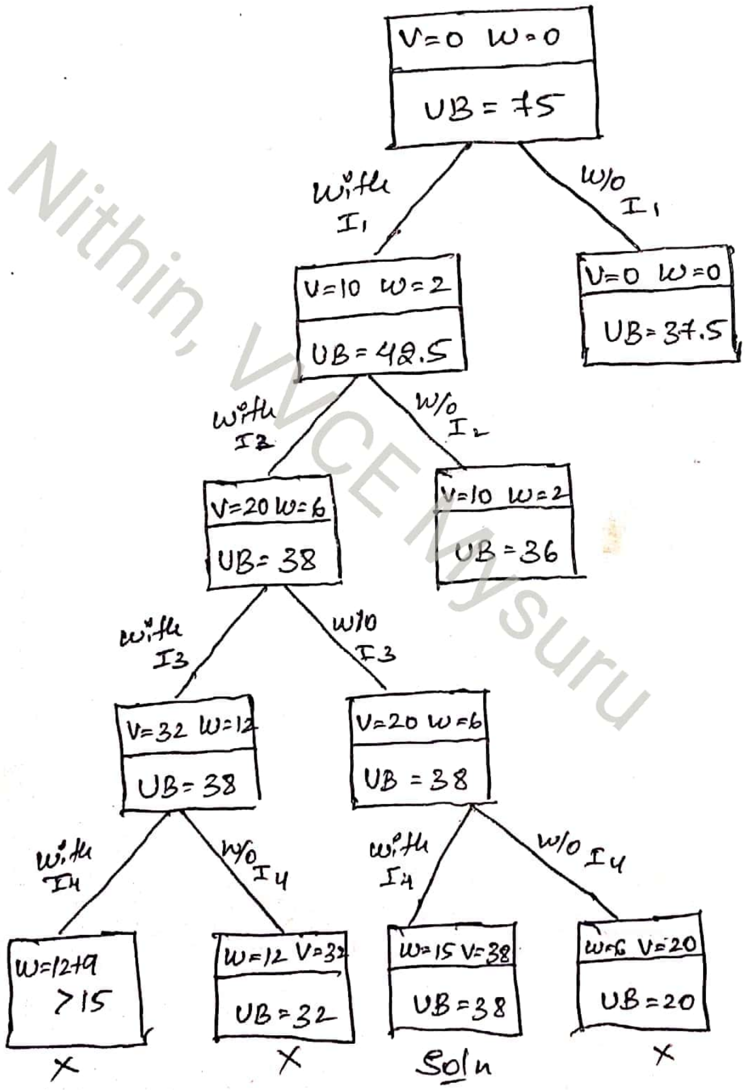
* Solve the given knapsack problem using Branch & Bound $V = \{10, 10, 12, 18\}$ $W = \{2, 4, 6, 9\}$ & the Maximum capacity of knapsack is $M = 15$.

→ Firstly, Compute V_i / w_i (ratio) & place it in decreasing order

item	Weight	Value	V_i/w_i
1	2	10	5
2	4	10	2.5
3	6	12	2
4	9	18	2

→ Compute upperbound & Construct State space tree

$$UB = V + (M - W) * V_{i+1}/w_{i+1}$$
$$= 0 + (15 - 0) * 5 = 75$$



$UB = 10 + (15-2) * 2.5 = 42.5$
$UB = 0 + (15-0) * 2.5 = 37.5$

$UB = 20 + (15-6) * 2 = 38$
$UB = 10 + (15-2) * 2 = 36$

$UB = 32 + (15-12) * 2 = 38$
$UB = 20 + (15-6) * 2 = 38$

$UB = 32 + (15-12) * 0 = 32$
$UB = 38 + (15-15) * 0 = 38$
$UB = 20 + (15-6) * 0 = 20$

∴ Total profit gained is "38"

items added are $\{I_1, I_2, I_4\}$