UNIT-5

Camlin Page
Date / /

BRANCH AND BOUND TECHNIQUE:

used for optimal soln — max & min value

Algorithm Design Technique used for solving optimization problems [mainly for minimisation problem]

→ As the algorithm problem, the tree of sub problems are found. The original problem is known as root problem.

→ At each node, we apply bounding function

→ If the bound function matches, feasible soln for the subproblem is obtained.

→ If bound function doesnot match, divide / partition the problem again into subproblems. Continue this process until all the subproblem are solved.

Back Tracking follows DFS (Depth first search) strategy. where as branch & bound follows BFS strategy (Breadth first search)

Mainly three types BB techniques:

- FIFO BB (first in first out branch & bound)
- LIFO BB
- LC BB (least cost BB)
- Out of 3, LC BB gives an optimal solution.

FIFO BB:

If the Data structure used for implementation of queue, we follow FIFO BB

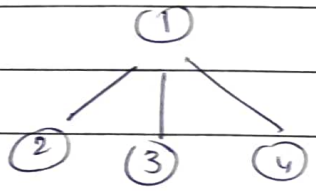
example:

Consider travelling Sales person problem with starting vertex ①

If we want to visit the other vertex 2, 3, 4 then state space tree is shown below

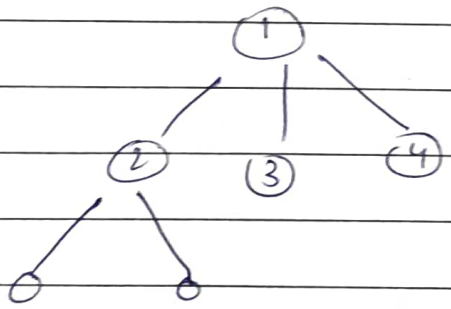
BB follows BFS.

So 1st level of the tree is



|2|3|4| Queue.

Now in next level, we need to explore node ② first, since that is 1st item present in queue.

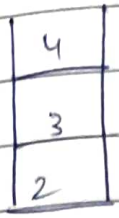


LIFO BB:

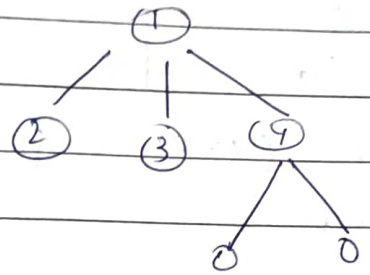
If Data structure used in Stack, we follow LIFO BB

In previous example, if newly generated node 2,3,4 is put into stack, the Stack looks as below:

2, 3, 4 is put into.

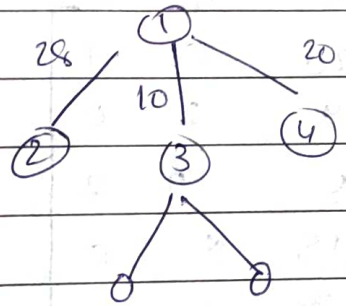


In the next level, we need to explore, node (4) since 4 is top most element in the stack



LC BB:

If cost is associated with each node we explore the node with min cost in further iteration



3 is explored since it is having least cost.

Applications:

TRAVELLING SALES PERSON PROBLEM:

	1	2	3	4	5	
1	∞	20	30	10	11	10
2	15	∞	16	4	2	2
3	3	5	∞	2	4	2
4	19	6	18	∞	3	3
5	16	4	7	16	∞	4

Solve travelling-sales person problem least cost branch and bound.

Reduce the matrix rowwise. find min value in each row & subtract & subtract value from each row element.

	1	2	3	4	5
1	∞	10	20	0	1
2	13	∞	14	2	0
3	1	3	∞	0	2
4	16	3	15	∞	0
5	12	0	3	12	∞
	1	0	3	0	0

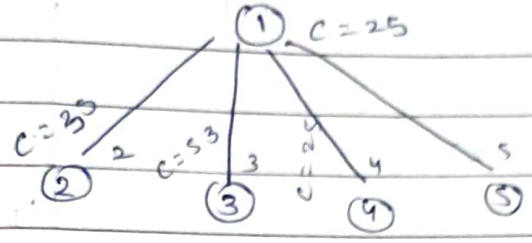
= 4.

Perform column reduction for the above matrix ie., find min value in each column & subtract the value from each column element.

	1	2	3	4	5	
1	∞	10	17	0	1	
2	12	∞	11	2	0	→ (I)
3	0	3	∞	0	2	
4	15	3	12	∞	0	
5	11	0	0	12	∞	

Cost of node 1 is, Σ row reduced cost + Given reduced Cost.

$$c(1) = \Sigma \text{ row reduced cost} + \text{Given reduced Cost}$$
$$= 21 + 4$$
$$= 25.$$



Now we need to find cost(1,2), c(1,3), c(1,4) & c(1,5). For this we will use Matrix I. After calculating the cost we explore or expand the node with min cost further.

Inorder to find the edge cost (1,2).
Set 1st row, 2nd column of Matrix-I to ∞ and edge(2,1) → ∞.
Check whether the matrix is reduced matrix or not.
If not we need to perform row wise, column wise reduction.

∞	∞	∞	∞	∞	
∞	∞	11	2	0	0
0	∞	∞	0	2	0 → II
15	∞	12	∞	0	0
11	∞	0	12	∞	0
0		0	0	0	

The matrix is already reduced.

Cost of node 2, $c(2) = c(1,2) + c(1) + r$ [new reduction value or Cost]
$= 10 + 25 + 0$
edge(1→2) = 35 calculated
M-I

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	12	∞	∞	2	0	0
3	∞	3	∞	0	2	0
4	15	3	∞	∞	0	0
5	11	0	∞	12	∞	
	11	0		2	0	

→ III → II

	1	2	3	4	5
∞	∞	∞	∞	∞	
1	∞	∞	2	0	
0	3	∞	0	2	
4	3	∞	∞	0	
0	0	∞	12	∞	

~~C(1,3) = C(1,3) + C(3)~~

C(1,3)

$C(1,3) = C(1,3) + C(1) + \hat{r}$

$= 17 + 25 + 11$

$= 53$

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	12	∞	11	∞	0	0
3	0	3	∞	∞	2	0
4	∞	3	12	∞	0	0
5	17	0	0	∞	∞	0
	0	0	0		0	

→ IV

$C(4) = C(1,4) + C(1) + \hat{r}$

$= 0 + 25 + 0$

$= 25$

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	12	∞	11	2	∞	2
3	0	3	∞	0	∞	0
4	5	3	12	∞	∞	3
5	∞	0	0	12	∞	0
	0	0	0	0		

$C(5) = C(1,5) + C(1) + \hat{r}$

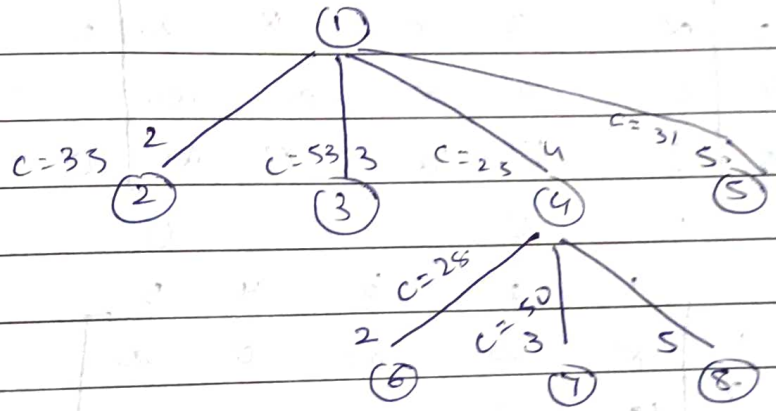
$= 1 + 25 + 2 + 3$

$= 31$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	10	∞	9	0	∞
3	0	3	∞	0	∞
4	12	0	9	∞	∞
5	∞	0	0	12	∞

— V

Since, Cost of node 4 c(4) is minimum. We expand this node further.



We need to find c(4,2), c(4,3), c(4,5) where matrix - IV is the base for this calculation.

Inorder, to find c(6) ie, c(4,2). Set 4th row, 2nd Column of matrix-IV to ∞ and e(2,1) in matrix-IV also ∞.

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	∞	∞	11	∞	0	0
3	0	∞	∞	∞	2	0 → VI
4	∞	∞	∞	∞	∞	
5	11	∞	0	∞	∞	0
	0		0		0	

$$c(6,2) = c(4,2) + c(4) + \hat{r}$$
$$= 3 + 25 + 0$$
$$= 28$$

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	12	∞	∞	∞	0	0
3	∞	3	∞	∞	2	2
4	∞	∞	∞	∞	∞	
5	11	0	∞	∞	∞	0
	11	0			0	

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	12	∞	∞	∞	0	0
3	∞	1	∞	∞	0	0 → VII
4	∞	∞	∞	∞	∞	
5	0	0	∞	∞	∞	0
	0	0			0	

$$c(7) = c(4,3) + c(4) + \hat{r}$$
$$= 12 + 25 + 11 + 2$$
$$= 50$$

37
13
50

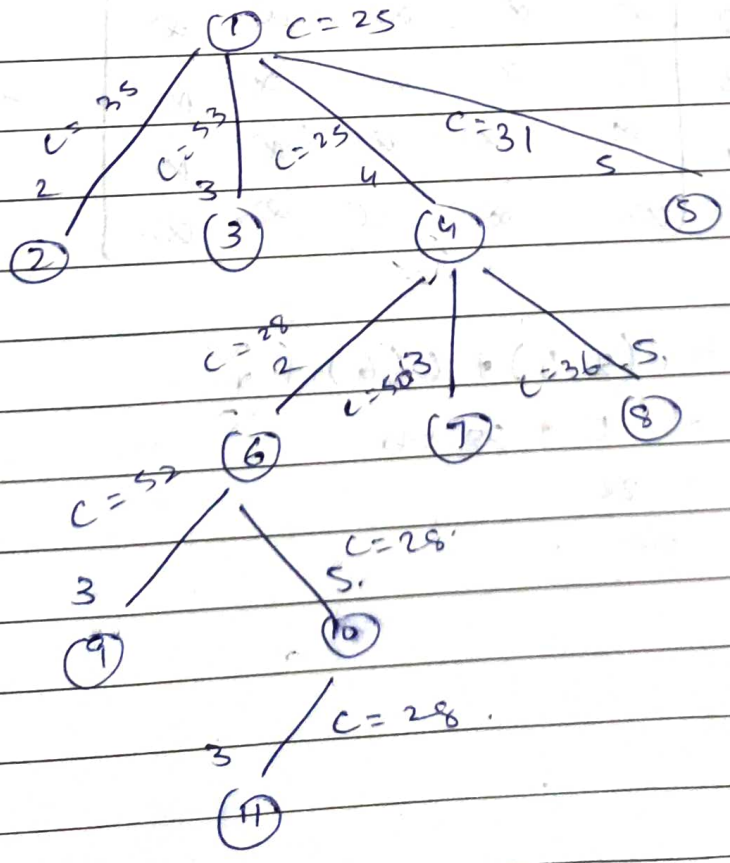
	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	1	∞	11	∞	∞	
3	0	3	∞	∞	∞	0
4	∞	∞	∞	∞	∞	
5	∞	0	0	∞	∞	0
	0	0	0			

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	
2	1	∞	0	∞	∞	0
3	0	3	∞	∞	∞	0 → VIII
4	∞	∞	∞	∞	∞	
5	∞	0	0	∞	∞	0
	0	0	0			

$$c(8) = c(4,5) + c(4) + \hat{r}$$

$$= 0 + 25 + 11$$

$$= 36.$$



	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	2
4	∞	∞	∞	∞	∞
5	11	∞	∞	∞	∞

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

→ IX

$$C(9) = C(2,3) + C(6) + \hat{r}$$

$$= 11 + 28 + 11 + 2$$

$$= 52$$

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

→ X

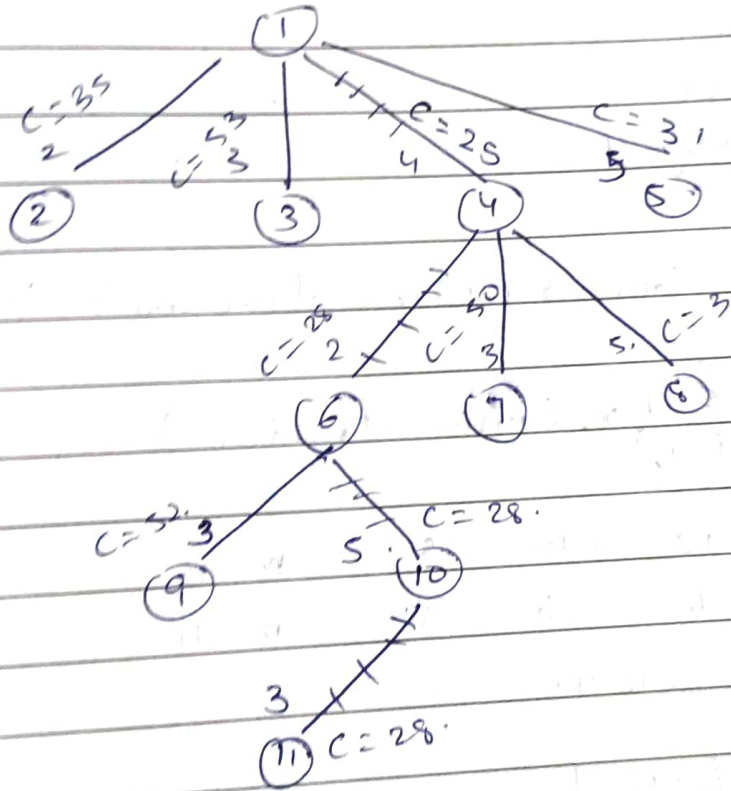
$$C(10) = C(2,5) + C(6) + \hat{r}$$

$$= 0 + 28 + 0$$

$$= 28$$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞

$$C(1) = C(5,3) + C(1,0) + r$$
$$= 0 + 28 + 0$$
$$= 28$$
$$\Rightarrow$$



Path: 1 — 4 — 2 — 5 — 3 — 1

Cost : 28

$$\Rightarrow$$

Application:

0/1 knapsack problem:

It is a maximization problem. Since, branch & bound is used for minimization, we convert the problem into a minimization by adding a -ve sign.

At each level, for each node we calculate two values - upper bound (without including fraction)

$$U = \sum_{i=1}^{n} P_i x_i \text{ (without fraction)}$$

P → profit

x → object

Lower bound (with fraction)

$$C = \sum_{i=1}^{n} P_i x_i \text{ (with fraction)}$$

P: -10 -10 -12 -18 m = 15

w: 2 4 6 9 n = 4

Cost of node 1.

$-18 \times \frac{3}{9}$ (2)	3/9		
-12	6	-12	6
-10	4	-10	4
-10	2	-10	2

$C = -10 - 10 - 12 + -6$

$= -38$

$U = -10 - 10 - 12$

$= -32$

$U = -32$

$L = -38$

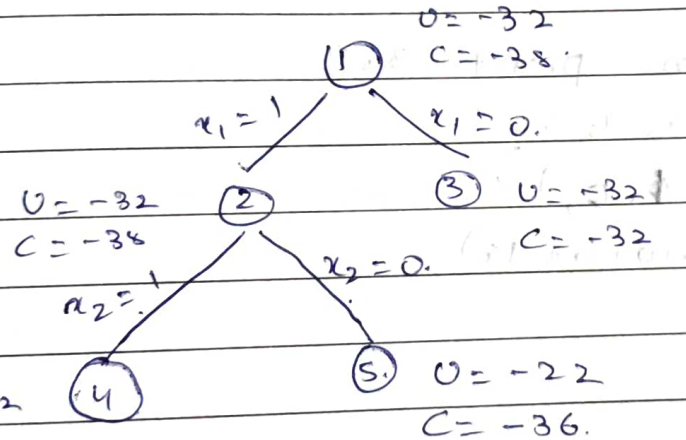
(1)

After each calculation check upper bound with global upper bound. [Global upper bound value is initialized to ∞].

Modify global upper bound with smaller upper bound. Since -ve, Consider v with higher numeric value

After each calculation if lower bound ie., c > Greater than Global upper bound, kill the node.

At each level, node with least ~~cost is~~ lower bound cost is expanded. further ie., c.



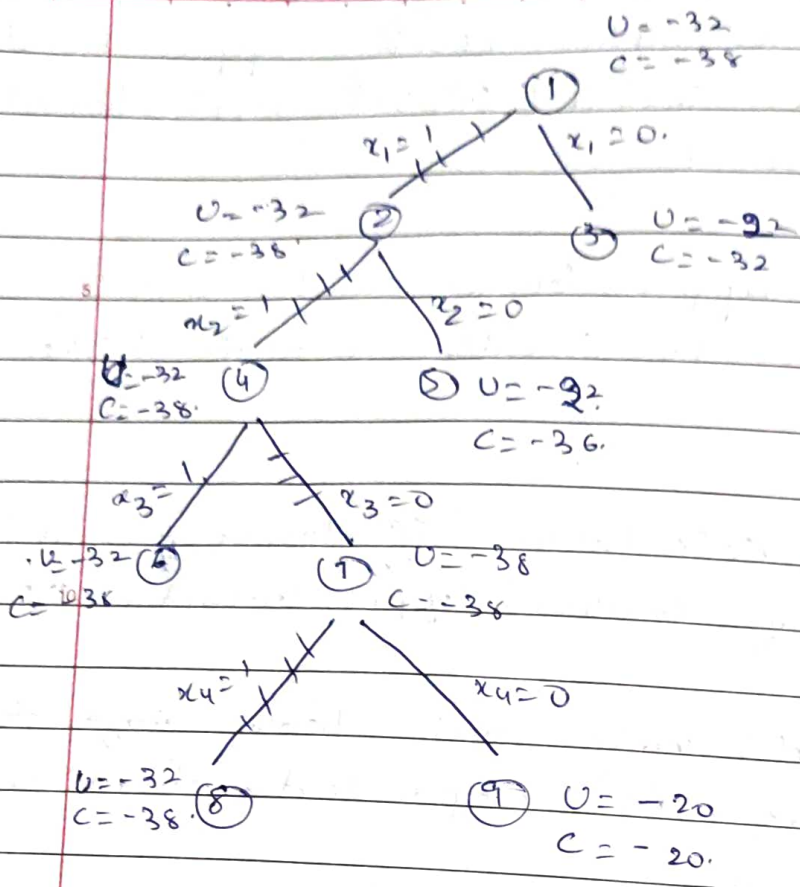
-15 7/9	7/9
-12	6
-10.	2

c = -10 + -12 + -4

= -36

v = -10 -12

= -22



GU = -32 -38

Maximum profit = 38

$x_1\ x_2\ x_3\ x_4$

(1, 0, 0, 1)

(10 + 10 + 0 + 18)

= 38

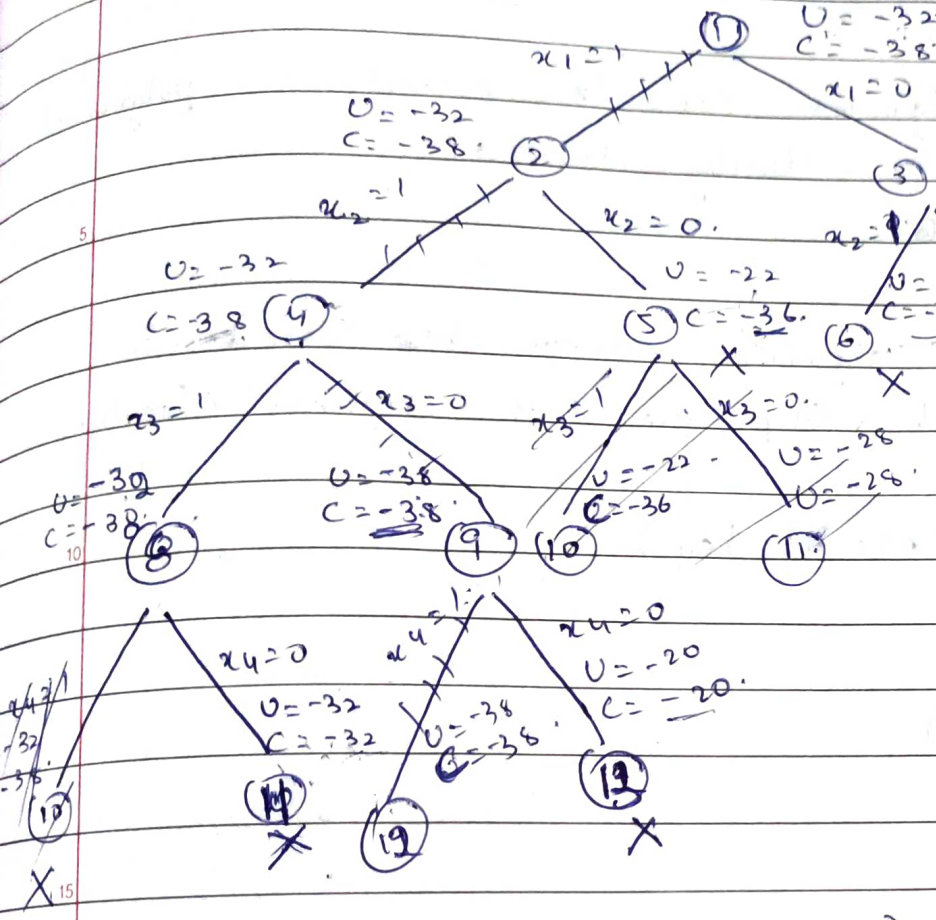
0/1 Knapsack Solution using FIFO Branch & Bound

P : 10 10 12 18

w : 2 4 6 9

n = 4

m = 15



x_1	x_2	x_3	x_4
1	1	0	1

Max profit = 10 + 10 + 18

= 38

1) Deterministic & Non-Deterministic Algorithm:

Deterministic algorithm can be solved in polynomial time. Each step is clearly defined.

ex:- linear search.

Non-Deterministic algorithm cannot be solved in polynomial time. Each steps are not defined clearly.

ex:

```
Algm Search(A[1:n], key)
{
  j = choice()
  if (A[j] = key)
  {
    Success();
    write A[j];
  }
  else
  {
    failure();
  }
}
```

Generally, non-deterministic algm include

* Choice() - which arbitarily choose one element from the set.
* Success() - signals as successful solution.
* failure() - signals as failure (or) unsuccessful solution.

Camlin Page
Date / /

Class-P and class-NP problems

Class-P (P → polynomial time)

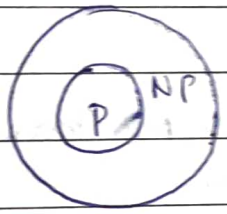
Problems under class-P can be solved and verified under polynomial time. It uses deterministic algorithm. The problems under class-P also known as tractable problem.

ex:- Searching, Sorting, etc.

Class-NP (

Takes exponential time for solving the problem and polynomial time for verification. It uses non deterministic algorithm. Problems under class-NP are also known as intractable problem

ex:- 0/1 knapsack, Hamiltonian cycle, travelling Sales-person problem.



$P \subseteq NP$

P is the subset of NP

The problems which are now in class-P were in NP before.

Satisfiability problem:

If we are not able to solve exponential time taking algorithm in polynomial time find the relation between the exponential time taking problems so that if one problem is solved in polynomial time the other also can be. In order to relate the problems, we use a non-NP hard ~~empty~~ hard problem which is known as Satisfiability problem.

Conjuctive Nomal Form (CNF):-

Diffent clauses connected using conjuction. Within the clause disjunction is used. If the clause consist of 3 literals is called as 3-CNF.

ex:- $(x_1 \cup \bar{x}_2 \cup x_3) \cap (\bar{x}_1 \cup x_2 \cup \bar{x}_3)$

If Satisfiability problem is reduced to P_1

$$Sat \propto P_1$$

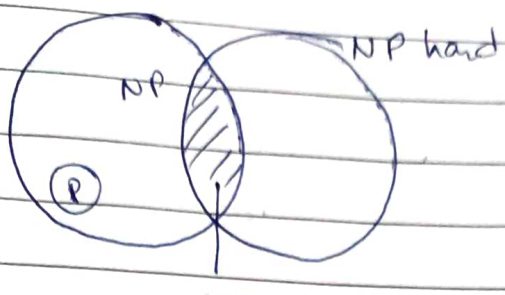
Since Satisfiability is NP-hard problem. P_1 is also an NP-hard problem.

$P_1 \propto P_2$ then P_2 is also an NP-hard problem.

- A non deterministic algorithm is already present for a satisfiability problem.

- For any NP-hard problem, if we are able to find non deterministic algorithm, then problems are called NP-Complete problem

NP hard +



NP - Complete.

COOKS THEOREM :-

If we prove that satisfiability problem is in P-class then P = NP