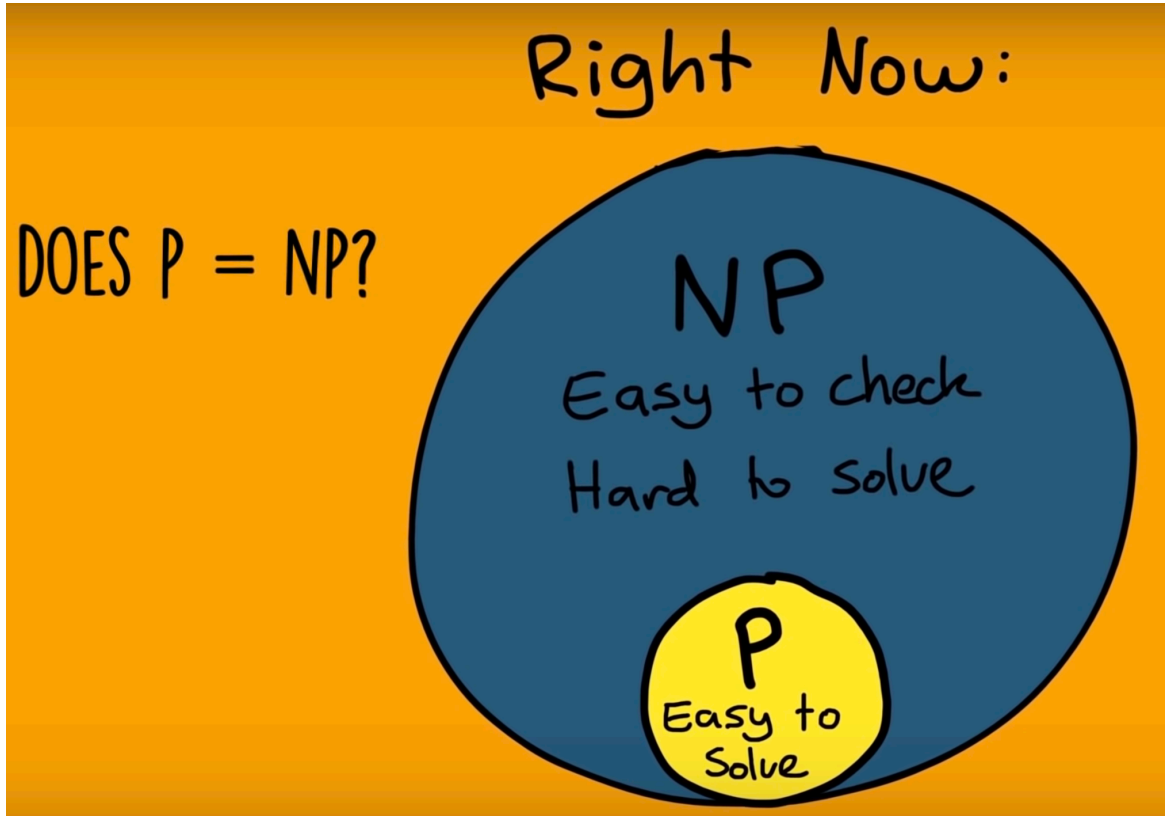
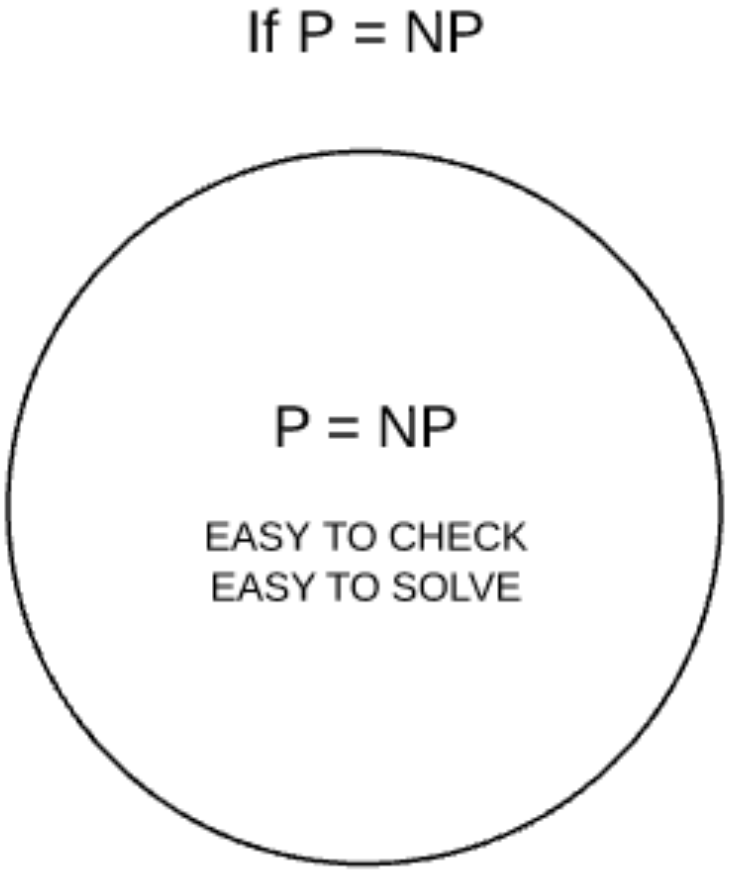
P Versus NP Complexity Theory

- Anuja Nagare

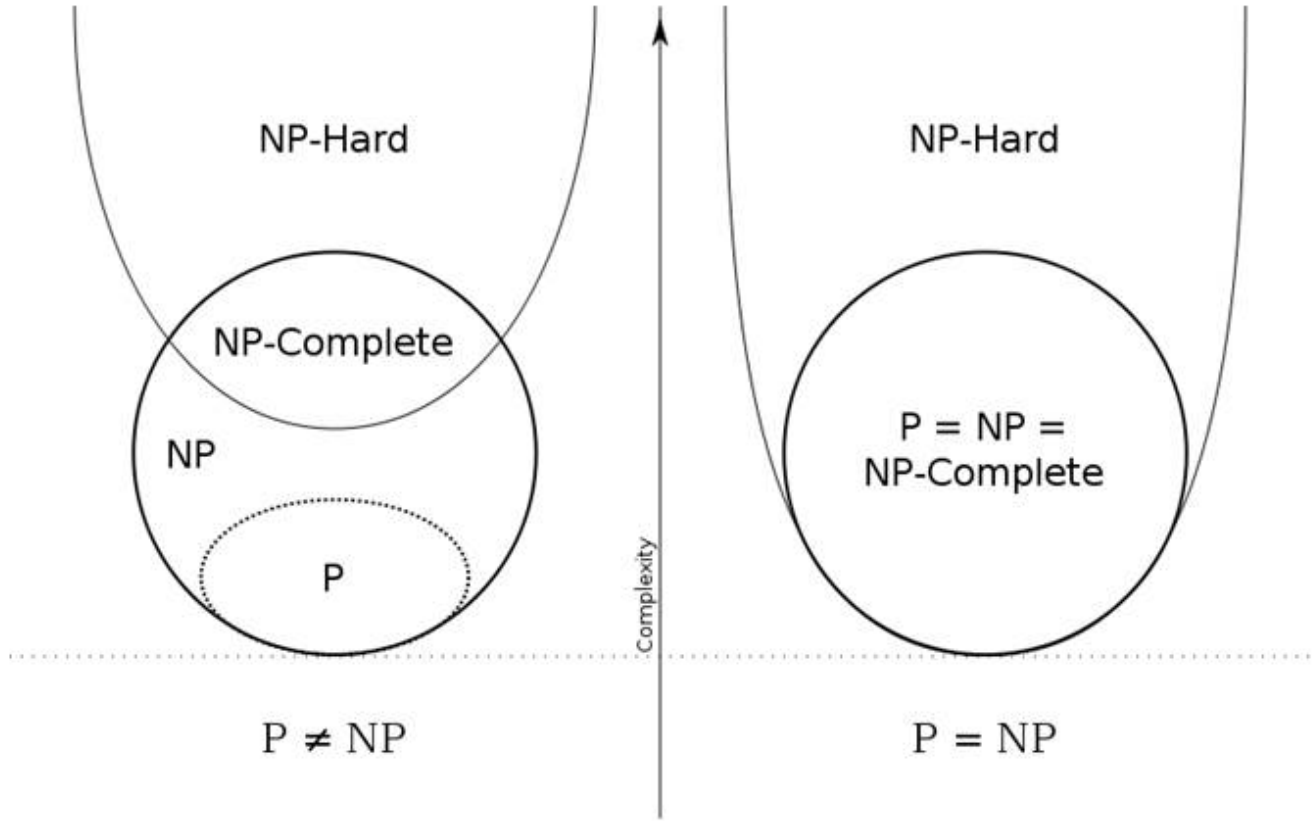


Are questions that are verified in polynomial time also solved in polynomial time ??



P Versus NP Complexity Theory

The **P versus NP problem** is a major unsolved problem in computer science. Informally, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer.



P: Polynomial Time - Algorithm running time is upper bounded by polynomial expression in the size of the input for the algorithm … e.g. it's "feasible", "efficient", "fast", such as Quicksort and all basic arithmetic operations

NP: Can only be solved in Non-deterministic Polynomial–time, such as determining whether two graphs can be drawn identically
NP-Complete: No fast solution is known, such as the Knapsack problem
NP-Hard: At least as hard to solve as the hardest problems in NP – an example is the Traveling Salesman Problem

Polynomial time		Exponential time	
Linear search	n	0/1 Knapsack	2^n
Binary search	log n	Travelling sales person	2^n
Ins	n^2	Sum	n
M	log n	Grap	2^n
Matrix multiplication	n^3	Hamiltonian C	2^n

Easy to solve

Easy to check

Framework for research on Exp time problems

- NP-Hard and NP-Complete

- Write polynomial time non-deterministic algorithm

- Solve / Try to relate problems :
  - Show similarity/relation/association between problems

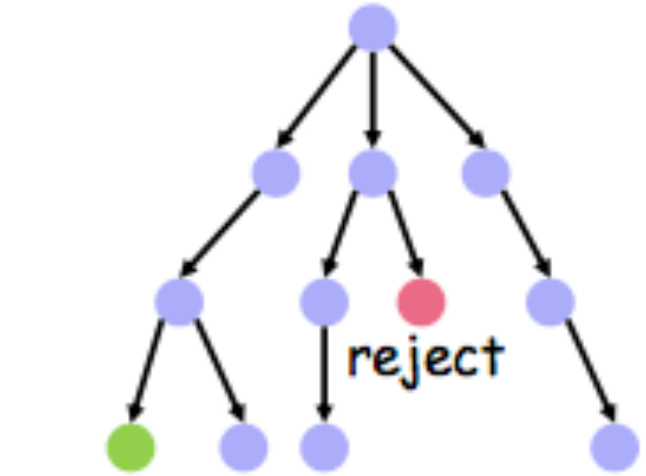
Reference link: https://www.youtube.com/watch?v=e2cF8a5aAhE

Write polynomial time non-deterministic algorithm

deterministic computation

accept or reject

nondeterministic computation



accepts if some branch reaches an accepting configuration

Nondeterministic

```
Algorithm NSearch(A, n, key)
{
    j = choice();
    if (key = A[j])
    {
        write(j);
        Success();
    }
    write(0);
    Failure();
}
```

Solve / Try to relate problems

- Step 1 : find a Base NP hard problem

- Step 2:
  - Try to relate your NP hard problem with base NP hard problem using reduction

Reduction Notation

Reduction

Denoted by

$Y \leq_p X$

or

$Y \longrightarrow X$

Two ways of using $Y \leq_p X$

1) X is easy
If we can solve X in polynomial time,
we can solve Y in polynomial time.

2) Y is hard
Then X is at least as hard as Y

Reduction Notation

$$Y \leq_p X$$

If we can solve X, we can solve Y.

Negate this statement.

If we cannot solve Y, we cannot solve X.

We use this to prove NP completeness: knowing that Y is hard, we prove that X harder.

In plain form: X is at least as hard as Y.

Note, we reduce TO the problem we want to show is the harder problem.

- Satisfiability problem is NP Hard problem

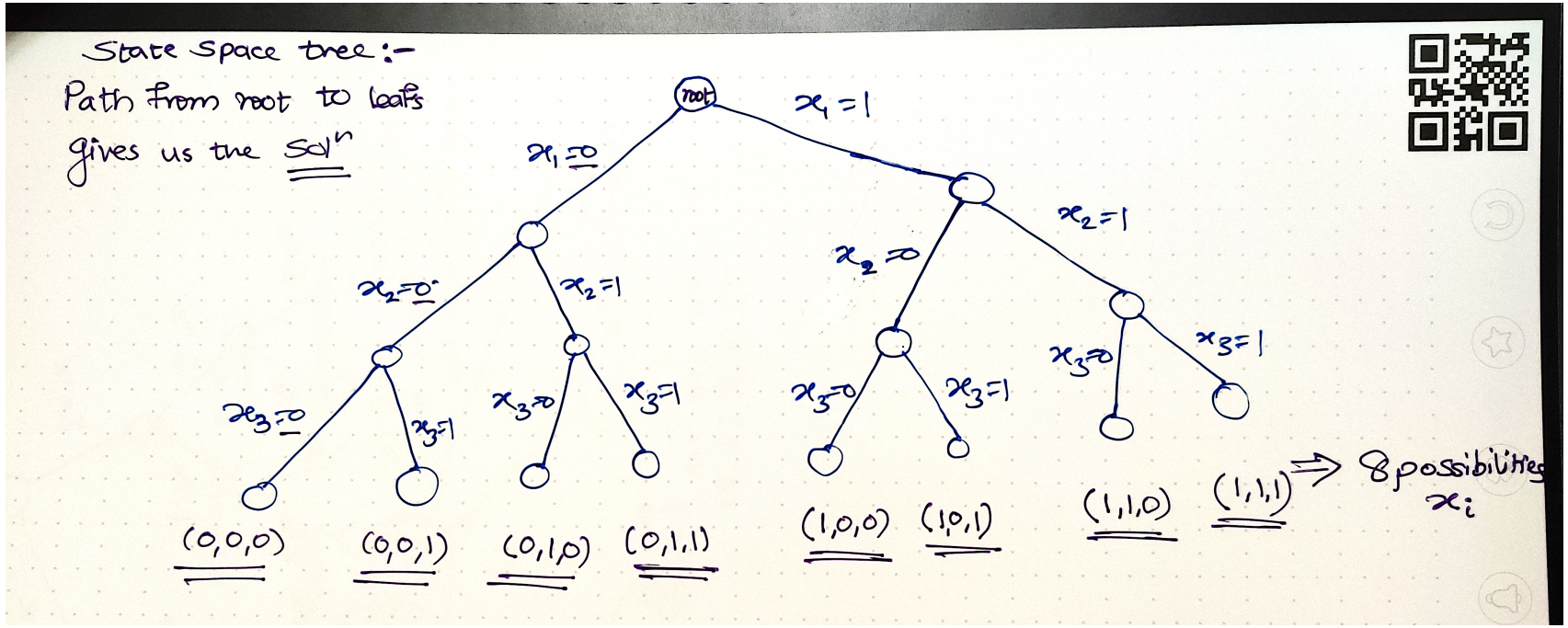
- Show that : Satisfiability reduces to 0/1 Knapsack problem
  - Satisfiability $\leq_p$ 0/1 Knapsack problem

  - Steps :
    - Take 1 example CNF Satisfiability formula
    - Take 1 example of 0/1 Knapsack problem
    - Show how similar technique can be used to solve CNF Satisfiability and 0/1 Knapsack problem
    - Then state : As both can be solved in similar way, both problems are NP Hard

- **Base Problem : CNF Satisfiability**
- $x_i = \{x_1, x_2, x_3\}$ → Boolean variables
- Eg. CNF formula => $\underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{Clause_1} \wedge \underbrace{(\overline{x_1} \vee x_2 \vee \overline{x_3})}_{Clause_2}$
- Satisfiability problem is to find out for what values of x_i given formula is true

Possible values of x_i ➔ 8 values → 2^3

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



0/1 Knapsack Problem

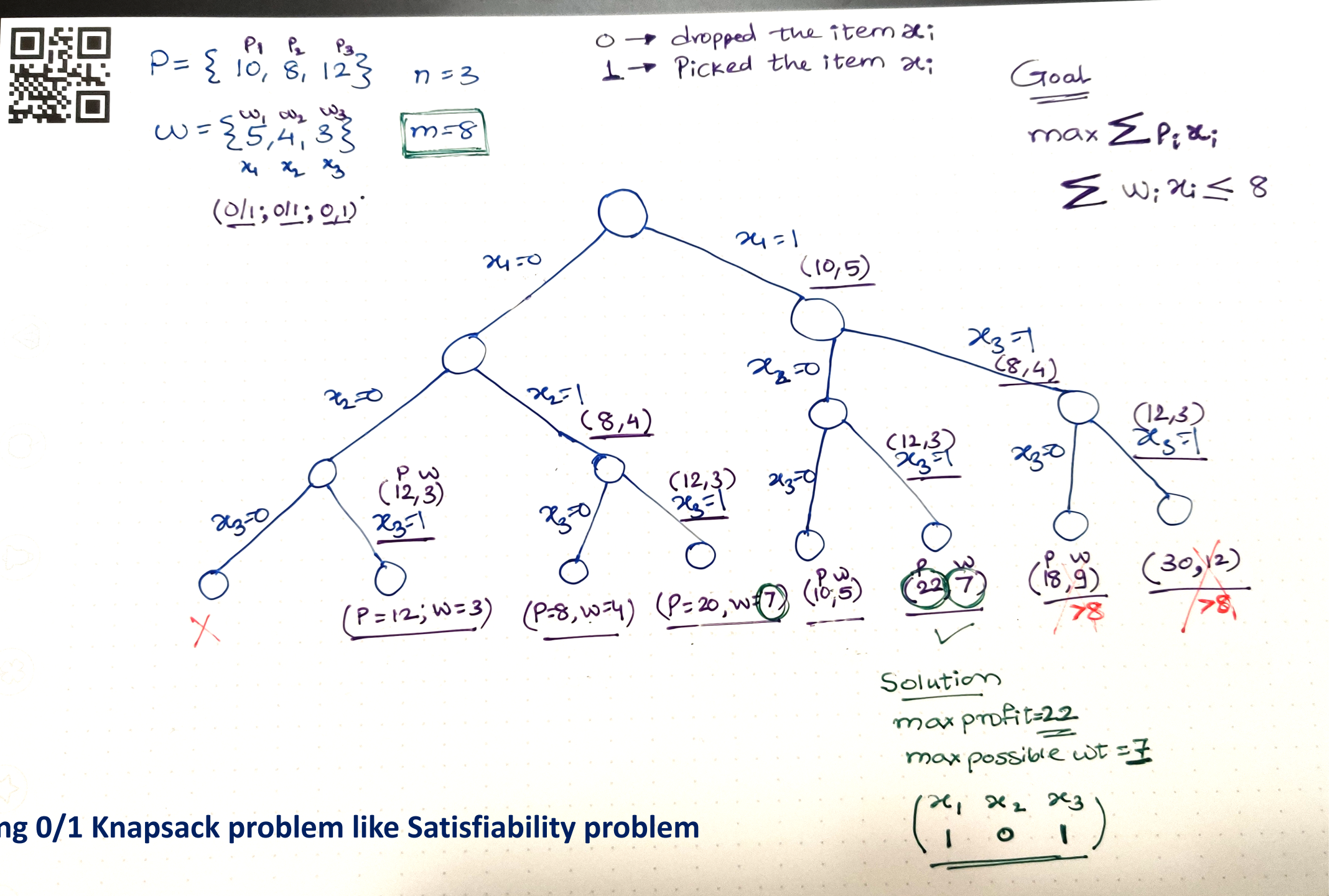
Given weights and profits of n items, Put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

- Given
  - Two integer arrays : Profit [0..n-1] and weight[0..n-1] which represent values and weights associated with n items respectively.
  - Weight which represents knapsack capacity,
- Find out :
  - Maximum profit subset of profit[] such that sum of the weights of this subset is smaller than or equal to W.

You can’t break an item, either pick complete item or don’t pick it (that’s 0/1 property).

Knapsack 0-1 Problem

- The goal is to **maximize the value of a knapsack** that can hold at most W units (i.e. lbs or kg) worth of goods from a list of items I_0, I_1, … I_{n-1}.
  - Each item has 2 attributes:
    1) Value – let this be v_i for item I_i
    2) Weight – let this be w_i for item I_i



Mapping 0/1 Knapsack problem like Satisfiability problem

Homework :

(try to solve in similar way by mapping 0/1 Knapsack problem like Satisfiability problem)

0/1 Knapsack Problem:

Weight= {2,3,4}
Profit = {1,2,5}
Knapsack max capacity weight **M=6**

Goal:
max $\sum p_i x_i$; $\sum w_i x_i \leq M$

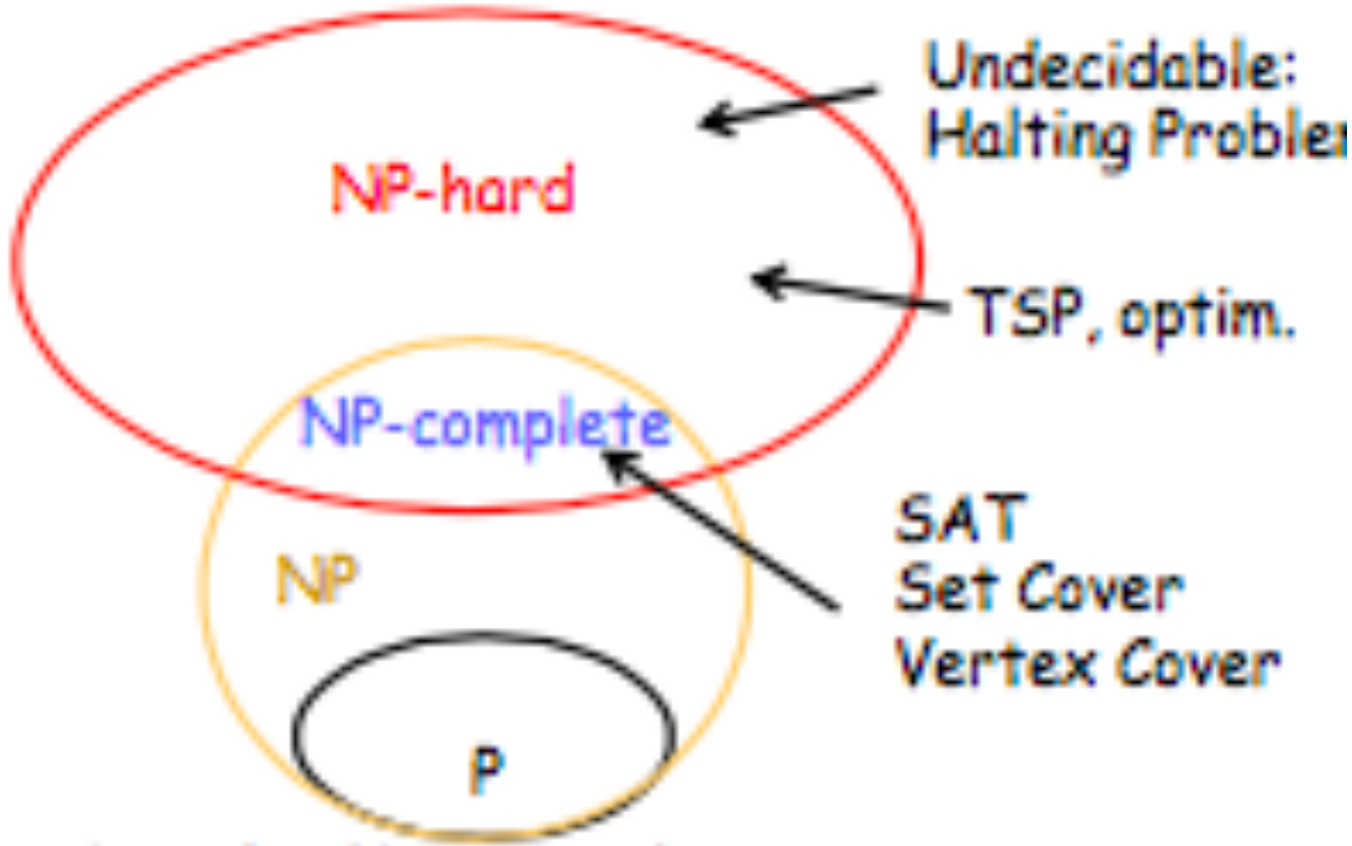
NP-Hard and NP-Complete

X is *NP-Hard*, if $\forall Y \in NP$ and $Y \leq_p X$.

X is *NP-Complete*, if X is NP-Hard and $X \in NP$.

Venn Diagram (P ≠ NP)

NPH problems do not have to be in NP.



NPC problems are the most difficult NP problems.

NPC problems can be solved by a *nondeterministic* TM in polynomial time.

It's not known if NPC problems can be solved by a *deterministic* TM in polynomial time.