Computational Complexity

- Deterministic Algo. (→ Polynomial time)
- Non- " "

- P, NP, NP-Hard, NP-Complete
- Cook's Theorem

Polynomial time Algo. → Merge Sort, Quick Sort, Linear search, ...

(2^n) Exponential " " → TSP, Graph coloring, 0/1 Knapsack

Research: Exponential Algo. → Polynomial Algo.

- Relate exponential time algo. in such a way that if one problem is solved in polynomial time then other problems can also be solved in polynomial time.
- Can't we find a NP (Non-deterministic polynomial time algo.) for these problems? if we don't find a deterministic polynomial time algo.

P: Set of deterministic algos which have polynomial time complexity.

NP: Set of non-deterministic algos. ..... polynomial

Some/All statements are non-deterministic.

Example:

```
Algorithm ND_Search (A, n, key)
{
  j := choice();  → ND
  if (key = A[j])
    {
      write(j);
      success();  → ND
    }
  write(0);
  failure();  → ND
}
```

key → Element to search

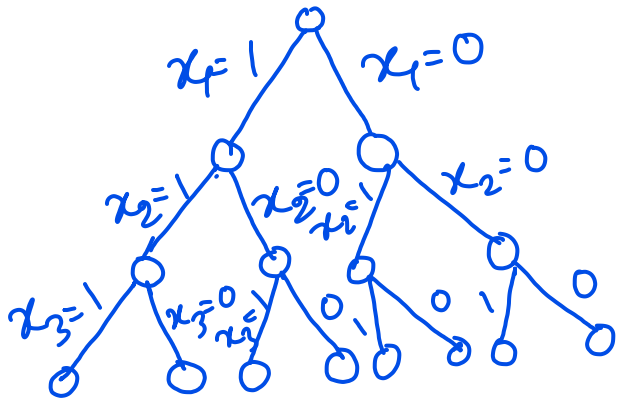
NP-Hard: Base problem

NP-Hard ← CNF - Satisfiability problem

$x_i = \{x_1, x_2, x_3\} \rightarrow 2^3$ $\quad n \Rightarrow 2^n$

CNF formula = $(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$

Aim: Find out the values of x_1, x_2, x_3 for which this CNF formula is true.



0/1 knapsack problem: x_1, x_2, x_3

State space tree: x_1 x_2 x_3 → 0/1 0/1 0/1

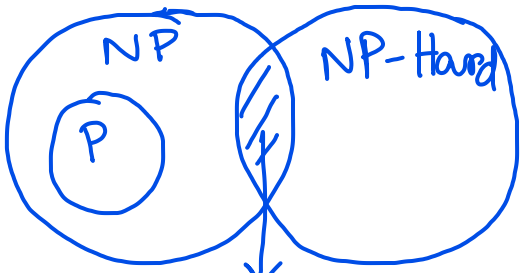
Reduction: ↓ Transitive property

Satisfiability (NP-Hard) —Reduce (Poly. time)→ X (NP-Hard) → Y (NP-Hard)

NP-Hard: A problem X is NP-Hard if a poly-time algo. for X would imply a poly. time algo. for every problem in NP.

NP-Complete: A problem is NP-Complete if it is both NP-Hard and an element of NP.

Relation:



NP-Complete
(CNF-Satisfiability)

Reduction: To prove that a problem is NP-Hard, we use reduction. Reducing problem A to another problem B means describing an algo. to solve prob. A under the assumption that an algo. for prob. B already exists.

Cook's Theorem / Cook-Levin theorem :

Boolean satisfiability problem is NP-Complete. i.e. it is in NP and any problem in NP can be reduced in poly. time by a deterministic Turing machine to the Boolean Satisfiability problem.