L₁ and L₂ are languages over some alphabet Σ.

L₁ is reducible to L₂ \( L₁ \leq L₂ \) if there exists a function \( f : Σ^* \rightarrow Σ^* \) s.t.

\[ x \in L₁ \iff f(x) \in L₂ \]

If \( f(x) \) can be computed in polynomial time, then \( L₁ \) is polynomial-time reducible to \( L₂ \).

**Claim:** If \( L₁ \leq \text{polynomial} \) \( L₂ \) and there is a polynomial-time algorithm to recognize \( L₂ \) then \( L₁ \) can also be recognized in polynomial time.
The algorithm to recognize $L$,

1. Given any string $x$, we first compute $y = f(x)$ in polynomial time (polynomial in $|x|$: length of string).

   $|y|$ is polynomial bounded by $|x|$

   $p_1$

2. We use $y$ as an input to the algorithm for $L_2$ and the running time is polynomial in $|y|$

   $p_2$

   If $y \notin L_2$ then return YES

   Else

   What is the running time

   $p_2 \left( p_1, (|x|) \right)$

   which is a polynomial function

   Claim: If $L_1 \leq_{poly} L_2$ and $L_2 \leq_{poly} L_3$

   $\Rightarrow$ $L_1 \leq_{poly} L_3$
\( g(y) \) is the mapping for \( L_1 \) to \( L_2 \) to \( L_3 \)

\( g(f(x)) \)

\( P \): class of polynomial time recognizable languages

\( NP \): class of languages recognizable in non deterministic polynomial time

(It is not the same as languages not in polynomial time)

\( P \subseteq NP \)

\( P = NP \iff P \subseteq NP \)
Hamilton cycle problem

Given a graph $G = (V, E)$, is there a cycle that visits all vertices exactly once.

Check all possible permutations for a legal tour.

Independent Set Problem

Given a graph $G = (V, E)$ and int $k$.

Is there an independent set of size $\geq k$?

Try all possible subsets of size $k$ and check independence.
Guessing a tour

Start with vertex 1

guess the next vertex

guess . . .

Then in a polynomial time, Verifier can test the property of Hamilton cycle.

Non-Hamilton

No independent set A size k
Knapsack

Decision version: Given n objects with weights \( w_1, w_2, \ldots, w_n \) and profits \( p_1, p_2, \ldots, p_n \) and knapsack of size \( B \), is there a subset to achieve profit \( K \)?

Knapsack \( \in \mathcal{NP} \)

Subset Sum Problem

Standard case: Given numbers
\[ x_1, x_2, \ldots, x_n \]
Can we partition them into two subsets \( S_1 \) and \( S_2 \) such that:
\[ \sum_{x \in S_1} x = \sum_{y \in S_2} y \]
\( S_1 \cap S_2 = \emptyset \)