1. **Polynomial-Time Reduction:**

- A polynomial-time reduction is a way to compare the computational difficulty of two problems. Specifically, it is a way to transform instances of one problem (K) into instances of another problem (L) in polynomial time.

2. **L is in P:**

- It is given that the problem L is in P. This means that there exists a deterministic Turing machine that can solve instances of L in polynomial time. In other words, L is a problem that can be efficiently solved.

3. **The Theorem Statement:**

- The theorem asserts that if problem K can be polynomial-time reduced to problem L, and L is efficiently solvable (in P), then problem K is also efficiently solvable (in P).

Here's an intuitive explanation of the theorem:

- When you say problem K is polynomial-time reducible to L, it means that you can use a polynomial-time algorithm to transform instances of K into instances of L. This transformation effectively allows you to solve instances of K by solving instances of L.

- Since L is in P, there is an efficient algorithm to solve it. If you can reduce K to L efficiently, it means that you can solve K efficiently as well. This is because you can transform an instance of K into an instance of L in polynomial time and then use the efficient algorithm for L to solve it.

In other words, if you can efficiently solve L, and you can efficiently transform instances of K into instances of L, then you can efficiently solve K. This theorem highlights the idea that problems that can be reduced to efficiently solvable problems remain efficiently solvable themselves. It's a fundamental result in computational complexity theory and helps in understanding the relationships between different complexity classes.