Theorem 9.1: If decision problem $B$ is in $P$ and problem $A$ reduces to problem $B$ ($A \preceq B$), then decision problem $A$ is in $P$.

Definition: a problem $B$ is called $NP$-complete if
1) $B$ is in $NP$, and
2) for every other problem $A$ in $NP$, $A \preceq B$.

1. If we find a polynomial-time algorithm for any $NP$-complete problem $B$, what can we conclude?

2. If
   • $B$ is known to be an $NP$-complete problem,
   • a problem $C$ is shown to be in $NP$, and
   • $B \preceq C$,
then what can we conclude? (Theorem 9.3)

Theorem 9.2: (Cook's Theorem 1971) CNF-Satisfiability is $NP$-complete.
(Proof uses common properties of $NP$ problems to show all of them reduce to CNF-SAT.)

$CNF$-Satisfiability Problem
$x_1, x_2, ..., x_n$ : Boolean variables
$\bar{x}_1, \bar{x}_2, ..., \bar{x}_n$ : complement of the Boolean variables
$\bar{x}_1, \bar{x}_2, ..., \bar{x}_n, x_1, x_2, ..., x_n$ : set of "literals"

A clause is the "or" ($\lor$) of a set of literals, e.g., $(\bar{x}_1 \lor \bar{x}_2 \lor x_4)$.

A Boolean expression is in $CNF$ (Conjunctive Normal Form) if it is the conjunction/"anding" ($\land$) of one or more clauses, e.g., $(x_1 \lor x_2) \land x_3 \land (\bar{x}_1 \lor \bar{x}_2 \lor x_4)$.

The CNF-Satisfiability Decision problem is to determine for a given CNF expression whether there is some truth assignment that makes the CNF expression "TRUE."

For the CNF-SAT Boolean formula $B = (x_1 \lor x_2) \land x_3 \land (\bar{x}_1 \lor \bar{x}_2 \lor x_4)$, the answer is "Yes" since the true assignment of $x_1$=TRUE, $x_2$=FALSE, $x_3$=TRUE, $x_4$=TRUE makes $B$ true.

Clique Problem
A clique in an undirected graph $G=(V, E)$ is a subset $W$ vertices such that each vertex $W$ has an edge to all other vertices of $W$ in the graph $G$.

Several cliques exist:
\{ $v_2, v_3, v_4$ \}
\{ $v_1, v_2, v_4, v_5$ \}

The optimization clique problem is to find a maximal clique, i.e., a clique with largest number of vertices.
The textbook shows the details of reducing CNF-Satisfiability Decision problem to the clique problem. So we have:

3. What can we conclude about the clique problem?

```
Any problem in NP
    ↓
CNF-SAT
    ↓
Clique Problem
```

Extend the discussion to non-decision problems

If problem A can be solved in polynomial time using a hypothetical polynomial-time algorithm for problem B, then A is polynomial-time Turing reducible to problem B. A $\alpha_T$ B.

```
Algorithm for A

<table>
<thead>
<tr>
<th>I_A</th>
<th>poly.-time transformation</th>
<th>I_B</th>
<th>Algorithm for B</th>
<th>O_B</th>
<th>transform of answer in poly.-time</th>
</tr>
</thead>
</table>

```

**Definition:** a problem B is called *NP-hard* if for some NP-complete problem A (a decision problem), A $\alpha_T$ B (A Turing reduces to B).

4. Why is the optimization problem corresponding to any NP-complete problem NP-hard?

**Handling NP-hard Problems**

1) Backtracking and Branch-and-Bound: Even though the worst case is still $\theta(2^n)$ or $\theta(n!)$, these often give efficient results for large problems.

2) Polynomial-time algorithms might exist for a subclass of NP-hard problem, e.g., for TSP with a restricted graph.

3) Approximation algorithm - give good, but not necessarily optimal solutions. Usually, the solution can bound how far from optimal its solution is.