This discussion will focus on showing that $\text{SUBSET-SUM}$ is NP complete. This is given in Theorem 7.56 in your book on page 292.

**SUBSET-SUM Problem**

The $\text{SUBSET-SUM}$ problem is defined on pages 268-269 of your book. Formally, it is defined as:

$$
\text{SUBSET-SUM} = \left\{ (S, t) | S = \{x_1, \ldots, x_k\} \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \sum y_i = t \right\}
$$

Informally, we have a set $S$ of numbers. Given a target number $t$, we want to know if there is a subset of $S$ which sums to $t$.

For example, suppose $S_1 = \{1, 15, -2, 44, 101\}$ and $t_1 = 100$. Is $(S_1, t_1) \in \text{SUBSET-SUM}$? Yes, there exists a subset $\{1, -2, 101\}$ such that $1 + (-2) + 101 = 100 = t_1$.

Both the sets $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_l\}$ are multisets, which allow repetition of elements.

As formulated here, it may not seem like the $\text{SUBSET-SUM}$ problem is interesting or important. However, forms of the $\text{SUBSET-SUM}$ problem show up in cryptography (and in many other fields). This problem is also related to the knapsack and partition problems. All of these problems have real-world applications (not just theoretical).

**Useful Tools**

There are several definitions, theorems, and results we will use to show this is true. We start with the definition of **NP-complete**.

**Definition 7.34**

A language $L$ is NP-complete if it satisfies two conditions:

1. $L \in \text{NP}$
2. Every $A \in \text{NP}$ is polynomial time reducible to $L$

To show that a language $L \in \text{NP}$, the following definition:

$\text{NP}$ is the class of languages that have polynomial time verifiers.

A **polynomial time verifier** is defined on page 265:
Definition 7.18
A verifier for a language $A$ is an algorithm $V$ where

$$A = \{ w | V \text{ accepts } \langle w, c \rangle \text{ for some string } c \}.$$  

A polynomial time verifier runs in polynomial time in the length of $w$.

If you know that (1) $L$ is in NP and (2) $A$ is NP-complete, you can use the following theorem:

**Theorem 7.36**
If $A$ is NP-complete and $A \leq_p L$ for some $L \in$ NP, then $L$ is NP-complete.

What does it mean for $A \leq_p L$? This brings us to the definition of a **polynomial time mapping reducibility**.

**Definition 7.28**
A function $f : \Sigma^* \rightarrow \Sigma^*$ is a polynomial time computable function if some polynomial time Turing machine $M$ exists that halts with just $f(w)$ on its tape, when started on any input $w$.

**Definition 7.29**
Language $A$ is a polynomial time mapping reducible to language $L$, written $A \leq_p L$, if a polynomial time computable function $f : \Sigma^* \rightarrow \Sigma^*$ exists, where for every $w$:

$$w \in A \iff f(w) \in L$$

Finally, we are going to need a language that we already know is NP-complete. The book uses the fact that 3SAT is NP-complete:

**Corollary 7.42**
3SAT is NP-complete.

The language is defined in your book on page 274 as:

$$3SAT = \{ \langle \phi \rangle | \phi \text{ is a satisfiable 3cnf-formula} \}$$

A **3cnf-formula** (conjunctive normal form-formula) is a Boolean formula that has several or-clauses with 3 literals each connected by and operations. For example:

$$(a \lor \overline{b} \lor \overline{c}) \land \cdots \land (\overline{x} \lor y \lor z)$$

**Proof Approach**

To show that SUBSET-SUM is NP-complete, we need to:

1. Show that SUBSET-SUM \in NP.
2. Show that 3SAT \leq_p SUBSET-SUM.

When we show the reduction, we’ll need to provide a polynomial time computable function $f$ and show that $\langle \phi \rangle \in 3SAT \iff \langle S, t \rangle \in SUBSET-SUM$.  

2
**SUBSET-SUM ∈ NP**

As pointed out in our “tool box” a language is in NP if it has a polynomial time verifier. Therefore, if we can provide a \( p \)-time verifier for **SUBSET-SUM**, we’ve shown it is in NP.

\[ V = \text{“On input } \langle S, t, c \rangle \text{:\n}\]
\[ 1. \text{ Test whether } c \text{ is a collection of numbers that sum to } t.\n\]
\[ 2. \text{ Test whether } S \text{ contains all the numbers in } c.\n\]
\[ 3. \text{ If both tests pass, } \textit{accept}.\n\]
\[ 4. \text{ Otherwise, } \textit{reject}.\n\]

This is given as the proof for Theorem 7.25 which states **SUBSET-SUM ∈ NP**.

**3SAT \leq_p SUBSET-SUM**

From this point on, please refer to my handwritten discussion notes from last year.