Discussion 10 Notes
Wednesday December 05, 2007

This discussion will focus on showing that *SUBSET-SUM* is NP complete. This is given in Theorem 7.56 in your book on page 292.

SUBSET-SUM Problem

The *SUBSET-SUM* problem is defined on pages 268-269 of your book. Formally, it is defined as:

$$SUBSET\text{-}SUM = \Big\{ \langle S, t \rangle \mid S = \{ x_1, \ldots, x_k \} \text{ and for some } \{ y_1, \ldots, y_l \} \subseteq \{ x_1, \ldots, x_k \}, \text{ we have } \sum y_i = t \Big\}$$

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 $\langle S_1, t_1 \rangle \in SUBSET\text{-}SUM$? 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 *SUBSET-SUM* problem is interesting or important. However, forms of the *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. $L \in$ NP
> 2. Every $A \in$ NP is polynomial time reducible to L

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

> 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

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

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 $A \leq_{\text{p}} L$ for some $L \in \text{NP}$, then L is NP-complete.

What does it mean for $A \leq_{\text{p}} L$? 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 $A \leq_{\text{p}} L$, 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:

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

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:

$$\big(a \vee \overline{b} \vee \overline{c}\big) \wedge \cdots \wedge (\overline{x} \vee y \vee z)$$

Proof Approach

To show that $SUBSET\text{-}SUM$ is NP-complete, we need to:

1. Show that $SUBSET\text{-}SUM \in \text{NP}$.
2. Show that $3SAT \leq_{\text{p}} SUBSET\text{-}SUM$.

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

$\boldsymbol{SUBSET\text{-}SUM} \in$ 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 $SUBSET\text{-}SUM$, we've shown it is in NP.

$V =$ "On input $\langle \langle S, t \rangle, c \rangle$:

1. Test whether c is a collection of numbers that sum to t.
2. Test whether S contains all the numbers in c.
3. If both tests pass, *accept*.
4. Otherwise, *reject*.

This is given as the proof for Theorem 7.25 which states $SUBSET\text{-}SUM \in$ NP.

$\boldsymbol{3SAT \leq_{\mathrm{p}} SUBSET\text{-}SUM}$

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