Problem

Let *UNARY-SSUM* be the subset sum problem in which all numbers are represented in unary. Why does the NP completeness proof for *SUBSET-SUM* fail to show *UNARY-SSUM* is NP-complete? Show that *UNARY-SSUM* ? P.

Step-by-step solution

Step 1 of 2

Let *UNARY-SSUM* be the subset sum problem in which all numbers are represented in unary.

The complexity of a function can be measured from the input size and number of steps required for that input.

• If the input is in unary, then the size of the input is equal to the value. When the input value is *x*, then number of squares needed to represent the input is *x*.

• If the input is in binary, then log *x* squares needed to represent the input value *x*. Thus, if the input size in binary is *N*, then the input size in unary is 2^N.

• It can be concluded that the size of the input is increasing exponentially.

• The reduction proof requires more than polynomial time to reduce 3*SAT* to *UNARY-SSUM*.

NP-completeness proof for *SUBSET-SUM* fails to show *UNARY-SSUM* is NP-complete because, the reduction is not polynomial time. The input size in unary is 2^N which takes exponential time.

Therefore, the NP-completeness proof for *SUBSET-SUM* fails to show *UNARY-SSUM* is NP-complete.

Comment

Step 2 of 2

To prove the language is in P, need to show that the polynomial time algorithm decides it. The standard algorithm for *SUBSET-SUM* runs in the time bounded by a polynomial in the input size *N* and the target integer *S*. The polynomial time algorithm for *UNARY-SSUM* is as follows:

• Compute the sum of numbers in the input.

• Check if the sum of the numbers is at least the target integer *S*.

• If the sum of the numbers is at least the target integer *S*, then run the standard dynamic programming algorithm of *SUBSET-SUM*.

• Otherwise, reject.

There exists a polynomial time algorithm to decide *UNARY-SSUM*. Therefore, $UNARY - SSUM \in \mathrm{P}$.

Comment