Reduction of
\textsc{Subset-Sum-Optimization} to
\textsc{Subset-Sum-Decision}

Chad Parry

\textbf{Subset Sum}

\begin{itemize}
  \item The Subset Sum problem involves searching through a collection of numbers to find a subset that sums to a certain number.
  \item The Subset Sum problem is known to be \textsc{NP}-complete.
\end{itemize}

\textbf{\textsc{Subset-Sum-Decision}}

\begin{itemize}
  \item Problem statement:
    \begin{itemize}
      \item Input:
        \begin{itemize}
          \item A collection of nonnegative integers \( A \)
          \item A nonnegative integer \( b \)
        \end{itemize}
      \item Output:
        \begin{itemize}
          \item Boolean value indicating whether some subset of the collection sums to \( b \)
        \end{itemize}
    \end{itemize}
\end{itemize}

\textbf{\textsc{Subset-Sum-Decision} Example}

\begin{itemize}
  \item Suppose you are given as inputs:
    \begin{itemize}
      \item The collection \( A = \{2, 3, 5, 7, 10\} \)
      \item The sum 14
    \end{itemize}
  \item The output is:
    \begin{itemize}
      \item TRUE
      \item \( 14 = 2 + 5 + 7 \)
    \end{itemize}
\end{itemize}

\textbf{\textsc{Subset-Sum-Optimization}}

\begin{itemize}
  \item Problem statement:
    \begin{itemize}
      \item Input:
        \begin{itemize}
          \item A collection of nonnegative integers \( A \)
          \item A nonnegative integer \( b \)
        \end{itemize}
      \item Output:
        \begin{itemize}
          \item A nonnegative integer \( b' \), which is the largest integer such that:
            \begin{itemize}
              \item \( b' \leq b \), and
              \item \textsc{Subset-Sum-Decision}(\( A, b' \)) is TRUE
            \end{itemize}
        \end{itemize}
    \end{itemize}
\end{itemize}

\textbf{\textsc{Subset-Sum-Optimization} Example}

\begin{itemize}
  \item Suppose you are given as inputs:
    \begin{itemize}
      \item The collection \( A = \{2, 3, 5, 7, 10\} \)
      \item The sum 16
    \end{itemize}
  \item The output is:
    \begin{itemize}
      \item 15
      \item 15 = 5 + 10
    \end{itemize}
\end{itemize}
Reduction Requirements

• The purpose of the reduction is to write an algorithm for \textsc{Subset-Sum-Optimization} which uses \textsc{Subset-Sum-Decision} as an oracle.
• A good reduction should run in polynomial time using the oracle.

Naïve Approach #1

• A brute force search though all combinations of the collection \( A \) will take exponential time.
• Any solution that involves guessing elements to remove from \( A \) will probably take exponential time.
• This approach doesn’t take advantage of the power of the oracle.

Naïve Approach #2

• Enumeration of the domain of \( b \) also takes exponential time.
  – The number \( b \) can be expressed with \( O(\log b) \) bits.
  – There are \((b + 1)\) integers to visit.
  – \((b + 1)\) is exponential with respect to \( \log b \).

Reduction Solution

• The algorithm is only a few lines long.

\[
\begin{align*}
\text{SUBSET-SUM-OPTIMIZATION}(A, b) \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto } 0 \text{ do} \\
A \leftarrow A + \{2^i\} \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto } 0 \text{ do} \\
A \leftarrow A - \{2^i\} \\
\text{if not } \text{SUBSET-SUM-DECISION}(A, b) \text{ then} \\
b \leftarrow b - 2^i \\
\text{return } b 
\end{align*}
\]

Adding Powers of Two

• The first step is to enumerate all the powers of two up to \( b \) and add them to \( A \).

\[
\begin{align*}
\text{SUBSET-SUM-OPTIMIZATION}(A, b) \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto } 0 \text{ do} \\
A \leftarrow A + \{2^i\} \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto } 0 \text{ do} \\
A \leftarrow A - \{2^i\} \\
\text{if not } \text{SUBSET-SUM-DECISION}(A, b) \text{ then} \\
b \leftarrow b - 2^i \\
\text{return } b 
\end{align*}
\]

Reduction Main Loop

• The next step is the main loop.
• Each power of two is removed from \( A \).

\[
\begin{align*}
\text{SUBSET-SUM-OPTIMIZATION}(A, b) \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto } 0 \text{ do} \\
A \leftarrow A + \{2^i\} \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto } 0 \text{ do} \\
A \leftarrow A - \{2^i\} \\
\text{if not } \text{SUBSET-SUM-DECISION}(A, b) \text{ then} \\
b \leftarrow b - 2^i \\
\text{return } b
\end{align*}
\]
Loop Invariants

- There are two loop invariants that allow the algorithm to work.
  1. \(b\) is always greater than or equal to the optimal solution.
  2. \(A\) contains a subset sum to \(b\).

No Sum Exists Condition

- When the oracle returns FALSE, the largest valid solution is \((b - 2^i)\).

\[
\text{SUBSET-SUM-OPTIMIZATION}(A, b) \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto 0 do} \\
\quad A \leftarrow A + \{2^i\} \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto 0 do} \\
\quad A \leftarrow A - \{2^i\} \\
\quad \text{if not } \text{SUBSET-SUM-DECISION}(A, b) \text{ then} \\
\qquad b \leftarrow b - 2^i \\
\text{return } b
\]

Return the Optimal Sum

- Finally the original collection \(A\) is restored.
- By this time \(b\) is optimal.

\[
\text{SUBSET-SUM-OPTIMIZATION}(A, b) \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto 0 do} \\
\quad A \leftarrow A + \{2^i\} \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ downto 0 do} \\
\quad A \leftarrow A - \{2^i\} \\
\quad \text{if not } \text{SUBSET-SUM-DECISION}(A, b) \text{ then} \\
\qquad b \leftarrow b - 2^i \\
\text{return } b
\]

Execution Example

Initial Values:
- \(A' = \{1, 5, 21\}\)
- \(b = 15\)

Adding powers of two:
- \(A' = \{1, 5, 21, 8, 4, 2, 1\}\)
- \(b = 15\)

Main loop initialization:
- \(A' = \{1, 5, 21, 8, 4, 2, 1\}\)
- \(b = 15\)
- \(i = 3, 2^i = 8\)
Execution Example
Remove the power of two:
- $A' = \{1, 5, 21, 4, 2, 1\}$
- $b = 15$
- $i = 3, 2^i = 8$

Execution Example
Try the oracle:
- $A' = \{1, 5, 21, 4, 2, 1\}$
- $b = 15$
- $i = 3, 2^i = 8$
- $\text{SUBSET-SUM-DECISION}(A', b) = \text{FALSE}$

Execution Example
No sum exists:
- $A' = \{1, 5, 21, 4, 2, 1\}$
- $b = 15 - 8 = 7$
- $i = 3, 2^i = 8$

Execution Example
Next loop iteration:
- $A' = \{1, 5, 21, 4, 2, 1\}$
- $b = 7$
- $i = 3, 2^i = 8$

Execution Example
Try the oracle:
- $A' = \{1, 5, 21, 4, 2, 1\}$
- $b = 7$
- $i = 2, 2^i = 4$
- $\text{SUBSET-SUM-DECISION}(A', b) = \text{TRUE}$
Execution Example

Next loop iteration:
• $A' = \{1, 5, 21, 2, 1\}$
• $b = 7$
• $i = 1, 2^i = 2$

Execution Example

Remove the power of two:
• $A' = \{1, 5, 21, \, \, \, 1\}$
• $b = 7$
• $i = 1, 2^i = 2$

Execution Example

Try the oracle:
• $A' = \{1, 5, 21, 1\}$
• $b = 7$
• $i = 1, 2^i = 2$
• $\text{SUBSET-SUM-DECISION}(A', b) = \text{TRUE}$

Execution Example

Next loop iteration:
• $A' = \{1, 5, 21, 1\}$
• $b = 7$
• $i = 0, 2^i = 1$

Execution Example

Remove the power of two:
• $A' = \{1, 5, 21, \, \, \, 1\}$
• $b = 7$
• $i = 0, 2^i = 1$

Execution Example

Try the oracle:
• $A' = \{1, 5, 21\}$
• $b = 7$
• $i = 0, 2^i = 1$
• $\text{SUBSET-SUM-DECISION}(A', b) = \text{FALSE}$
Execution Example

No sum exists:
- $A' = \{1, 5, 21\}$
- $b = 7 - 1 = 6$
- $i = 0, 2^i = 1$

Execution Example

Return the optimal sum:
- $A' = \{1, 5, 21\}$
- $b = 6$

Summary

- Each loop has $O(\log b)$ iterations, which is linear with respect to the size of $b$.
- The correct solution takes advantage of the NP-complete power of the oracle.