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

Chad Parry
Subset Sum

• The Subset Sum problem involves searching through a collection of numbers to find a subset that sums to a certain number.

• The Subset Sum problem is known to be NP-complete.
# Subset-Sum-Decision

### Problem statement:

- **Input:**
  - A collection of nonnegative integers $A$
  - A nonnegative integer $b$

- **Output:**
  - Boolean value indicating whether some subset of the collection sums to $b$
**SUBSET-SUM-DECISION Example**

- Suppose you are given as inputs:
  - The collection $A = \{2, 3, 5, 7, 10\}$
  - The sum 14
- The output is:
  - TRUE
  - $14 = 2 + 5 + 7$
SUBSET-SUM-OPTIMIZATION

• Problem statement:
  – Input:
    • A collection of nonnegative integers \( A \)
    • A nonnegative integer \( b \)
  – Output:
    • A nonnegative integer \( b' \), which is the largest integer such that:
      – \( b' \leq b \), and
      – Subset-Sum-Decision\((A, b')\) is TRUE
SUBSET-SUM-OPTIMIZATION

Example

• Suppose you are given as inputs:
  – The collection $A = \{2, 3, 5, 7, 10\}$
  – The sum 16

• The output is:
  – 15
  – $15 = 5 + 10$
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.

```
SUBSET-SUM-OPTIMIZATION(A, b)
    for i ← floor(log₂ b) downto 0 do
        A ← A + { 2^i }
    for i ← floor(log₂ b) downto 0 do
        A ← A - { 2^i }
    if not SUBSET-SUM-DECISION(A, b) then
        b ← b - 2^i
    return b
```
Adding Powers of Two

- The first step is to enumerate all the powers of two up to $b$ and add them to $A$.

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

- The next step is the main loop.
- Each power of two is removed from $A$.

\[
\text{SUBSET-SUM-OPTIMIZATION}(A, b)
\]

for $i \leftarrow \text{floor}(\log_2 b)$ downto 0 do
  
  $A \leftarrow A + \{ 2^i \}$

for $i \leftarrow \text{floor}(\log_2 b)$ downto 0 do
  
  $A \leftarrow A - \{ 2^i \}$

  if not \text{SUBSET-SUM-DECISION}(A, b) then
    
    $b \leftarrow b - 2^i$

return $b$
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 \((b - 2^i)\).

\[
\text{SUBSET-SUM-OPTIMIZATION}(A, b) \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ do} \\
A \leftarrow A + \{2^i\} \\
\text{for } i \leftarrow \text{floor}(\log_2 b) \text{ do} \\
A \leftarrow A - \{2^i\} \\
\text{if not SUBSET-SUM-DECISION}(A, b) \text{ then} \\
b \leftarrow b - 2^i \\
\text{return } b
\]
Return the Optimal Sum

- Finally the original collection $A$ is restored.
- By this time $b$ is optimal.

```
SUBSET-SUM-OPTIMIZATION(A, b)
  for $i$ ← floor(log$_2$b) downto 0 do
    $A$ ← $A$ + $\{2^i\}$
  for $i$ ← floor(log$_2$b) downto 0 do
    $A$ ← $A$ - $\{2^i\}$
    if not SUBSET-SUM-DECISION($A$, $b$) then
      $b$ ← $b$ - $2^i$
  return $b$
```
Execution Example

Initial Values:
• $A' = \{1, 5, 21\}$
• $b = 15$
Execution Example

Adding powers of two:

- \( A' = \{1, 5, 21, 8, 4, 2, 1\} \)
- \( b = 15 \)
Execution Example

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:
• $A' = \{1, 5, 21, \times 4, 2, 1\}$
• $b = 15$
• $i = 3, 2^i = 8$
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:
- $A' = \{1, 5, 21, 4, 2, 1\}$
- $b = 7$
- $i = 2$, $2^i = 4$
Execution Example

Remove the power of two:

- $A' = \{1, 5, 21, \times, 2, 1\}$
- $b = 7$
- $i = 2, \ 2^i = 4$
Execution Example

Try the oracle:

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

- $A' = \{1, 5, 21, \times 1\}$
- $b = 7$
- $i = 1, 2^i = 2$
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:

- \( A' = \{1, 5, 21, \text{X}\} \)
- \( b = 7 \)
- \( i = 0, \ 2^i = 1 \)
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.