We've seen:

\[
\begin{align*}
\text{SAT} & \quad \downarrow \\
\text{3SAT} & \\
\text{Subset-Sum} & \quad \text{Independent Set} \\
\text{Knapsack} & \quad \text{Clique} \\
& \quad \text{Vertex Cover}
\end{align*}
\]

**Subset-sum:**

*Input:* positive integers \( a_1, \ldots, a_n \) & \( t \)

*Output:* subset \( S \) of objects \( \{1, \ldots, n\} \)

where \( \sum_{i \in S} a_i = t \)

*No* if no such \( S \).

Using dynamic programming can solve in \( O(nt) \) time. But input size is \( O(n \log t) \).
Theorem: Subset-Sum is NP-complete

Proof:

a) Subset-Sum ∈ NP:

Given input \( \{a_1, \ldots, a_n, t, S\} \)

then in \( O(n \log t) \) time can check that

\[
\sum_{i \in S} a_i = t.
\]

b) 3SAT \( \implies \) Subset-Sum:

Take input \( f \) for 3SAT: variables \( x_1, \ldots, x_n \)

& clauses \( C_1, \ldots, C_m \)

Basic assumptions about \( f \) (o/w can simplify):

- No clause contains \( x_i \) & \( \overline{x_i} \)
  (if it does can drop the clause)

- Each \( x_i \) is in at least 1 clause
  (o/w can set \( x_i = F \))

& each \( \overline{x_i} \) is in at least 1 clause.
We'll create a subset-sum instance with:

numbers $v_1, v_2, \ldots, v_n, v_n, s_1, s_1, \ldots, s_m, s_m$

&

all $2n+2m+1$ numbers are base 10 & n+m digits.

$v_i$ corresponds to $X_i : v_i \in S \Leftrightarrow X_i = 1$

$v_i' \Leftrightarrow \overline{X_i} : v_i' \notin S \Leftrightarrow \overline{X_i} = 0$

So need that $v_i$ or $v_i'$ in $S$ but not both.

In $i$th digit of $v_i$, $v_i'$ & put a 1
& all other numbers put a 0 in $i$th digit.

Digit $n+j$ corresponds to clause $C_j$:

if $X_i \in C_j$ then put a 1 in row $\overline{v_i}$ in digit $n+j$

if $X_i \notin C_j$ then put a 1 in row $v_i$ in digit $n+j$. 
Want that 1, 2, or 3 literals in $G$ are included in $S$.

So put a 3 in digit $n+j$ of $T$.

& use $S_j, S_j^*$ as buffers:

Put a 1 in digit $n+j$ of $S_j$.

Put 0 in digit $n+j$ for all other numbers.

Then to get a sum of 3 in digit $n+j$ need to include 1, 2, or 3 of literals in $G$ & 0, 1, or 2 of $S_j, S_j^*$. 

Example:

\[ f = (x_1 v x_2 v x_3) \land (\overline{x_1} v x_2 v x_3) \land (x_1 v x_2 v x_3) \land (x_1 v x_2) \]

<table>
<thead>
<tr>
<th>( V_1 )</th>
<th>( V_2 )</th>
<th>( V_3 )</th>
<th>( V_4 )</th>
<th>( V_5 )</th>
<th>( V_6 )</th>
<th>( V_7 )</th>
<th>( V_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ V_i = 1000011 \]
\[ V_i = 1001100 \]

all blanks are zeros

+ = 11133333
3SAT input \( f \) is Satisfiable \( \iff \) Subset Sum \( \exists \{ y_1, y_2, \ldots, y_m \}, \{ s_1, s_2, \ldots, s_m \}, f \) has a solution

\[ \Rightarrow \text{Consider satisfying assignment } \sigma \text{ for } f \]

- if \( \sigma \) sets \( x_i = T \) then add \( y_i \) to \( S \)
- if \( x_i = F \) then add \( y_i \)' to \( S \).

For clause \( C_j \), \( \geq 1 \) literal satisfied by \( \sigma \)

\[ S \text{, 1-3 vertex numbers } v_1, v_2, \ldots, v_k, v_i \]

are in \( S \).

Add \( S_j \) &/ or \( S_j \)' to get a sum of \( 3 \) in digit \( n+j \).

Note, max sum in a digit is \( \leq 5 \)

So digits are independent (no carry).

\[ \Leftarrow \text{for 1st } n \text{ digits, if } v_i \in S \text{ then } x_i = T \]

- if \( v_i \in S \) then \( x_i = F \)

for digit \( n+j \) to get a sum of \( 3 \)

need at least one of \( v_1, v_2, \ldots, v_k \) in \( S \)

So \( C_j \) is satisfied.
**Knapsack**

**Input:** \( w_1, \ldots, w_n, v_1, \ldots, v_n \) & \( B \) & \( V \)

**Output:** Subset \( S \) where \( \sum_{i \in S} w_i \leq B \) \& \( \sum_{i \in S} v_i \geq V \)

or **NO**.

**Theorem:** Knapsack is NP-complete.

**Proof:**

a) Knapsack \( \in \) NP:

In \( O(n \log B) \) time

Can check that \( \sum_{i \in S} w_i \leq B \)

\& in \( O(n \log V) \) time check that \( \sum_{i \in S} v_i \geq V \).

b) Subset-Sum \( \rightarrow \) Knapsack.

Take input \( \{a_1, \ldots, a_n, +, -\} \) for Subset-Sum.

Set \( v_i = w_i = a_i \) \& \( B = V = + \).

Then \( \sum_{i \in S} w_i \leq B \) \& \( \sum_{i \in S} v_i \geq V \) \( \iff \sum_{a_i = +} \)

\( \iff \sum_{a_i = -} \).