Subset Sum:

Input: Positive integers $a_1, \ldots, a_n$ and $t$

Output: Subset $S$ of objects $\{1, \ldots, n\}$ where $\sum_{i \in S} a_i = t$

if such a $S$ exists

No otherwise

Using Dynamic Programming can solve in $O(n^t)$ time.

But Subset-Sum is NP-complete.

We'll prove this today.

Why is $O(n^t)$ not poly-time?

Should be poly in $n$ & $\log t$. 
Proof that Subset Sum is NP-complete:

a) Subset-Sum ∈ NP:

Given input $a_1, ..., a_n, t, s$ in $O(n)$ time can check that

$$\sum_{i \in S} a_i = t$$

b) 3SAT → Subset Sum

Take input $f$ for 3SAT.

Variables $x_1, ..., x_n$

Clauses $C_1, ..., C_n$

Basic assumptions about $f$:

- no clause contains $x_i$ & $\overline{x_i}$
  otherwise it's clearly satisfied & we can drop the clause

- each $x_i$ is in at least 1 clause
  otherwise set $x_i = F$ & simplify

  Similarly, $\overline{x_i}$ is in at least 1 clause.
Input for subset-sum will be numbers:

\[ V_1, V_2, \ldots, V_i, V_i', S_1, S_i, \ldots, S_m, S_m' \]

\( n \) \& \( m \) will be a \( n \times m \) digit number

all numbers are base 10

\( V_i \) corresponds to \( x_i \): \( V_i \in S \) then \( x_i = 1 \)

\( V_i' \) \( \overline{x_i} \): \( V_i' \in S \) then \( x_i = 0 \)

Need that exactly 1 of \( V_i, V_i' \) in \( S \)

\( \Rightarrow \) In the \( i \)th digit of \( V_i, V_i' \& \) put a 1

all other numbers have a 0 in \( i \)th digit

Digit \( n+j \) corresponds to clause \( C_j \)

If \( x_i \) appears in \( C_j \) put a 1 in row \( V_i \) in digit \( n+j \)

If \( \overline{x_i} \) appears in \( C_j \) put a 1 in \( V_i \) in digit \( n+j \)
Wait that 1, 2, or 3 of the literals in G are included in S.

Put a 3 in digit n+j of +

Use S_j, S_j' as buffers:

So put a 1 in digit n+j of S_j & S_j'.

Put a 0 in digit n+j of all other numbers.

Then to get a sum of 3 in digit n+j

need to include 1, 2, or 3 of

literals in C_j & 0, 1, or 2 of S_j, S_j'.
**Example:**

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_1' )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_2' )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_3' )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_4 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( v_4' )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( f )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( v_1 = 1000011 \)
\( v_1' = 1001100 \)

\( t = 1113333 \)
Subset Sum has a solution iff 3SAT \( f \) is satisfiable.

\[ \iff \]
For 1st \( n \) digits, include \( v_i \) or \( v_i' \) to get a 1 in digit \( i \),
if \( v_i \) included \( \implies x_i = T \)
if \( v_i' \) included \( \implies x_i = F \)

For digit \( n+j \),
to get a sum of 3 need to include at least one of the literals
So \( C_j \) is satisfied

\[ \iff \]
If \( x_i = T \), add \( v_i \) to \( S \)
\( x_i = F \), add \( v_i' \) to \( S \)
So \( i \)th digit of \( f \) is satisfied

For clause \( C_j \) at least one literal is satisfied
add \( S_j \) or \( S_j' \) if needed
to get a sum of 3 in digit \( n+j \).
Knapsack:
input: \( n \) objects with weights \( w_1, \ldots, w_n \) & values \( v_1, \ldots, v_n \)
Capacity \( B \)
Value \( V \)
output: subset of objects with
\[
\sum_{i \in S} w_i \leq B
\]
& \[
\sum_{i \in S} v_i \geq V
\]
NO if no such \( S \) exists

Knapsack is NP-complete:

a) Knapsack \( \in \text{NP} \):
Given Knapsack input \& \( S \), in \( O(n) \) time
Can check that \( S \) is a solution.
b) Subset Sum $\rightarrow$ Knapsack

Take input $a_1, \ldots, a_n$ & $t$ for Subset Sum

Set $V_i = w_i = a_i$

Set $B = V = t$

Then Knapsack is trying to find a subset $S$ where

\[ \sum_{i \in S} w_i \leq B \iff \sum_{i \in S} a_i \leq t \]

\[ \sum_{i \in S} V_i \geq V \iff \sum_{i \in S} a_i \geq t \]

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

So same as Subset Sum.