- **Fractional Knapsack**

<table>
<thead>
<tr>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1 = 2$</td>
<td>$V_2 = 3$</td>
<td>$V_3 = 4$</td>
</tr>
<tr>
<td>$\frac{V_1}{W_1} = \frac{1}{2}$</td>
<td>$\frac{V_2}{W_2} = \frac{3}{5}$</td>
<td>$\frac{V_3}{W_3} = \frac{4}{7}$</td>
</tr>
</tbody>
</table>

- Previously (0/1 knapsack problem) solution: put item 1 + item 2
  value = 5

- Fractional Knapsack
  
  Optimal solution: $item 2 + \left( \frac{5}{7} \right) item 3$
  
  Value = $3 + \frac{5}{7} \cdot 4 = 5 + \frac{6}{7} > 5$

- Algorithm
  
  Sort items in decreasing order of $\frac{V_i}{W_i}$ (value per weight)
  
  for $i = 1 \to n$

  if $W_i \leq W$ then
  
  weight remaining capacity
  
  $W \leftarrow W - W_i$
  
  $V \leftarrow V + V_i$

  else

  $V \leftarrow V + V_i \cdot \left( \frac{W}{W_i} \right)$

  max fraction of item $i$ that can fit in remaining capacity

  $W \leftarrow 0$

  break

  return $V$

  running time $O(n \log n)$ (for sorting)

- Proof: Assume towards contradiction that there is an instance of fractional knapsack s.t. algorithm is not optimal.
without loss of generality assume

1. items are sorted in decreasing order by $\frac{V_i}{W_i}$
2. no two items that have the same $\frac{V_i}{W_i}$

Justification: suppose there are two items i, j

\[ \text{st. } \frac{V_i}{W_i} = \frac{V_j}{W_j} \]

then the problem is equivalent if we merge these two and include an item with value $V_i + V_j$ weight $W_i + W_j$

\[
\begin{align*}
V_i &= 1 \quad W_i = 2 \\
V_j &= 2 \quad W_j = 4
\end{align*} \implies V_i = 3 \quad W_i = 6
\]

\[ \frac{V_i}{W_i} > \frac{V_j}{W_j} \]

let alg's solution be $P_1, P_2, P_3, \ldots, P_n$ (fractions of items in the knapsack)

opt's solution be $q_1, q_2, q_3, \ldots, q_n$

by assumption $\sum_{i=1}^{n} P_i V_i < \sum_{i=1}^{n} q_i V_i$

alg's value < opt's value

let i be the first coordinate where $P_i \neq q_i$

by design of the algorithm $P_i > q_i$

let j be a coordinate where $P_j < q_j$

now consider a solution

\[
\begin{align*}
q'_i &= q_i + 3 \\
q'_j &= q_j - \frac{3 W_i}{W_j} (\text{weight} - 3 W_i) \\
q'_k &= q_k \quad \text{for any other coordinates } q'_k = q_k
\end{align*}
\]

for new solution $q'$ $\sum_{i=1}^{n} q'_i W_i = \sum_{i=1}^{n} q_i W_i$ \text{ (total weight does not change)}

\[
\sum_{i=1}^{n} q'_i V_i = \left( \sum_{i=1}^{n} q_i V_i \right) + 3 V_i - \frac{3 W_i}{W_j} V_j \geq \sum_{i=1}^{n} q_i V_i
\]

$q'$ is a valid solution with value more than opt, this is a contradiction
\[ w_1 \quad w_2 \]

\[ q' \text{ is a valid solution with value more than }OPT, \text{this is a contradiction} \]

- **Horn-SAT**

  - Simple example
    \[
    \begin{align*}
    &x \land y \Rightarrow z \\
    &x \\
    &x \lor y \lor z
    \end{align*}
    \]
    One satisfying assignment
    \[
    x = \text{true} \quad y = \text{false} \quad z = \text{false}
    \]
    On the other hand
    \[
    x = y = \text{true} \quad z = \text{false} \quad \text{violates clause #1}
    \]

  - Example #2
    \[
    \begin{align*}
    &x \land y \Rightarrow z \\
    &\checkmark
    \end{align*}
    \]
    Claim: no satisfying assignment.

    2:
    \[
    \checkmark
    \]
    init: \[ x = y = z = \text{false} \]

    4:
    \[
    \checkmark
    \]
    \[
    \begin{align*}
    &x = y = \text{true} \quad \text{for clauses } 2, 3 \\
    &z = \text{true} \quad \text{for clause } 1
    \end{align*}
    \]

- Algorithm (sketch)
  - Check clause #4, it is not satisfied, output No.

  - Initialize all variables to false.

  - Set all variables in 2nd type of clauses to true.

  - Repeat if for some clause of type 1, all assumptions are true.
    - Set the conclusion to true.
    - Until we don't need to set more variables.

  - Check all clauses of type 3, output Yes if they are all satisfied.

  (only the part where alg outputs no)

- Proof: assume towards contradiction that there is a satisfying assignment

  Let \[ x_1, x_2, \ldots, x_k \] be the variables set to true by the alg.

  In the order that they are set to true.

  If \[ x_1, \ldots, x_k \] are true in the satisfying assignment,

  then one of clauses of type 3 will be violated.

  Else let \[ x_1 \] be the first variable that is false in set.

  Assign \[ x_1 \] is in clause of type 2, the clause is violated.

  Else: \[ x_1 \] must belong to the conclusion of a clause of type 1.

  By design of alg, all the assumptions are true.

  This particular clause is violated.
by design of alg, all the assumptions are true
this particular clause is violated.