**Even More Greed**

**Greedy Algorithm for Fractional Knapsack Problem**

The Fractional Knapsack Problem is defined as follows: Given a container that can hold \( k \) kilograms of mass, and a set of \( n \) objects each with known value and mass, find the most valuable combination of objects that will fit in the container, allowing fractions of objects to be used, where the value of a fraction of an object is the same fraction of the value of the object.

More formally: Given \( k \) and a set of \( n \) pairs of the form \((v_i, m_i)\) find a set of values \(\{p_1, p_2, \ldots, p_n\}\) such that \(0 \leq p_i \leq 1\) \(\forall i\) and \(\sum_{i=1}^{n}(p_i \cdot m_i) \leq k\) and \(\sum_{i=1}^{n}(p_i \cdot v_i)\) is maximized.

In this formulation, the \(p_i\) values are the fractions.

Our goal is to find a greedy algorithm to solve this problem. As with all greedy algorithms, our first task is to sort the objects. We can experiment with a variety of sorting criteria but the one that leads to success in this case is to sort the objects in decreasing \(\frac{v_i}{m_i}\) order.

**Greedy FKS:**

Sort the objects in decreasing \(\frac{v_i}{m_i}\) order

While \(k > 0\) and there are still objects to consider:
   - Take as much of the next item as possible
   - Reduce \(k\) by the mass amount just added to the knapsack

That’s about as simple as an algorithm can get, and it clearly runs in \(O(n \log n)\) time (the sort is the longest part of the algorithm.)
Proof of Correctness:

Rather than use proof by induction, we will prove this using a technique we can call “eliminate the differences” - it is often useful when proving that Greedy Algorithms find optimal solutions.

WLOG we will assume that object $o_1$ has the highest $\frac{v_i}{m_i}$ ratio, object $o_2$ has the second highest, etc.

Let $A$ be the algorithm’s solution, and let $O$ be an optimal solution. If $A$ and $O$ are identical, then $A$ is optimal.

Suppose $A$ and $O$ are not identical. In fact, suppose $A$ contains more of $o_1$ than $O$ does. That means $O$ contains some other objects in at least the same amount as the amount of $o_1$ that $O$ leaves out. But since $\frac{v_1}{m_1}$ is ≥ all the other $\frac{v_i}{m_i}$ ratios, we can replace some of the “other stuff” in $O$ with the left-out $o_1$, and the total value cannot decrease. Call this new optimal solution $O’$. This means that $A$ and $O’$ have one fewer difference than $A$ and $O$ did.

We can continue eliminating differences in this way until we reach a point where $A = O$... (we don’t know how many differences there were to start with, so I used “...” for the superscript after they have all been eliminated) Since every time we eliminate a difference we get a new optimal solution, we eventually show that $A$ is equal to an optimal solution ... so $A$ is optimal.

Example of the algorithm in action:

$$k = 15$$

<table>
<thead>
<tr>
<th>$v_i$</th>
<th>15</th>
<th>14</th>
<th>30</th>
<th>100</th>
<th>2</th>
<th>100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_i$</td>
<td>6</td>
<td>7</td>
<td>20</td>
<td>80</td>
<td>2</td>
<td>100001</td>
</tr>
<tr>
<td>$\frac{v_i}{m_i}$</td>
<td>2.5</td>
<td>2</td>
<td>1.5</td>
<td>1.25</td>
<td>1</td>
<td>$\frac{100000}{100001}$</td>
</tr>
</tbody>
</table>

The algorithm takes all of $o_1$, all of $o_2$, and 2 of the available 20 units of $o_3$ with a total value of $15 + 14 + \frac{2}{20} \cdot 30 = 32$

By the proof given above, there is no solution with a value > 32