Outline

- Introduction
- The Knapsack problem.
- A greedy algorithm for the fractional knapsack problem
- Correctness
A greedy algorithm for an optimization problem always makes the choice that looks best at the moment and adds it to the current subsolution.
A greedy algorithm for an optimization problem always makes the choice that looks best at the moment and adds it to the current subsolution.

Final output is an optimal solution.
A greedy algorithm for an optimization problem always makes the choice that looks best at the moment and adds it to the current subsolution.

Final output is an optimal solution.

Greedy algorithms don’t always yield optimal solutions.
A greedy algorithm for an optimization problem always makes the choice that looks best at the moment and adds it to the current subsolution.

Final output is an optimal solution.

Greedy algorithms don’t always yield optimal solutions but, when they do, they’re usually the simplest and most efficient algorithms available.
Outline

- Introduction
- The Knapsack problem.
- A greedy algorithm for the fractional knapsack problem
- Correctness
The Knapsack Problem...

Capacity of knapsack: $K = 4$
The Knapsack Problem...

Capacity of knapsack: \( K = 4 \)

**Fractional Knapsack Problem:**

\[
\begin{align*}
A & \quad 2 \text{ pd} \quad \$100 \\
B & \quad 2 \text{ pd} \quad \$10 \\
C & \quad 3 \text{ pd} \quad \$120
\end{align*}
\]
The Knapsack Problem...

Capacity of knapsack: $K = 4$

**Fractional** Knapsack Problem:
Can take a fraction of an item.
The Knapsack Problem...

Capacity of knapsack: $K = 4$

**Fractional Knapsack Problem:** Can take a fraction of an item.

Solution:

<table>
<thead>
<tr>
<th>2 pd</th>
<th>2 pd</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>
| $100$| $100$| $80$
The Knapsack Problem...

Capacity of knapsack: $K = 4$

**Fractional Knapsack Problem:**
Can take a fraction of an item.

**0-1 Knapsack Problem:**
Can only take or leave item. You can’t take a fraction.

Solution:

<table>
<thead>
<tr>
<th>2 pd A $100</th>
<th>2 pd C $80</th>
</tr>
</thead>
</table>

Version of September 17, 2016

Greedy Algorithms: The Fractional Knapsack
The Knapsack Problem...

Capacity of knapsack: \( K = 4 \)

**Fractional Knapsack Problem:**
Can take a fraction of an item.

**0-1 Knapsack Problem:**
Can only take or leave item. You can’t take a fraction.
The Fractional Knapsack Problem: Formal Definition

Given $K$ and a set of $n$ items:

<table>
<thead>
<tr>
<th>weight</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>...</th>
<th>$w_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>$v_1$</td>
<td>$v_2$</td>
<td>...</td>
<td>$v_n$</td>
</tr>
</tbody>
</table>

Find: $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, n$ such that

$$\sum_{i=1}^{n} x_i w_i \leq K$$

and the following is maximized:

$$\sum_{i=1}^{n} x_i v_i$$
Outline

- Introduction
- The Knapsack problem.
- A greedy algorithm for the fractional knapsack problem
- Correctness
Sort items by decreasing value-per-pound
Greedy Solution for Fractional Knapsack

Sort items by decreasing value-per-pound

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$200</td>
<td>1 pd</td>
</tr>
<tr>
<td>B</td>
<td>$240</td>
<td>3 pd</td>
</tr>
<tr>
<td>C</td>
<td>$140</td>
<td>2 pd</td>
</tr>
<tr>
<td>D</td>
<td>$150</td>
<td>5 pd</td>
</tr>
</tbody>
</table>

If knapsack holds K = 5 pd, solution is: 1 pd A, 3 pd B, 1 pd C.
Greedy Solution for Fractional Knapsack

Sort items by **decreasing** value-per-pound

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1 pd</td>
<td>3 pd</td>
<td>2 pd</td>
<td>5 pd</td>
</tr>
<tr>
<td>$200</td>
<td>$240</td>
<td>$140</td>
<td>$150</td>
</tr>
<tr>
<td>200</td>
<td>80</td>
<td>70</td>
<td>30</td>
</tr>
</tbody>
</table>

If knapsack holds $K = 5$ pd, solution is:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pd</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>3 pd</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1 pd</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>
Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$. 
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound \( \rho_i = \frac{v_i}{w_i} \) for \( i = 1, 2, \ldots, n \).
- Sort the items by decreasing \( \rho_i \).
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be $1, 2, \ldots, i, \ldots n$,
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound \( \rho_i = \frac{v_i}{w_i} \) for \( i = 1, 2, \ldots, n \).
- Sort the items by decreasing \( \rho_i \).
  Let the sorted item sequence be \( 1, 2, \ldots, i, \ldots n \), and the corresponding value-per-pound and weight be \( \rho_i \) and \( w_i \) respectively.

Let \( k \) be the current weight limit (Initially, \( k = K \)).
In each iteration, we choose item \( i \) from the head of the unselected list.
- If \( k \geq w_i \), set \( x_i = 1 \) (we take item \( i \)), and reduce \( k = k - w_i \), then consider the next unselected item.
- If \( k < w_i \), set \( x_i = \frac{k}{w_i} \) (we take a fraction \( \frac{k}{w_i} \) of item \( i \)), then the algorithm terminates.

Running time: \( O(n \log n) \).
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound \( \rho_i = \frac{v_i}{w_i} \) for \( i = 1, 2, \ldots, n \).
- Sort the items by decreasing \( \rho_i \).
  Let the sorted item sequence be 1, 2, \ldots, i, \ldots n, and the corresponding value-per-pound and weight be \( \rho_i \) and \( w_i \) respectively.
- Let \( k \) be the current weight limit (Initially, \( k = K \)).
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be $1, 2, \ldots, i, \ldots n$, and the corresponding value-per-pound and weight be $\rho_i$ and $w_i$ respectively.
- Let $k$ be the current weight limit (Initially, $k = K$). In each iteration, we choose item $i$ from the head of the unselected list.

Running time: $O(n \log n)$. 
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be $1, 2, \ldots, i, \ldots n$, and the corresponding value-per-pound and weight be $\rho_i$ and $w_i$ respectively.
- Let $k$ be the current weight limit (Initially, $k = K$).
  In each iteration, we choose item $i$ from the head of the unselected list.
    - If $k \geq w_i$, 

Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be $1, 2, \ldots, i, \ldots, n$, and the corresponding value-per-pound and weight be $\rho_i$ and $w_i$ respectively.
- Let $k$ be the current weight limit (Initially, $k = K$). In each iteration, we choose item $i$ from the head of the unselected list.
  - If $k \geq w_i$, set $x_i = 1$ (we take item $i$), and reduce $k = k - w_i$, then consider the next unselected item.
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be 1, 2, \ldots, i, \ldots n, and the corresponding value-per-pound and weight be $\rho_i$ and $w_i$ respectively.
- Let $k$ be the current weight limit (Initially, $k = K$).
  In each iteration, we choose item $i$ from the head of the unselected list.
    - If $k \geq w_i$, set $x_i = 1$ (we take item $i$), and reduce $k = k - w_i$, then consider the next unselected item.
    - If $k < w_i$,  

Running time: $O(n \log n)$. 
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be $1, 2, \ldots, i, \ldots n$, and the corresponding value-per-pound and weight be $\rho_i$ and $w_i$ respectively.
- Let $k$ be the current weight limit (Initially, $k = K$). In each iteration, we choose item $i$ from the head of the unselected list.
  - If $k \geq w_i$, set $x_i = 1$ (we take item $i$), and reduce $k = k - w_i$, then consider the next unselected item.
  - If $k < w_i$, set $x_i = k/w_i$ (we take a fraction $k/w_i$ of item $i$),
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound \( \rho_i = \frac{v_i}{w_i} \) for \( i = 1, 2, \ldots, n \).
- Sort the items by decreasing \( \rho_i \).
  Let the sorted item sequence be \( 1, 2, \ldots, i, \ldots n \), and the corresponding value-per-pound and weight be \( \rho_i \) and \( w_i \) respectively.
- Let \( k \) be the current weight limit (Initially, \( k = K \)). In each iteration, we choose item \( i \) from the head of the unselected list.
  - If \( k \geq w_i \), set \( x_i = 1 \) (we take item \( i \)), and reduce \( k = k - w_i \), then consider the next unselected item.
  - If \( k < w_i \), set \( x_i = k / w_i \) (we take a fraction \( k/w_i \) of item \( i \)). Then the algorithm terminates.

Running time: \( O(n \log n) \).
Greedy Solution for Fractional Knapsack

- Calculate the value-per-pound $\rho_i = \frac{v_i}{w_i}$ for $i = 1, 2, \ldots, n$.
- Sort the items by decreasing $\rho_i$.
  Let the sorted item sequence be $1, 2, \ldots, i, \ldots n$, and the corresponding value-per-pound and weight be $\rho_i$ and $w_i$ respectively.
- Let $k$ be the current weight limit (Initially, $k = K$). In each iteration, we choose item $i$ from the head of the unselected list.
  - If $k \geq w_i$, set $x_i = 1$ (we take item $i$), and reduce $k = k - w_i$, then consider the next unselected item.
  - If $k < w_i$, set $x_i = k / w_i$ (we take a fraction $k / w_i$ of item $i$), Then the algorithm terminates.

Running time: $O(n \log n)$. 
Observe that the algorithm may take a fraction of an item.
Observe that the algorithm may take a fraction of an item. This can \textit{only} be the \textit{last} selected item.
Observe that the algorithm may take a fraction of an item. This can only be the last selected item.

We claim that the total value for this set of items is the optimal value.
Outline

- Introduction
- The Knapsack problem.
- A greedy algorithm for the fractional knapsack problem
- Correctness
Correctness

Given a set of \( n \) items \( \{1, 2, \ldots, n\} \).
Given a set of $n$ items $\{1, 2, \ldots, n\}$.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$. 
Correctness

Given a set of $n$ items \{1, 2, ..., $n$\}.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq ... \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, ..., x_k \rangle$
Correctness

Given a set of \( n \) items \( \{1, 2, \ldots, n\} \).

- Assume items sorted by per-pound values: \( \rho_1 \geq \rho_2 \geq \ldots \geq \rho_n \).

Let the greedy solution be \( G = \langle x_1, x_2, \ldots, x_k \rangle \)

- \( x_i \) indicates fraction of item \( i \) taken
Correctness

Given a set of \( n \) items \( \{1, 2, \ldots, n\} \).

- Assume items sorted by per-pound values: \( \rho_1 \geq \rho_2 \geq \ldots \geq \rho_n \).

Let the greedy solution be \( G = \langle x_1, x_2, \ldots, x_k \rangle \)

- \( x_i \) indicates fraction of item \( i \) taken (all \( x_i = 1 \), except possibly for \( i = k \)).
Given a set of $n$ items $\{1, 2, ..., n\}$.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq ... \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, ..., x_k \rangle$

- $x_i$ indicates fraction of item $i$ taken (all $x_i = 1$, except possibly for $i = k$).

Consider any optimal solution $O = \langle y_1, y_2, ..., y_n \rangle$
Correctness

Given a set of $n$ items \{1, 2, ..., $n$\}.
- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, \ldots, x_k \rangle$
- $x_i$ indicates fraction of item $i$ taken (all $x_i = 1$, except possibly for $i = k$).

Consider any optimal solution $O = \langle y_1, y_2, \ldots, y_n \rangle$
- $y_i$ indicates fraction of item $i$ taken in $O$
Given a set of $n$ items \{1, 2, ..., $n$\}.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq ... \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, ..., x_k \rangle$

- $x_i$ indicates fraction of item $i$ taken (all $x_i = 1$, except possibly for $i = k$).

Consider any optimal solution $O = \langle y_1, y_2, ..., y_n \rangle$

- $y_i$ indicates fraction of item $i$ taken in $O$ (for all $i$, $0 \leq y_i \leq 1$).
Given a set of $n$ items $\{1, 2, \ldots, n\}$.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, \ldots, x_k \rangle$

- $x_i$ indicates fraction of item $i$ taken (all $x_i = 1$, except possibly for $i = k$).

Consider any optimal solution $O = \langle y_1, y_2, \ldots, y_n \rangle$

- $y_i$ indicates fraction of item $i$ taken in $O$ (for all $i$, $0 \leq y_i \leq 1$).

Knapsack must be full in both $G$ and $O$:

$$\sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} y_i w_i = K.$$
Correctness

Given a set of \( n \) items \( \{1, 2, \ldots, n\} \).

- Assume items sorted by per-pound values: \( \rho_1 \geq \rho_2 \geq \ldots \geq \rho_n \).

Let the greedy solution be \( G = \langle x_1, x_2, \ldots, x_k \rangle \)

- \( x_i \) indicates fraction of item \( i \) taken (all \( x_i = 1 \), except possibly for \( i = k \)).

Consider any optimal solution \( O = \langle y_1, y_2, \ldots, y_n \rangle \)

- \( y_i \) indicates fraction of item \( i \) taken in \( O \) (for all \( i \), \( 0 \leq y_i \leq 1 \)).

Knapsack must be full in both \( G \) and \( O \):

\[
\sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} y_i w_i = K.
\]

Consider the first item \( i \) where the two selections differ.
Correctness

Given a set of \( n \) items \( \{1, 2, \ldots, n\} \).
- Assume items sorted by per-pound values: \( \rho_1 \geq \rho_2 \geq \ldots \geq \rho_n \).

Let the greedy solution be \( G = \langle x_1, x_2, \ldots, x_k \rangle \)
- \( x_i \) indicates fraction of item \( i \) taken (all \( x_i = 1 \), except possibly for \( i = k \)).

Consider any optimal solution \( O = \langle y_1, y_2, \ldots, y_n \rangle \)
- \( y_i \) indicates fraction of item \( i \) taken in \( O \) (for all \( i \), \( 0 \leq y_i \leq 1 \)).
- Knapsack must be full in both \( G \) and \( O \):
  \[
  \sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} y_i w_i = K.
  \]

Consider the first item \( i \) where the two selections differ.
- By definition, solution \( G \) takes a greater amount of item \( i \) than solution \( O \).
Given a set of $n$ items $\{1, 2, \ldots, n\}$.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, \ldots, x_k \rangle$

- $x_i$ indicates fraction of item $i$ taken (all $x_i = 1$, except possibly for $i = k$).

Consider any optimal solution $O = \langle y_1, y_2, \ldots, y_n \rangle$

- $y_i$ indicates fraction of item $i$ taken in $O$ (for all $i$, $0 \leq y_i \leq 1$).

Knapsack must be full in both $G$ and $O$:

$$\sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} y_i w_i = K.$$ 

Consider the first item $i$ where the two selections differ.

- By definition, solution $G$ takes a greater amount of item $i$ than solution $O$ (because the greedy solution always takes as much as it can).
Given a set of $n$ items $\{1, 2, ..., n\}$.

- Assume items sorted by per-pound values: $\rho_1 \geq \rho_2 \geq ... \geq \rho_n$.

Let the greedy solution be $G = \langle x_1, x_2, ..., x_k \rangle$

- $x_i$ indicates fraction of item $i$ taken (all $x_i = 1$, except possibly for $i = k$).

Consider any optimal solution $O = \langle y_1, y_2, ..., y_n \rangle$

- $y_i$ indicates fraction of item $i$ taken in $O$ (for all $i$, $0 \leq y_i \leq 1$).

Knapsack must be full in both $G$ and $O$:

$$\sum_{i=1}^{n} x_i w_i = \sum_{i=1}^{n} y_i w_i = K.$$ 

Consider the first item $i$ where the two selections differ.

- By definition, solution $G$ takes a greater amount of item $i$ than solution $O$ (because the greedy solution always takes as much as it can). Let $x = x_i - y_i$. 
Consider the following new solution $O'$ constructed from $O$:

For $j < i$, keep $y'_{j} = y_{j}$.

Set $y'_{i} = x_{i}$.

In $O$, remove items of total weight $x_{w_i}$ from items $i + 1$ to $n$, resetting the $y'_{j}$ appropriately.

This is always doable because $\sum_{j=i}^{n} x_{j} = \sum_{j=i}^{n} y_{j}$.

The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?) since $O$ is largest possible solution and value of $O'$ cannot be smaller than that of $O$, $O$ and $O'$ must be equal.

Thus solution $O'$ is also optimal.

By repeating this process, we will eventually convert $O$ into $G$, without changing the total value of the selection. Therefore $G$ is also optimal!
Correctness...

Consider the following new solution $O'$ constructed from $O$:

- For $j < i$, keep $y'_j = y_j$.
- Set $y'_i = x_i$.
- In $O$, remove items of total weight $xw_i$ from items $i + 1$ to $n$, resetting the $y'_j$ appropriately.

This is always doable because $\sum_{j=i}^{n} x_j = \sum_{j=i}^{n} y_j$.

The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?)

Since $O$ is largest possible solution and value of $O'$ cannot be smaller than that of $O$, $O$ and $O'$ must be equal.

Thus solution $O'$ is also optimal.

By repeating this process, we will eventually convert $O$ into $G$, without changing the total value of the selection.

Therefore $G$ is also optimal!
Correctness...

Consider the following new solution $O'$ constructed from $O$:

- For $j < i$, keep $y'_j = y_j$.
- Set $y'_i = x_i$.
- In $O$, remove items of total weight $xw_i$ from items $i + 1$ to $n$, resetting the $y'_j$ appropriately.

This is always doable because $\sum_{j=i}^{n} x_j = \sum_{j=i}^{n} y_j$

- The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?)
Correctness…

Consider the following new solution $O'$ constructed from $O$:

- For $j < i$, keep $y'_j = y_j$.
- Set $y'_i = x_i$.
- In $O$, remove items of total weight $xw_i$ from items $i + 1$ to $n$, resetting the $y'_j$ appropriately.

This is always doable because $\sum_{j=i}^{n} x_j = \sum_{j=i}^{n} y_j$

- The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?)

- Since $O$ is largest possible solution and value of $O'$ cannot be smaller than that of $O$, $O$ and $O'$ must be equal.
Consider the following new solution $O' \text{ constructed from } O$:

- For $j < i$, keep $y'_j = y_j$.
- Set $y'_i = x_i$.
- In $O$, remove items of total weight $xw_i$ from items $i + 1$ to $n$, resetting the $y'_j$ appropriately.

This is always doable because $\sum_{j=i}^{n} x_j = \sum_{j=i}^{n} y_j$

- The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?)

- Since $O$ is largest possible solution and value of $O'$ cannot be smaller than that of $O$, $O$ and $O'$ must be equal.

- Thus solution $O'$ is also optimal.
Consider the following new solution $O'$ constructed from $O$:

- For $j < i$, keep $y'_j = y_j$.
- Set $y'_i = x_i$.
- In $O$, remove items of total weight $xw_i$ from items $i + 1$ to $n$, resetting the $y'_j$ appropriately.

This is always doable because $\sum_{j=i}^n x_j = \sum_{j=i}^n y_j$

- The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?)

- Since $O$ is largest possible solution and value of $O'$ cannot be smaller than that of $O$, $O$ and $O'$ must be equal.

- Thus solution $O'$ is also optimal.

By repeating this process, we will eventually convert $O$ into $G$, without changing the total value of the selection.
Correctness...

Consider the following new solution $O'$ constructed from $O$:

- For $j < i$, keep $y'_j = y_j$.
- Set $y'_i = x_i$.
- In $O$, remove items of total weight $xw_i$ from items $i + 1$ to $n$, resetting the $y'_j$ appropriately.

This is always doable because $\sum_{j=i}^{n} x_j = \sum_{j=i}^{n} y_j$

- The total value of solution $O'$ is greater than or equal to the total value of solution $O$ (why?)

- Since $O$ is largest possible solution and value of $O'$ cannot be smaller than that of $O$, $O$ and $O'$ must be equal.

- Thus solution $O'$ is also optimal.

By repeating this process, we will eventually convert $O$ into $G$, without changing the total value of the selection.

Therefore $G$ is also optimal!
The 0-1 Knapsack Problem does not have a greedy solution!

Example

\[ K = 4. \text{ Solution is } \]

\[ \text{item B + item C} \]
The 0-1 Knapsack Problem does not have a greedy solution!

Example

\[ K = 4. \text{ Solution is item B } + \text{ item C} \]
The 0-1 Knapsack Problem does not have a greedy solution!

**Example**

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$300</td>
<td>3 pd</td>
</tr>
<tr>
<td>B</td>
<td>$190</td>
<td>2 pd</td>
</tr>
<tr>
<td>C</td>
<td>$180</td>
<td>2 pd</td>
</tr>
</tbody>
</table>

Value per pound:
- A: 100
- B: 95
- C: 90

K = 4. Solution is item B + item C

**Question**

Suppose we tried to prove the greedy algorithm for 0-1 knapsack problem **does** construct an optimal solution.
The 0-1 Knapsack Problem does not have a greedy solution!

**Example**

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
<th>Weight</th>
<th>Value/Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$300</td>
<td>3 pd</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>$190</td>
<td>2 pd</td>
<td>95</td>
</tr>
<tr>
<td>C</td>
<td>$180</td>
<td>2 pd</td>
<td>90</td>
</tr>
</tbody>
</table>

\[ K = 4. \text{ Solution is item B + item C} \]

**Question**

Suppose we tried to prove the greedy algorithm for 0-1 knapsack problem **does** construct an optimal solution. If we follow exactly the same argument as in the fractional knapsack problem where does the proof fail?