Greedy-Tracks(T,W)// T is set of n gold types/bags  
// W is knapsack weight limit
1. for i = 1 to n,
   \( G_i = \frac{v_i}{w_i} \) // "worth" of type i

2. Sort G in descending order
3. \( m = 0 \) // weight to take of current type
4. \( c = 0 \) // current weight of knapsack
5. while \( (c < W) \)

   \{ \}
   6. get next type, \( i \), from \( G \) ⇒ see example!
   7. \{ \}
   8. if \( w_i + c \leq W \) // taking all of \( c \) does not exceed \( W \)
      \hspace{1cm} \( m = w_i \) // take all
   9. else
      10. \hspace{1cm} \( m = W - c \) // take as much that fits

11. \( x_i = m/w_i \)
12. \hspace{1cm} \( c = c + m \)

More concisely:
- if: \( w_i \leq W - c \) ⇒ \( m = w_i \) \( \geq \) \( \min(w_i, W-c) \)
- else: \( w_i > W - c \) ⇒ \( m = W - c \)

Run-Time: \( n \) types
Sort G: \( O(n \log n) \)
while loop: \( O(n) \) (Step 6 - get next type, \( n \) types)
\( = O(W) \) only if \( c \) (and \( \therefore m \)) increases by \( 1 \) each time, so
( all \( w_i = 1 \) ) \( \therefore \) Time: \( O(\min(n, W)) \)
**Big Picture:**

<table>
<thead>
<tr>
<th>Divide and Conquer</th>
<th>Dyn. Prog.</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mergesort</td>
<td>Change-Making (non-U.S)</td>
<td>Change-Making (U.S)</td>
</tr>
<tr>
<td>Selection</td>
<td>Assem. Line</td>
<td>Frac-KS</td>
</tr>
<tr>
<td>Closest Points</td>
<td>LCS</td>
<td>01-KS</td>
</tr>
</tbody>
</table>

How to know if a greedy algorithm is optimal?

Prove it! (Many ways, we will discuss one)

Use something like an Inductive Proof.

Show 2 properties:

1. Greedy Choice Property - Some optimal solution contains the first choice made by greedy
   
   => "first choice is optimal" ~ Base Case

2. Optimal Substructure Property - Optimal solution to entire problem contains optimal solutions to subproblems.
   
   => "If some sub-solution is optimal, the next sub-solution is optimal" ~ Inductive Step
For 01-KS, saw greedy alg of choosing item with highest \( \frac{v_i}{w_i} \) was not optimal. Which property was lacking?

Recall example:

\[
\begin{array}{c|c|c|c}
W=50 & 1 & 2 & 3 \\
\hline
10 \text{ lbs} & 20 \text{ lbs} & 30 \text{ lbs} \\
\$60 & $100 & $120 \\
\hline
\frac{v_i}{w_i} & 6 & 5 & 4
\end{array}
\]

Greedy first choice: 1
Optimal: 2, 3 \( \Rightarrow \$220 \)

To prove that a greedy algorithm is optimal, prove (1) + (2) via a proof by contradiction:

- **Claim** what we want to prove \( \rightarrow \) Claim!
- Assume opposite of claim \( \rightarrow \) \( \neg \) Claim
- Show that a contradiction occurs: \( \Rightarrow \neg \) Claim is false
  - (e.g., an assumption that opposes \( \neg \) Claim, something obviously wrong \( 1 \equiv 0 \))

\( \therefore \) Claim is true
\[ w = 2 \]
(1) Greedy choice property:

What is the first choice made by greedy alg?

1st choice: take as much as possible $x_i$ ($\frac{10}{10}$) of type 1 ($\frac{1}{w_i}$ is max over all types).

Lemma (1): GreedyFracs has greedy choice property.

Claim: There is some optimal solution that contains $x_i$ of type 1

Proof:

What is $\text{Claim} \Rightarrow \text{Every optimal solution contains } \leq x_i \text{ of type 1} \quad \text{(note that no solution can take } > x_i \text{ since } x_i \text{ is the max we can take)}$?

Suppose by way of contradiction (bwoc) that every optimal solution contains $< x_i$ of type 1 (\text{Claim})

$S^*$ optimal solution that contains $< x_i$ of type 1

How can we show a contradiction?
Main Idea:
Show that if in $S^*$ we take $x_i$ of type 1, we get a better solution than $S^*$.

$\Rightarrow$ not possible since $S^*$ is optimal
First, note we can assume $S^*$ is full.

Why?

If not full, then we can get a better solution by taking more of type 1.

So $S^*$ full: an looks like this:

\[ a \rightarrow \frac{w_a}{w_a} < \frac{v_1}{w_1} \]
\[ b \rightarrow \frac{w_b}{w_b} < \frac{v_1}{w_1} \]
\[ c \rightarrow \frac{w_c}{w_c} < \frac{v_1}{w_1} \]

How to get a better solution than $S^*$?

"Replace some of type $a$ or $b$ or $c$ (or in general, type $k$) with some of type 1."

Can replace some of type $k$ with type 1 to get better solution than $S^*$.

\[ \therefore S^* \text{ is not optimal } \Rightarrow \text{ (contradiction) } \]
\[ \therefore \text{ Some optimal solution must contain } x_1 \text{ of type 1 } \]
\[ \therefore \text{ GreedyFracks has greedy choice prop.} \]
(2) Optimal substructure property

**Lemma (2):** Greedy-KS exhibits optimal substructure property.

"Want to show that an optimal solution contains optimal solutions to sub-problems"

$S^* =$ an optimal solution

**Main Idea:** Show that if we remove all of one type from $S^*$ then we have a solution (sub-solution) that is optimal for some sub-problem.
"Suppose we remove all of type $i$ from $S^*$".

Show that this new solution, call it $S$, is optimal for a sub-problem.

Which sub-problem?

Recall input is: $(T, W)$

Set of types/bags

(Weights, value)

Claim:

Let $S^*$ be optimal solution for types $T=1 \ldots i \ldots n$ and weight limit $W$.

Let $x = x_i \cdot w_i$ denote weight of type $i$ in $S^*$.

If we remove $x$ of type $i$ from $S^*$, we get a solution, $S$, that is optimal for types $T - i$ and (weight limit) $W - x$.

Proof: (bwoc)

Suppose bwoc that $S$ is not optimal for $T - i$, $W - x$. (negate claim)

"What does this mean about $S^*$? Can replace some of $S$ with a different type with higher worth."
"Let's try to do that and show a contradiction"

(Condensed:)

"If $S$ is not optimal can replace some of type (j) in $S$ w/a worthier type"

If $S$ not optimal, then can replace some (or all)
of type j in $S$ with some other type k s.t.

$$\frac{V_k}{W_k} > \frac{V_j}{W_j}$$

$$\begin{array}{c}
\text{(j)} \\
S
\end{array} \rightarrow \begin{array}{c}
\text{better than S} \\
\text{-k}
\end{array}$$

Where's the contradiction?

Then could also replace (type) j with (type) k
in $S^*$ to get a better solution than $S^*$ (to original problem)

$$\begin{array}{c}
\text{(j)} \\
S^*
\end{array} \rightarrow \begin{array}{c}
\text{better than S^*} \\
\text{-k}
\end{array}$$