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:

\[
\begin{align*}
\text{type} & \quad \left\{ \begin{array}{l}
i \to V_i < \frac{V_i}{W_i} \quad \left( \text{How to get a better solution than } S^* \right) \\
a \to \frac{V_a}{W_a} < \frac{V_i}{W_i} \\
b \to \frac{V_b}{W_b} < \frac{V_i}{W_i} \\
c \to \frac{V_c}{W_c} < \frac{V_i}{W_i}
\end{array} \right.
\end{align*}
\]

"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 \( S^* \).

\[
\therefore S^* \text{ is not optimal } \Rightarrow \text{(contradiction)}
\]

\[
\therefore \text{Some optimal solution must contain } x_i \text{ of type 1}
\]

\[
\therefore \text{GreedyFrackS has greedy choice prop.}
\]
(2) Optimal substructure property.

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

Want to show that an optimal solution to entire problem contains optimal solutions to subproblems.

\( S^* = \) an optimal solution to entire problem

**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, values)

Claim:

Let $S^*$ be optimal solution for types $T = 1 \ldots n$ 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: (by o.c.)

Suppose by o.c. that $S$ is not optimal for $T - i$, $W - x$. (7 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."

(condused:)

"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.
  \[
  \frac{V_k}{W_k} > \frac{V_j}{W_j}
  \]

\[
\begin{array}{c}
  S \\
  \Rightarrow \\
  \text{better than } S
\end{array}
\]

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)

\[
\begin{array}{c}
  S^* \\
  \Rightarrow \text{better than } S^*
\end{array}
\]
*Contradiction b/c $S^*$ is optimal solution

\[ \therefore S \text{ must be optimal (claim is false)} \]

\[ \because \text{GreedyFracs has opt. sub. property} \]

Thrm: Greedy Frac-KS is optimal

Proof: By Lemmas 1 and 2

Re-cap

Interesting observation - one change to the problem: discrete vs continuous makes it significantly easier to solve (NP-hard vs poly-time!)
Activity Selection Problem.

Rides at a carnival. Each has start and end times. Want to go on as many rides as possible. (Non-scheduling overlapping events)

Given: A set of n activities (a₁, ..., aₙ)
      sᵢ = start time of activity aᵢ
      fᵢ = finish time of activity aᵢ

2 activities aᵢ, aⱼ are non-overlapping if
"one starts after the other finishes"

sᵢ ≥ fⱼ or sⱼ ≥ fᵢ

Goal: Find a maximum-size subset of non-overlapping activities.

ex.

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>sᵢ</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>12</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>fᵢ</td>
<td>9</td>
<td>13</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>16</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

ex: {a₄, a₈, a₆, 8}; non-overlapping subset
{a₅, a₇, a₈, a₆, 8}; max-size non-overlapping subset

⇒ There may be other optimal solutions (ties)!
Brute Force:
1. enumerate all possible subsets of $A$: $O(2^n)$.
2. eliminate subsets with overlapping activities: $O(2^n)$.
3. find non-overlapping subset of max-size: $O(2^n)$.

Ideas? Not all "obvious" solutions work
Intuition: Don't want to schedule an activity that takes a long time (including waiting for the activity to start).

Main Idea:
At each step choose the activity with the earliest finish time that does not overlap with previous activity.

Notice that other greedy strategies (like choosing the activity with shortest time) don't work. Why not? (May have a late start time)

Greedy Alg. Set $(A, n)$ // $A$ is set of $n$ activities:
1. Sort $A$ in increasing order of finish times $A = a_1, a_2, \ldots, a_n$.
2. current = 1
3. $S = \{a_{\text{current}}\}$ // $S$ is solution.
4. for $i = 2$ to $n$:
   if $S_i \geq f_{\text{current}}$ // $a_i, \text{current}$ don't overlap
      add $a_i$ to $S$