Fractional Knapsack

- Decide \( x_i : 0 \leq x_i \leq w_i \)
- Require \( \sum_i x_i \leq W \)
- Goal: \( \max \sum_i b_i \left( \frac{x_i}{w_i} \right) \)

Suppose \( W = 10 \) and

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight:</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Benefit:</td>
<td>12</td>
<td>32</td>
<td>40</td>
<td>30</td>
<td>50</td>
</tr>
</tbody>
</table>

\( \frac{b_i}{w_i} \) : 3 4 20 5 50

\( x_i \) : 1 2 6 1

blk don carry
Greedy Algorithm for Fractional Knapsack

- Sort by $\frac{b_i}{w_i}$
- For each item in order
  - Take all (if possible) or remaining carrying capacity
- Suppose FSOC solution exists better than ours
- What do we know because of that?
  - $i < k$, $X_i < W_i$, and $X_k > 0$
  - $\frac{b_i}{w_i} > \frac{b_k}{w_k}$
- How do we improve this "better" solution?
  - decrease take of $X_k$, increase $X_i$ equally by
    $\min(W_i - X_i, X_k)$
Review Conversations

- Why don’t these work for fractional knapsack?
  - Sort by weight
  - Sort by benefit

- Why don’t these work for 0-1 Knapsack?
  - Sort by weight
  - Sort by benefit
  - Sort by benefit per unit weight
Example 1: What is the optimal schedule for the following input?

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

\[ \text{Penalty: 0} \]
**Example 2**: What is the optimal schedule for the following input?

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

**Diagram**: Not necessarily optimal schedule.

- $T_1$: 0 penalty (for $T_1$)
- $T_2$: 0 penalty
- $T_3$: 4 penalty

**Penalty (global)** = 4 = $\max(0, 0, 3, 4)$
Possible Scheduling Algorithms

- Sort the jobs by increasing time $t_i$; schedule them in that order.

<table>
<thead>
<tr>
<th>Time:</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>1040</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$:</td>
<td>1050</td>
<td>1050</td>
<td>1050</td>
<td>1040</td>
</tr>
</tbody>
</table>

- Sort the jobs by $d_i - t_i$; schedule them in that order.

    "Slack"
Can we break up tasks?

Is it beneficial to break up tasks? Why or why not?
Proof: Lemma 1

When deciding start times, don’t leave any gaps; $s_{i+1} = s_i + t_i$. 
Proof: Lemma 2

Any schedule that doesn’t agree with our algorithm has at least one pair of consecutive intervals $i, i + 1$ that are inverted relative to our order.

Any diff $\rightarrow \exists i, j \quad i < j$ but $A[i] > A[j]$

if $j = i + 1$, dom
else $k = i + 1$. $i, k$ inverted? idk
if yes, dom

We can now finish the proof

Our Algorithm: Sort the jobs by

Claim: Any schedule with an inversion can be modified to be more like our algorithm’s output without making it worse.

\[ f_i = S_i + t_i \]
\[ f_j = S_i + t_i + t_j \]

\( i, j \) inverted, \( j = i+1 \). But \( d_i \geq d_j \)

Swap \( i, j \) in alt. order

\( f_i' = S_i + t_j + t_i = f_j \)

No worse if \( d_i \geq d_j \)

\( f_j' = S_i + t_j < f_j \)
Proof of Correctness

We proved this:
**Claim**: Any schedule with an inversion can be modified to be more like our algorithm’s output without making it worse.

What does the full proof look like?