Knapsack Problem

Carola Wenk

Slides courtesy of Charles Leiserson with changes and additions by Carola Wenk
Knapsack Problem

• Given a knapsack with weight capacity \( W > 0 \), and given \( n \) items of positive integer weights \( w_1, \ldots, w_n \) and positive integer values \( v_1, \ldots, v_n \).
(So, item \( i \) has value \( v_i \) and weight \( w_i \).)

• **0-1 Knapsack Problem:** Compute a subset of items that maximize the total value (sum), and they all fit into the knapsack (total weight at most \( W \)).

• **Fractional Knapsack Problem:** Same as before but we are allowed to take fractions of items (\( \rightarrow \) gold dust).
Greedy Knapsack

• Greedy Strategy:
  - Compute $\frac{v_i}{w_i}$ for each $i$
  - Greedily take as much as possible of the item with the highest value/weight. Then repeat/recurse.

  $\Rightarrow$ Sort items by value/weight

  $\Rightarrow O(n \log n)$ runtime
Knapsack Example

item | 1 | 2 | 3
value | 12 | 15 | 4 | W=4
weight | 2 | 3 | 1
value/weight | 6 | 5 | 4

• **Greedy fractional**: Take item 1 and 2/3 of item 2
  \[ \Rightarrow \text{weight}=4, \text{value}=12+2/3 \cdot 15 = 12+10 = 22 \]

• **Greedy 0-1**: Take item 1 and then item 3
  \[ \Rightarrow \text{weight} = 1+2=3, \text{value}=12+4=16 \]

• **Optimal 0-1**: Take items 2 and 3, value =19
Optimal Substructure

• Let $s_1, \ldots, s_n$ be an optimal solution, where $s_i$ = amount of item $i$ that is taken; $0 \leq s_i \leq 1$

• Suppose we remove one item. $\rightarrow n - 1$ items left

• Is the remaining “solution” still an optimal solution for $n - 1$ items?

• Yes; cut-and-paste.
Correctness Proof for Greedy

- Suppose items 1, ..., n are numbered in decreasing order by value/weight.

- Greedy solution G: Takes all elements 1, ..., j, ..., i*-1 and a fraction of i*.

- Assume optimal solution S is different from G. Assume S takes only a fraction $\frac{1}{a}$ of item j, for $j \leq i*-1$.

- Create new solution S’ from S by taking $w_j - \frac{1}{a}$ weight away from items $> j$, and add $w_j - \frac{1}{a}$ of item j back in. Hence, all of item j is taken.

$\Rightarrow$ New solution S’ has the same weight but increased value. This contradicts the assumption that S was optimal.

$\Rightarrow$ S=G.
General Solution: DP

- $D[i, w] =$ max value possible for taking a subset of items 1, ..., i with knapsack constraint $w$.

- $D[0, w] = D[i, 0] = 0$ for all $0 \leq i \leq n$ and $0 \leq w \leq W$
  
- $D[i, w] = 0$ for $w < 0$

- $D[i, w] = \max(D[i - 1, w], v_i + D[i - 1, w - w_i])$
  
  - don’t take item i
  - take item i

- Compute $D[n, W]$ by filling an $n \times W$ DP-table.
  
  ⇒ Two nested for-loops, runtime and space $\Theta(nW)$

- Trace back from $D[n, W]$ by redoing computation or following arrows. ⇒ $\Theta(n + W)$ runtime
### DP Example

W = 4

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
<th>Weight</th>
<th>Value/Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Solution:
- Take items 3 and 2

\[
D[i, w] = \max\left(D[i-1, w], v_i + D[i-1, w-w_i]\right)
\]

- don’t take item i
- take item i