0-1 Knapsack — 4th most useful problem (out of 75).

Cave: n bars of gold i.e., n, each with weight w_i, value v_i. (You have) knapsack with weight limit W.

Goal: Choose bars to fill knapsack while maximizing sum of v_i.

Applications:

Choosing:

1. Shares of stocks to invest in w/ max budget.
   - Stocks ~ items (some stocks more valuable).
   - Budget ~ weight limit.

2. Questions to answer on an exam w/in time limit.
   - Question ~ items (some questions worth more points).
   - Time limit ~ weight limit.

3. Ads to place on website with max space.
   - Ads ~ items
   - Space ~ weight limit.

Formally:

For each item i, make choice x_i:

\[ x_i = \begin{cases} 0 & \text{if leaving item } i \\ 1 & \text{" taking " } i \end{cases} \]

Goal: Find x_i for i that maximizes \( \sum_{i=1}^{n} x_i \cdot v_i \) such that: \( \sum_{i=1}^{n} w_i \cdot x_i \leq W \)
Ideas? Does greedy work?

**Greedy:**
Sort by $\frac{v_i}{w_i}$ and choose next item from list.
1. Decreasing value
2. Increasing weight
3. Decreasing value/weight

3-item examples where greedy fails?

(1) $W=10$ lbs

<table>
<thead>
<tr>
<th>10 lbs</th>
<th>5 lbs</th>
<th>5 lbs</th>
<th>4</th>
<th>6</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100$</td>
<td>$90$</td>
<td>$80$</td>
<td>$1$</td>
<td>$1$</td>
<td>$100$</td>
</tr>
</tbody>
</table>

Greedy: $G = $100
Optimal: $OPT = $170

(2) $W=10$ lbs

<table>
<thead>
<tr>
<th>10 lbs</th>
<th>10 lbs</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>$60$</td>
<td>$100$</td>
<td>$120$</td>
</tr>
</tbody>
</table>

Greedy: $G = $160
Optimal: $OPT = $220

(3) $W=50$

$\frac{V_i}{W_i}$ = 6 5 4

Doesn't work!

Optimal substructure property? Yes! $\Rightarrow$ Dynamic Programming
Example: \( W = 5 \), \( n = 4 \) items

<table>
<thead>
<tr>
<th>item</th>
<th>weight</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>40</td>
</tr>
</tbody>
</table>

\[ \text{Solve: items: 2, 4; value = 90} \]

2 factors affect problem size: (1) #items, (2) weight limit

**Dynamic Programming**

1. Size of array/matrix? \( n \times W \) array \( A \).
2. What each entry holds?

\[ A[i,j] = \text{max value from first } i \text{ items with weight limit } j \] (ex. \( A[5,40] \))

\[ \text{For each item } i, \ 1 \leq i \leq n, \ \text{weight limit } j, \ 1 \leq j \leq W, \text{ decide:} \]

- Leave \( i \): \( A[i,j] = A[i-1,j] \)

Best value we can get with the first \( i-1 \) items (and weight limit \( j \)) Value is in \( A[i-1,j] \)
take \( i \)? \( A[i,j] = vi + \max\{A[i-1,j-wi]\} \)

What about the previous \( i-1 \) items? Where in \( A \) can we find the best value for them?

Since we are taking \( i \), previous \( i-1 \) items can weigh at most \( j-wi \).

ex: Suppose deciding on 5\(^{th}\) item \( (i=5) \), weight limit is 15 \( (j=15) \).

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 \\
\checkmark & \times & \checkmark & \checkmark & \checkmark \\
\end{array}
\]

(5 lbs)

already made a decision for each of these, got some total value $100.

If we leave \( i \), value is still $100.

If we take \( i \), total weight of all items must be 15.
So \( i-1 \) items can weigh at most 10.
One more case: What if weight of current item exceeds current weight limit?

ex: 5th item weighs 25 lbs?

⇒ must leave item i!

\[
A[i,j] = \begin{cases} 
0 & \text{if } i=0 \text{ or } j=0 \\
A[i-1,j] \text{ (leave)} & \text{if } i,j>0 \text{ and } w_i > j \\
\max \left\{ A[i-1,j] \text{ (leave)} \right. \\
\left. \max \left\{ A[i-1,j-w_i] \text{ (take)} \right. \right. \\
\left. \text{if } i,j>0, \right. \\
\left. w_i \leq j \right. \right. \\
\left. \text{(diagonal =)} \right. \\
\end{cases}
\]

(4) How to fill \( A \)? For loop \( i=1 \ldots n, j=1 \ldots W \).

(5) Where is value (optimal value)? in \( A[i,n] \).

(6) Run Time? \( O(n \cdot W) \).

(7) Actual solution (optimal items)? (Later).