Knapsack and rounding

Less special special case: values are small integers

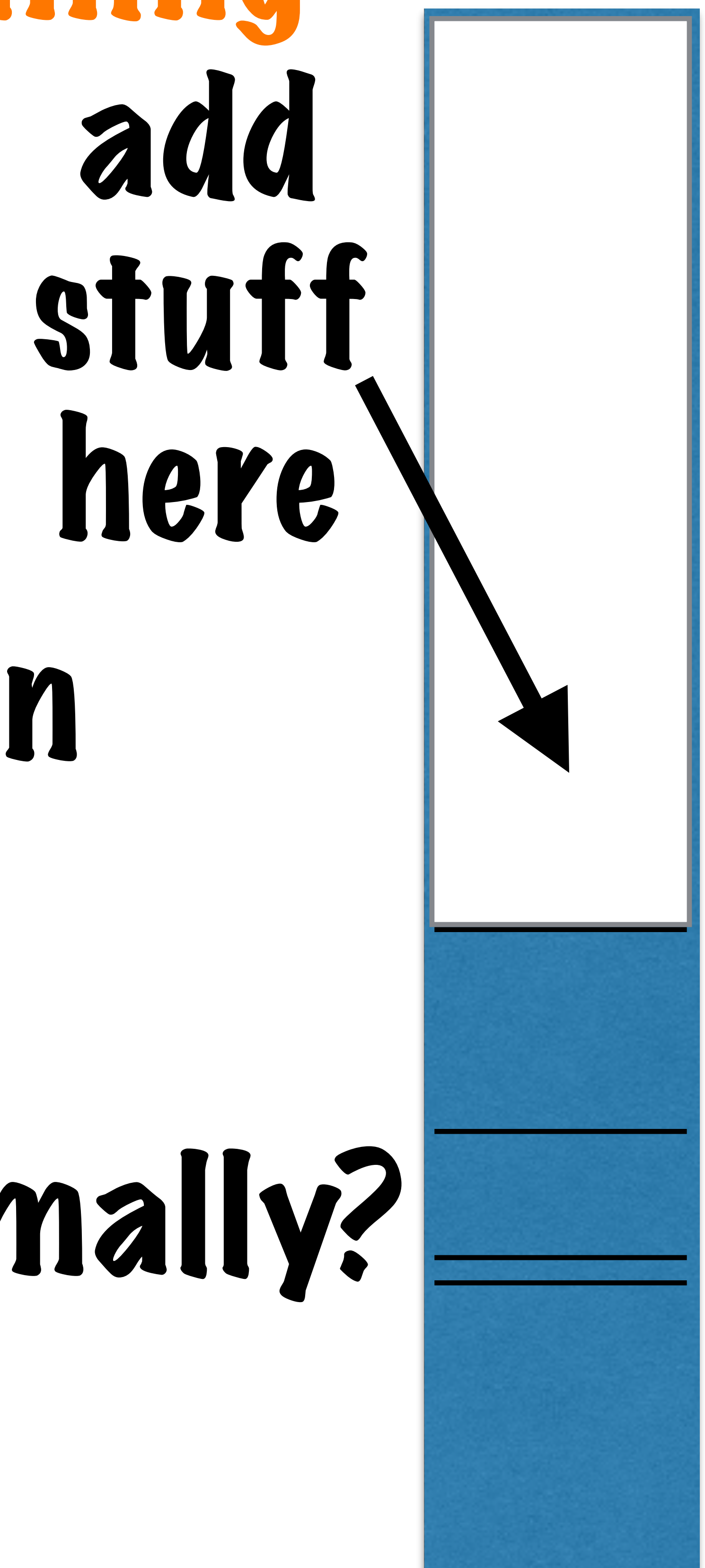
All values

$$\in \{1, 2, \ldots, \mathrm{N}\}$$

N: "small" integer

Extend previous ideas

Dynamic programming



interface

Given partial solution
for first i items,
what to remember
to complete solution optimally?

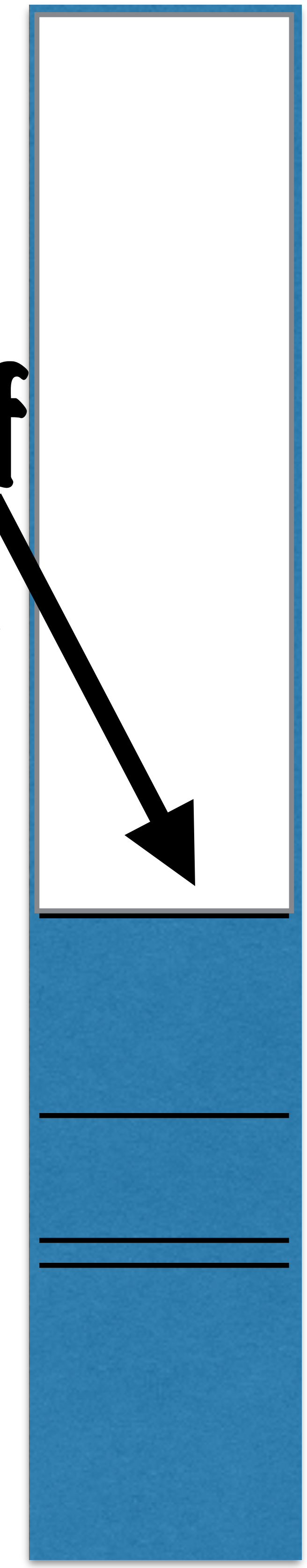
Dynamic programming

Q: What to remember?

A[i,v]=~~whether v achievable with subset of first i items~~

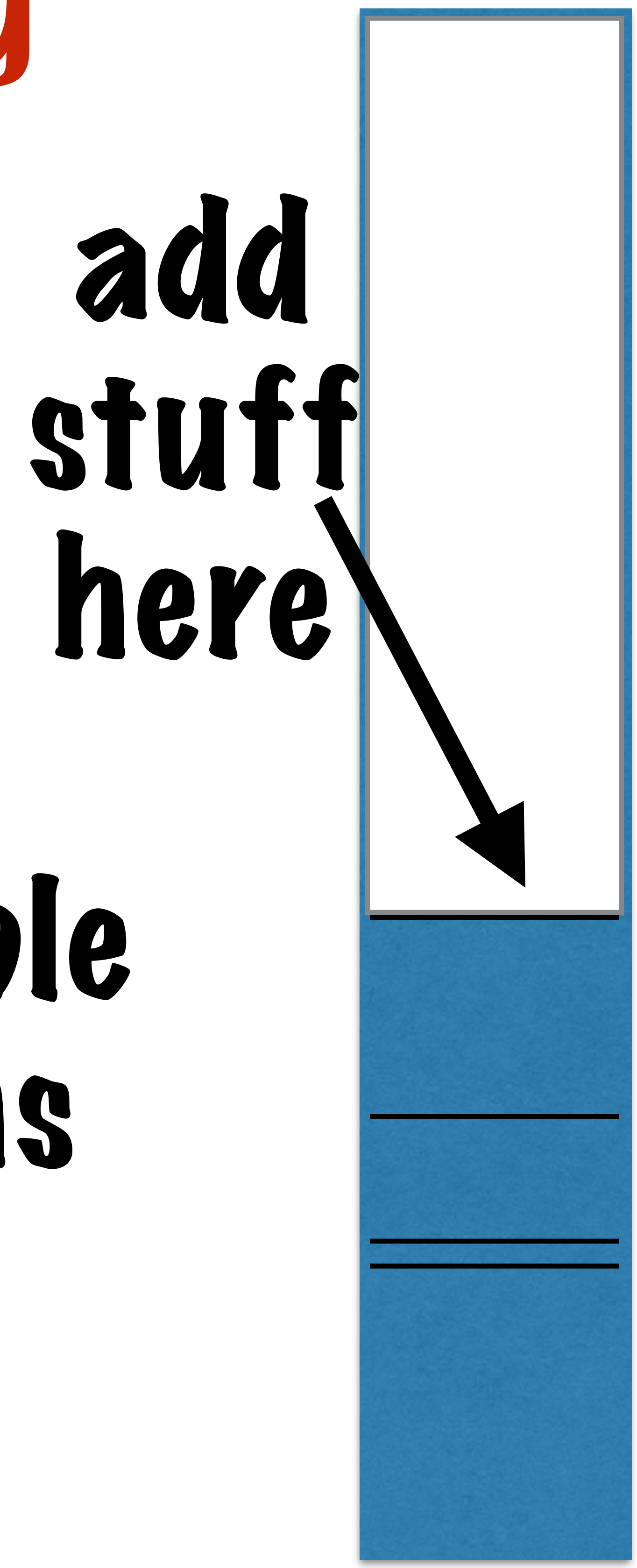
A[i,v]=must remember size

add stuff here



Dynamic programming

Q: What to remember?



A[i,v]=min size achievable for subset of first i items of value v

Q: v,s achievable with subset of first i items iff...
A: ...it depends on **whether** subset contains i

If not:
v,s reached
with first i-1 items

If yes:
$v - v_i, s - s_i$ reached
with first i-1 items

Dynamic program

A[i,v]=min size achievable for subset of first i items of value v

$$\text{If } \mathbf{v} \geq \mathbf{v_i}$$

$$\mathbf{then}\ \mathbf{A}[\mathbf{i},\mathbf{v}] = \min(\mathbf{A}[\mathbf{i}-\mathbf{1},\mathbf{v}],\ \mathbf{A}[\mathbf{i}-\mathbf{1},\mathbf{v}-\mathbf{v_i}]+\mathbf{s_i})$$

$$\mathbf{else}\ \ \ldots$$

$$
\begin{array}{l}
\text{For } v = 0 \dots nN : A[1, v] \leftarrow B + 1 \\
A[1, v_1] \leftarrow s_1, A[1, 0] \leftarrow 0 \\
\text{For } i = 2 \cdots n, \\
\quad \text{For } v = 0 \dots v_i - 1 : A[i, v] \leftarrow A[i-1, v] \\
\quad \text{For } v = v_i, v_i + 1, \dots, nN : \\
\qquad A[i, v] \leftarrow \min(A[i-1, v], A[i-1, v - v_i] + s_i) \\
\text{Output } \max\{v : A[n, v] \leq B\}
\end{array}
$$

Runtime: O(n^2N)

Q: What's the main idea?

For $v = 0 \ldots nN : A[1, v] \leftarrow B + 1$
$A[1, v_1] \leftarrow s_1, A[1, 0] \leftarrow 0$
For $i = 2 \cdots n,$
  For $v = 0 \ldots v_i - 1 : A[i, v] \leftarrow A[i-1, v]$
  For $v = v_i, v_i + 1, \ldots, nN :$
    $A[i, v] \leftarrow \min(A[i-1, v], A[i-1, v - v_i] + s_i)$
Output $\max\{v : A[n, v] \leq B\}$

A: The definition of A[i,v]

dynamic program key step = DP table definition

Knapsack and rounding