**KnapSack**: n objects with
integer weights \(w_1, \ldots, w_n\)
& integer values \(v_1, \ldots, v_n\)
and capacity \(B\)

**Goal**: find subset \(S\) of objects
which maximizes \(\sum_{i \in S} v_i\)
subject to \(\sum_{i \in S} w_i \leq B\)

**Last class**: \(O(nB)\) time solution for version
with one copy of each object.

**Today**: unlimited supply of each object.
First try define subproblems like last class:
\[ K(i, b) = \text{max value achievable using a subset of objects } 1 \ldots i \]
\& capacity \( b \).

Now we can't just decide whether to include object \( i \) or not, we need to decide how many copies to include.

for capacity \( b \),
we can include \( \leq \frac{b}{w_i} \) copies.

So we try \( l = 0 \Rightarrow \left\lfloor \frac{b}{w_i} \right\rfloor \)

if we add \( l \) copies of object \( i \),
we gain value \( lv_i \).
\& then to fill the rest
we use \( K(i-1, b-lw_i) \).

Hence,
\[ K(i, b) = \max_{0 \leq j \leq \left\lfloor \frac{b}{w_i} \right\rfloor} \left\{ K(i-1, b-lw_i) + lv_i \right\} \]
Is there a simpler solution?

Since unlimited supply we don’t need to keep track of which objects we’re used

\[ K(b) = \max \text{ value achievable with capacity } b, \]
\[ \text{all objects } 1, \ldots, n \text{ are allowed} \]

To get the recurrence, try all possibilities for the last object, call it \( l \), that we add \( l \) in. We need that object \( l \) fits, so that means \( w_l \leq b \).

Then, we gain \( v_l \) for it, and we use the best solution for the remaining capacity \( b - w_l \)

So, it’s \( v_l + K(b - w_l) \)

Hence:

\[ K(b) = \max_1^n \left( \sum_{l=1}^n K(b - w_l) + v_l \right) : 1 \leq l \leq n, w_l \leq b \]

In words, \( K(b) \) equals the max of \( K(b-w_l) \) where we maximize over \( l \) such that \( 1 \leq l \leq n \) & \( w_l \leq b \).
$k$ is now a one-dimensional table

\[ k = \begin{bmatrix} 0 & 1 & \cdots & B \end{bmatrix} \]

fill from $b = 0 \to B$

KnapSack WithRepeat $(w_i, \ldots, w_n, v_i, \ldots, v_n, B)$

for $b = 0 \to B$

\[ k(b) = 0 \]

for $l = 1 \to n$

if $w_l \leq b \& k(b) < k(b - w_l) + v_l$

then $k(b) = k(b - w_l) + v_l$

Return($k(B)$)

Running time: $O(nB)$