Two Item Unbounded Knapsack Problem

We are given:

- An item A with size s_A and cost c_A
- An item B with size s_B and cost c_B
- A target size T

Assumptions

Without loss of generality, let A be the *most efficient* item, i.e.,

$$\frac{s_A}{c_A} \geq \frac{s_B}{c_B} \quad \text{or equivalently} \quad \frac{c_A}{s_A} \leq \frac{c_B}{s_B}.$$

Assume a solution exists. Then it must be of the form:

$$T = n_A s_A + n_B s_B$$

for some integers n_A and n_B.

Equivalently, we can rewrite:

$$T - n_A s_A = n_B s_B.$$

Claim

We claim that:

$$n_A = \lfloor T/s_A \rfloor$$

is the largest value such that $T - n_A s_A$ is divisible by s_B.

Proof of Minimal Cost

Let the cost of this solution be:

$$C = n_A c_A + n_B c_B.$$

Assume there exists a different solution with cost:

$$C' = n'_A c_A + n'_B c_B,$$

where $C' < C$, and of course:

$$T = n'_A s_A + n'_B s_B.$$

Since A is the more efficient item, we must have:

$$n'_A > n_A.$$

Proof: From $C' < C$, we have:

$$n'_A c_A + n'_B c_B < n_A c_A + n_B c_B.$$

Rearranging:

$$(n'_A - n_A)c_A < (n_B - n'_B)c_B.$$

Substituting $n_B = \frac{T - n_A s_A}{s_B}$ and $n'_B = \frac{T - n'_A s_A}{s_B}$:

$$(n'_A - n_A)c_A < \frac{(T - n_A s_A) - (T - n'_A s_A)}{s_B} c_B,$$

$$(n'_A - n_A)c_A < (n'_A - n_A)\frac{s_A c_B}{s_B}.$$

Rearranging:

$$(n'_A - n_A)\left(c_A - \frac{s_A c_B}{s_B}\right) < 0. \tag{1}$$

From the assumption $\frac{c_A}{s_A} \le \frac{c_B}{s_B}$, we know:

$$c_A - \frac{s_A c_B}{s_B} \le 0.$$

Thus, for (1) to hold, it must be that:

$$n'_A - n_A > 0,$$

i.e., $n'_A > n_A$.

Contradiction: If $n'_A > n_A$, then we can write:

$$T = n_A s_A + (n'_A - n_A)s_A + n'_B s_B,$$

$$T - n_A s_A = (n'_A - n_A)s_A + n'_B s_B.$$

From the construction of n_A, we know $T - n_A s_A$ is divisible by s_B. Therefore:

$$(n'_A - n_A)s_A + n'_B s_B$$

is also divisible by s_B. Since $n'_A - n_A > 0$, this contradicts the maximality of n_A.

Conclusion: The solution with $n_A = \lfloor T/s_A \rfloor$ and $n_B = \frac{T - n_A s_A}{s_B}$ minimizes the cost.

Algorithm

The algorithm to find the solution is as follows:

```
nA = floor(T / sA)
while (T - nA * sA) is not divisible by sB:
    nA -= 1
nB = (T - nA * sA) / sB
```

Complexity

If no solution exists, the loop will run more than s_B times. Hence, the time complexity is $O(s_B)$.

Note that this is not yet sufficient to prove that this is poly-time as this is still only pseudo-polynomial time.

Making it Log

However, given a GCD, Let $F = gcd(s_A, s_B)$

$$T = n_A s_A + n_B s_B$$

$$T/F = n_A s_A/F + n_B s_B/F$$

We can scale all the solutions by their GCD and still be valid. Notably, after scaling, s_A, s_B must be coprime. (if d divides s_A and s_B then d must divide T. If it does not there is no solution)

Hence the goal is to solve for the largest n_A such that $T - n_A s_A$ is divisible by s_B where s_A, s_B coprime. Or equivalently find the largest n_A

$$T - n_A s_A = 0 \mod s_B$$

$$n_A s_A = T \mod s_B$$

Since s_A, s_B are coprime, there exists a multiplicative inverse s_A^{-1} which we can solve for with the extended euclidean algorithm:

$$s_A s_A^{-1} = 1 \mod s_B$$

We can solve for a base solution b:

$$b = T s_A^{-1} \mod s_B$$

$$b s_A = T \mod s_B$$

However, to maximize the solution we need to be able to keep adding 0 mod s_B. As s_A and s_B coprime, the only solution is:

$$s_B s_A = 0 \mod s_B$$

So combining everything, the maximizing n_A is:

$$n_A = b + \text{floor}(\frac{T - x s_A}{s_A s_B})$$

The complexity of this is now about $O(\log(T))$

Quicker Nonconstructive Proof

This can be written as a fixed size integer linear program which has solution complexity poly in size (number of bits) of inputs.