Knapsack Problem

Dateng Lin

Bloomberg LP

Nov 9, 2018

The problem

There are N kinds of items and a knapsack with capacity V. The cost of putting the i-th kind is C_i and the weight of item is W_i. We want to determine which items to select to put into the knapsack in order to maximize total weight.

1. There is only one item in the i-th kind: 0/1 knapsack.
2. There are infinite number of items in the i-th kind: complete knapsack.
3. There are k_i $(k_i \geq 1)$ items in the i-th kind: multiple choice knapsack.

0/1 knapsack

$opt[i][j]$: the maximum weight when we select from the first i items and the maximum weight is not beyond j.

$$opt[0][j] = 0, 0 \leq j \leq V,$$

$$opt[i][j] = \max(opt[i-1][j], opt[i-1][j-C_i] + W_i),$$
$$1 \leq i \leq N, C_i \leq j \leq V.$$

0/1 knapsack

The relationship is only between i and $i-1$, hence:

$opt[i\%2][j] = \max(opt[(i-1)\%2][j], opt[(i-1)\%2][j-C_i] + W_i)$.

However actually, we do not even need two lines.
If we enumerate j from V to C_i, we can reuse the status of $i-1$, hence:

$$opt[j] = \max(opt[j], opt[j-C_i] + W_i),$$

with j counting from V to C_i.

0/1 Knapsack

There might be no way to reduce the overall level of time complexity. However, we can observe
$opt[N][V]$ depends on $opt[N-1][\max(V - C_N, 0)]$,
$opt[N-1][\max(V - C_N, 0)]$ depends on
$opt[N-2][\max(V - C_N - C_{N-1}, 0)]$
...
$opt[i+1][\max(V - \sum_{k=i+2}^{N} C_k, 0)]$ depends on
$opt[i][\max(V - \sum_{k=i+1}^{N} C_k, 0)]$,
so we can choose the lower bound of the inner loop to be
$\max(V - \sum_{k=i+1}^{N} C_k, C_i)$.

Complete Knapsack

It is obvious we have

$$opt[i][j] = \max(opt[i-1][j-k*C_i] + k*W_i),$$

$$1 \leq i \leq N, k*C_i \leq j \leq V,$$

but is this really all?

Complete Knapsack

We can do a math trick:

$$\begin{aligned} opt[i][j] &= \max(opt[i-1][j-k*C_i]+k*W_i) \\ &= \max(opt[i-1][j], \max(opt[i-1][j-C_i-(k-1)*C_i] \\ &+ (k-1)*W_i+W_i)) \\ &= \max(opt[i-1][j], opt[i][j-C_i]+W_i), \end{aligned}$$

and k disappeared! Moreover, since this time we want to reuse the status of i, we have

$$opt[j] = \max(opt[j], opt[j-C_i]+W_i),$$

with j counting from C_i to V. The opposite of 0/1 Knapsack!

Multiple Choice Knapsack

What about multiple choice version?

$$opt[i][j] = \max(opt[i-1][j - k * C_i] + k * W_i),$$

$$1 \leq i \leq N, k * C_i \leq j \leq V, 0 \leq k \leq k_i.$$

There is no math trick like the case of complete knapsack since we do not know the range of every k_i.

One step back, this can also be treated as 0/1 knapsack if we do it one by one. Can we optimize based on this?

Multiple Choice Knapsack

The key is to decompose k_i while at the same time maintain the ability to represent 0 to k_i. A common technique is using binary/bit.

Let n be the largest integer such that

$$\sum_{m=0}^{n} 2^m < k_i,$$

and we have

$$1, 2, 4, ..., 2^n, k_i - 2^{n+1} + 1$$

to be able to represent 0 to k_i (Apparently this is true for numbers between 0 and $2^{n+1} - 1$, for number x between 2^{n+1} and k_i, just consider $k_i - x$). And we only need to consider $O(\log k_i)$ items.

Multiple Choice Knapsack

No. The former method is not the end :-)
There is a crazy trick from an internal training material when I studied ACM/ICPC, which is of complexity $O(NV)$:
Let $j = q * C_i + r$, then

$$\begin{aligned} opt[i][j] &= \max(opt[i-1][(q-k) * C_i + r] + k * W_i), \\ &= \max(opt[i-1][u * C_i + r] - u * W_i) + q * W_i, \end{aligned}$$

$$q - k_i \le u \le q,$$

then for every i, classify 0 to V by r into C_i groups, and compute the maximum within the same group with length restriction to be k_i, which is the sliding window maximum problem and thus can be solved in linear time.