0 - 1 Knapsack Problem

Niu Yunpeng A0162492A

1 Problem

Given n items, each has a weight and a value; and a bag that can hold up to *W kg*. How do you put items in the bag to maximize the total value, but does not exceed weight *W*. Return this value.

2 Recurrence Function

1. Relationship

$$V[n, w] = max(V[n-1, w], value[n] + V[n-1, w - weight[n]])$$

2. Explanation

Assume that there are n items, and we traverse them one by one as [1,..,n]. For each item, we have only two choices, to *put it in the bag* or *not put it in the bag.* So let's consider these two situations seperately:

To make things easier, we consider the last item first, that is, the item with index n, and with *value[n]* and *weight[n]*.

1) If we put it the bag, then the other items should have a total weight of no more than *W* - *weight[n]* kg. And the total value of the bag is the value of this item plus the total value of the other items.

2) If we do not put it in the bag, then the other items should have a total weight of no more than W kg. And the total value of the bag is the total value of the other items.

Therefore, for the first condition, the problem changes from $V[n, w]$ to $V[n-1, w - weight[n]]$, since the other items should be chosen from the first *n - 1* elements in the array; for the second condition, the problem changes from $V[n, w]$ to $V[n-1, w]$. For both, the scale of the problem has decreased. Therefore, use some wishful thinking, we believe this problem can be solved. Hence, the recurrence function is proven to be correct.

Furthermore, the initial condition should be addressed. When $n = 0$, that means there is no item to choose, so *V[0, w]* should always be 0; when $w = 0$, the capacity of the bag is 0. In other words, the bag cannot contain any item, so *V[n, 0]* should always be 0. Additionally, it should return 0 when $weight[n] > w$ as well. This is because the bag cannot hold another item into it at this time.

3 Implementation

Please see Knapsack.java attached for a java implementation.

— End —