问题描述：

有n个物品，每个物品有对应的重量和价值。

|  |  |  |
| --- | --- | --- |
| 编号 | 重量 | 价值 |
| 1 | w[1] | v[1] |
| 2 | w[2] | v[1] |
| 3 | w[3] | v[3] |
| … | … | … |
| n | w[n] | v[n] |

现在有一个可以承重m的背包（比较喜欢李宁的包）。求一组物品的组合（无重复），能使这组物品能全部放入包中（总重不超过m）而使包中物品总价值最大。

典型的动态规划题。用贪心算法是得不到最优解的。

问题分解：

构建一个数组value[n+1][m+1]，value[i][j]的意义是：前i个物品的组合去装入载重量为j的背包能达到的最大价值。

我们先来把问题集中在这个数组上，关注这个迭代关系。

for(i=1;i<row;i++)

{

for(j=1;j<col;j++)

{

//前i-1个物品中的组合+物品i: 这一组合装入载重为j的包所能够达到的最大价值

int temp = value[i-1][j-w[i]] + v[i];

//w[i]<=j,且第i个物品装入背包后的价值>value[i-1][j],则更新当前最大价值

//否则第i个物品不装入背包

if(w[i]<=j && temp>value[i-1][j])

value[i][j]=temp;

else

value[i][j] = value[i-1][j];

}

}

当上面这个数组的内容全部确定之后，很明显value[n][m]是最大的价值量。那么这个物品组合是什么呢？可以逆推回来，如果value[n][m]>value[n-1][m]，则第n个物品肯定是被装入背包的，且前n-1个物品被装入载重量为m-w[n]的背包中。我们设一个c[n+1]，若物品i要被放入背包中则c[i]=1，否则c[i]=0。

for(i = n,j = m;i > 0;i--)

{

if(value[i][j]>value[i-1][j])

{

c[i]=1;

j = j - w[i];

}

}

解决了上面上个问题后，该问题已经解出了。只需要在循环前做好初始化工作，令value数组的值全为零。

本博文介绍的问题中，每个物品只可以选一次，选中状态只会是0-1，因此称为是0-1背包问题。如果物品的选中次数是可以大于1且有具体限定的，则是有限背包问题；如果物品的选中次数是无限制的，则是无限背包问题。