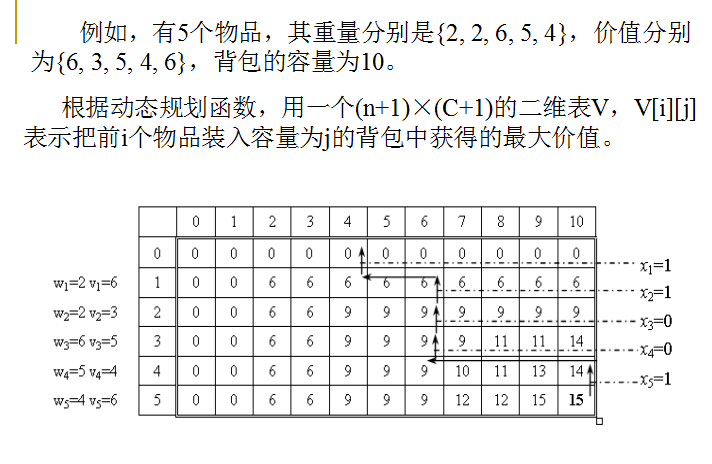
**0-1背包问题属于动态规划的范畴，动态规划解决问题的核心在于填表。**



**（1）01背包问题本质上是穷举背包容量和可供选择的物品(意思是里面的物品可能会放进背包，可能不会放进背包)，取得最优解，只不过在穷举的过程中，会根据状态转移方程，只计算可能获得的最优解的部分，不去计算不是最优解的部分。具体来看，解题思路是把该问题分解为一个一个的小问题，一步步的通过小问题的最优解，最终得到大问题的最优解，跟我们人脑解题的思路是一样的。**比如第一个小问题是“当我的背包承重M=1，只有编号为a的物品可供选择时，最优解是什么”，然后下一个小问题是建立在前一个小问题的基础上“当我的背包承重M=1，有编号为a,b的物品可供选择时，最优解是什么”，以此类推。

**（2） 为什么能列出状态转移方程？是因为每个状态的最优解，都是根据之前的状态的最优解获得的。**具体到背包问题，有以下几点：

　　a) 当物品备选情况（物品备选情况指：可供选择的物品的集合）一致时，背包容量M越大，那么sum\_v一定大于等于原来的值。

　　b) 背包容量M确定时，可供选择的物品N越多，那么sum\_v一定大于等于原来的值。

　　c) **由a)和b)可得，sum\_v的最大值就是当M和N取到最大值时的sum\_v**

　　c) 从思路上说，01背包问题有两个维度：背包容量M，和供选择物品数N。编程的本质是实现人类解决现实问题的思路。仔细想想，如果不借助计算机，你该如何解决这个问题？答案是，例如考虑M=1时，先考虑a能否放入背包，取得最大值，再考虑a和b能否放入背包（a和b都是备选，最终放入背包的可能是a，可能是b，也可能是ab），这时因此与之前只考虑a的情况相比，多了一个b，所以：

* 要先判断b能否单独放进背包：
* 如果不能，那么备选为a,b时最大值，等于备选只有a时的最大值（因为b是放不进背包的）。
* 如果能，即b能够放进去，还有两种可能（即将b放进背包，和不将b放进背包），对这两种可能性，要取最大值：
* 最终将b放进去(注：此时物品a是否被放进背包是未知的，原因是：剩余的背包容量可能不足以放进物品a，即要在剩余可选物品里找出最优解。
* 最终没有将b放进去（因为后面可能有比b更合适的物品放进去），此时最大值等于备选只有a时的最大值

0-1背包问题的公式

v(i,j)= (1)v(i-1,j) j<wi;

(2)max{v(i-1,j),v(i-1,j-wi)+vi} j>wi;